Multitask variational autoencoders

Author: Edvards Zakovskis
Supervisors: Nello Blaser, Kristian Gundersen

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

May, 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 13th May, 2024

Contents

List of Figures

Chapter 1

Introduction

This chapter provides an overview of the research topic and the motivation behind the research in this thesis. In the first section, I will talk about the motivation behind the 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 [9, 10, 19, 14]. However, VAEs, given a limited set of data and compute power, have limitations and challenges, which are described below.

One of the key challenges traditional VAEs face is that they tend to produce blurry, over-smoothed samples. This is due to the fact that the VAE objective is to maximize the evidence lower bound (ELBO) which is a trade-off between the reconstruction accuracy and the KL divergence between the approximate posterior and the prior. Also, the prior distribution is often chosen to be a simple distribution such as a standard normal

distribution, which can limit the expressiveness of the latent space and the generative capabilities of the model [10].

More recently Vector Quantized Variational Autoencoders (VQ-VAEs) have been introduced to address VAE's limitations. There are two key differences between VQ-VAEs and VAEs. One is that the latent space of VQ-VAEs is discrete, which is achieved by Vector Quantization (VQ). The other is that the prior distribution is learned instead of being assumed to be static. This allows the model to capture the underlying structure of the data more effectively, which allows VQ-VAEs to capture fine-grained details in the data, whilst maintaining the interpretability of latent representations and generational properties of VAEs. The prior distribution is learned by training a separate model called PixelCNN, which is trained to model the prior distribution of the latent space and thus achieve the ELBO objective with two models instead of one. VQ-VAEs have been shown to produce samples with higher fidelity and have been since used in a variety of applications such as high-resolution image generation, text-to-image generation and speech synthesis [19, 15, 14].

The VQ-VAE architecture solves the problem of capturing fine-grained details in the data, whilst maintaining the interpretability of latent representations and generational properties of VAEs. Although VQ-VAEs are more effective in capturing fine-grained details in the data, it is still challenging to capture all the sources of variation in the data in a single model. This is especially true when the data has multiple sources of variation, such as in the case of images, where the data can have multiple types of objects, backgrounds, and lighting conditions [10, 19, 15]. One potential approach to address this issue could involve employing multitask learning, where the model is trained to tackle numerous tasks simultaneously.

Semi-conditional Variational Autoencoders (SC-VAEs) are a type of VAEs in which the decoder distribution is conditioned on a specific subset of the input data. While SC-VAEs sacrifice the ability to efficiently generate samples solely from the latent space, they retain the capability to generate samples based on the provided subset of input data. This additional information is then used to reconstruct the input data, which has been shown to be useful in the context of reconstructing very sparse data and uncertainty quantification [5].

Multitask learning is a paradigm that aims to improve model generalization and performance by simultaneously learning multiple related tasks. The concept of combining VAEs with multitask learning has undergone some experimental exploration in the research community, although it has not been thoroughly investigated [13]. Multitask

Variational Autoencoders (MT-VAEs) extend the VAE model to take advantage of the extra information or tasks that 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 [3].

The research in this thesis aims to investigate the effectiveness and potential of integrating semi-conditional and standard VAEs within a single model through multitask learning. The investigation will explore the potential of this approach in the context of both standard VAEs and VQ-VAEs. The research will introduce two methods for combining semi-conditional and non-conditioned VAEs:Multidecoder method and Single decoder method.

In Multidecoder method, the semi-conditional and non-conditioned VAEs are combined with two decoders, where one decoder is conditioned, and the other is not. In Single decoder method, the semi-conditional and non-conditioned VAEs are combined with a single decoder, which can be conditioned or non-conditioned by masking the input data.

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 performing a thorough 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 that can be combined with standard non-conditioned VAEs through multitask learning. My objective is to determine whether my proposed methods:Multidecoder method and Single decoder method which leverage multitask learning can enhance the performance of VAEs compared to traditional ones, and if so, under what conditions. Additionally, I will explore the applicability of these methods to VQ-VAEs.

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

- How can SCVAEs be combined with standard VAEs through multitask learning?
- Can the combination of semi-conditional and non-conditional VAEs be used to improve the reconstruction accuracy and generalization capabilities of VAEs on non-conditioned data?
- Could the same approaches be applied to VQ-VAEs?
- What are the limitations of these methods?

1.3 Thesis Outline

Chapter 2

Background

In this chapter, I am going to introduce the concepts that are necessary to understand the research presented in this thesis. The chapter is divided into four 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 the concept of semi-conditioned VAEs. The fourth section introduces the concept of multitask learning. The chapter concludes with a section that delves into additional concepts that are necessary to understand the research presented in this thesis.

2.1 VAEs

Variational Autoencoders (VAEs), first introduced in 2013 by Kingma and Welling [9], 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 [10].

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 [9].

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 $p(z)$
- $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ is the reconstruction term

The Kullback-Leibler divergence term encourages the encoder (posterior) distribution to align with the prior distribution, acting as a regularization term. This alignment is crucial for an effective utilization of the latent space. On the other hand, the reconstruction term encourages the decoder to reconstruct the input data as accurately as possible, which encourages the decoder to accurately capture the data distribution.

While the expression of the ELBO might appear different from the original formulation, it remains equivalent, merely using different terms to express the same notion [10].

Sampling from $q_\phi(z|x)$ enables the Monte Carlo estimation

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

where $z^{(l)} \sim q_\phi(z|x)$ and L is the number of samples and the first term is the Kullback-Leibler divergence term and the second term is the reconstruction term [10].

The individual data point ELBO and its gradients are in general intractable to compute. However, unbiased estimates of the ELBO and its gradients can be obtained using the reparameterization trick, which is described in the next section [10].

2.1.1 The reparameterization Trick

The reparameterization 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 [10].

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 expectation 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 [10]. As a result, the gradients the expectation and gradient operators become commutative, and there can be formed a Monte Carlo estimate of the gradients

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

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

$$\simeq \frac{1}{L} \sum_{l=1}^{L} \nabla_\phi f(g_\phi(\epsilon^{(l)}, x))$$

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

Gradients of the ELBO

When applying the reparameterization 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

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

$$= E_{p(\epsilon)}[\nabla_{\phi,\theta}[\log p_\theta(x, g_\phi(\epsilon, x)) - \log q_\phi(g_\phi(\epsilon, x)|x)]]$$
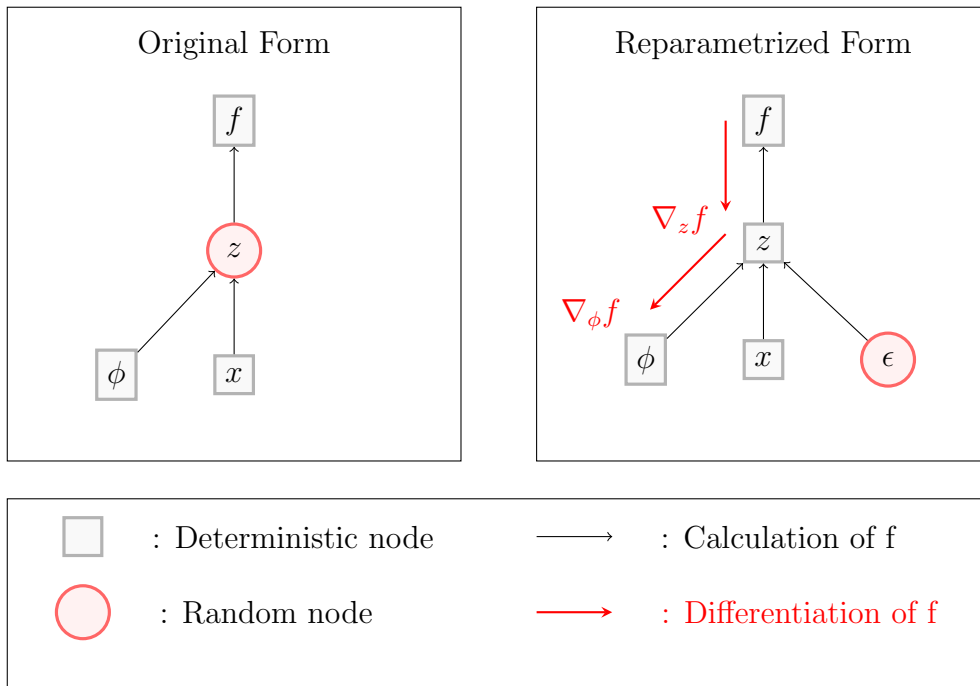
Figure 2.1: Illustration diagram of the reparameterization trick. The input of a function f is x. The parameters θ affect the objective of the function f through a random variable z. In the original form, we can not compute the gradients $\nabla_\phi f$, because direct backpropagation is not possible through a random variable. In the reparameterized form, the randomness is separated 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 [10]

.

Credit: Adapted from Kingma and Welling[10]

$$\simeq \frac{1}{L}\sum_{l=1}^{L}\nabla_{\phi,\theta}[\log p_\theta(x, g_\phi(\epsilon^{(l)}, x)) - \log q_\phi(g_\phi(\epsilon^{(l)}, x)|x)]$$

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 [10].

2.1.2 Gaussian VAEs

Although 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 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 reparameterization 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 VAEs usually used in practice 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

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

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 and the Z is the dimension of the latent space [10, 8]. The second term in the function is derived from simplifying 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.

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

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

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

Figure 2.2: 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 [10]

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. [19]. The VQ-VAEs have shown various improvements over the standard VAEs, such as higher quality of the generated samples, 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 [15, 4, 6, 14].

The VQVAEs fundamentally differ in two key ways from VAEs. Firstly, the latent representation is discrete instead of continuous. Secondly, the prior distribution is learned 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 [19]. VQ-VAE learning process consists of two stages. In the first stage, the encoder and the decoder are trained. In the second stage a prior over these discrete latent variables is trained [19].

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 naturally is a stream of discrete units, such as words or phonemes. Images can be often well described by language, which can the discrete representations well-suited for images as well. Moreover, discrete representations work very well with complex reasoning, and decision-making [19].

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 neighbor 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 [19].

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 $\hat{x}$ [19]. 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 [19].

Since the proposed posterior distribution $q_\phi(z|x)$ is deterministic by applying it to the 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 the posterior distribution is **maximized**, which means the encoder and the decoder are trained with the prior distribution being arbitrary. The training objective in the first stages was reduced 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 [19].

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-differentiable function, and copy the gradients of $z_q(x)$ to $z_e(x)$ [19]. 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. [22]

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 commitment loss [19].

However, since the embedding space can be arbitrarily large the embedding vectors can be arbitrarily 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 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 [19].

Thus, the resulting training objective becomes

$$L = \log p_\theta(x|z = k) - \left(||sg(z_e(x)) - e_k||^2 + \beta||z_q(x) - sg(e_k)||^2 \right), \qquad (2.5)$$

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

The loss function to be **minimized** for VQ-VAEs 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

$$L = \frac{1}{D} \sum_{i=1}^{D} ||x_i - \hat{x}_i||^2 + \frac{1}{Z} \sum_{i=1}^{Z} \left(||sg(z_e(x))_i - e_{k_i}||^2 + \beta||z_q(x)_i - sg(e_{ki})||^2 \right),$$

where $\hat{x} = f_\theta(z_q(x))$, where function f_θ is the decoder network, D is the dimension of the input data, Z is the number of latent space vectors and k_i is the index of the nearest embedding vector in the embedding space for the i-th latent space vector, which is defined in equation 2.3 [19].

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 [19].

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
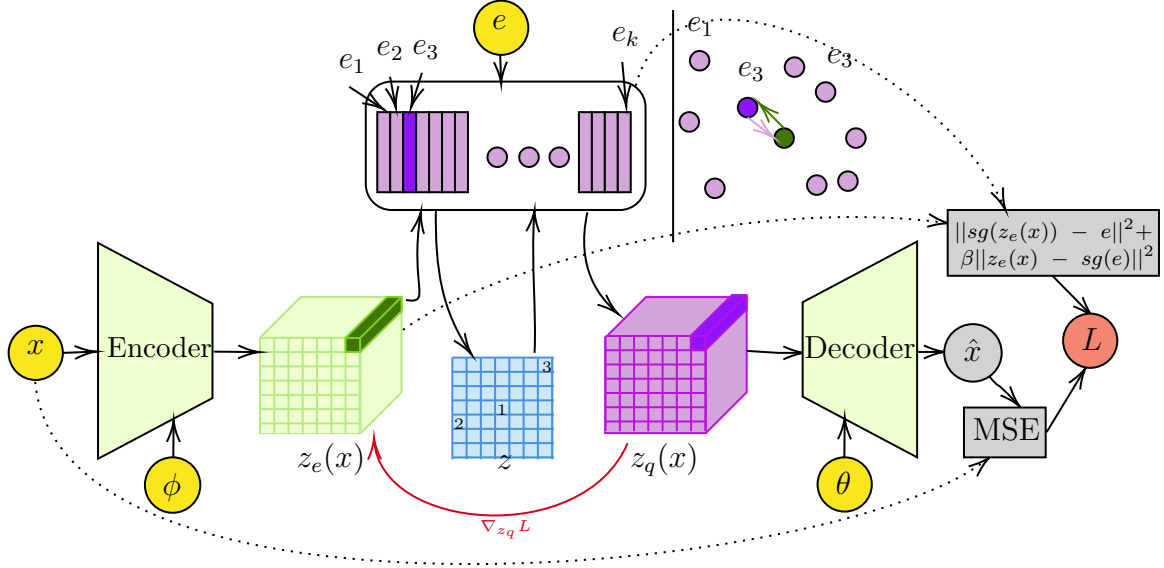
Figure 2.3: Architecture of VQ-VAEs. The input x is passed through the encoder convolutional neural network producing the output $z_e(x)$. For each output vector in $z_e(x)$, the nearest embedding vector in the embedding table e is found. The indices of the nearest embedding vectors are then used as 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 commitment and codebook loss encourage both the encoder output vector and the nearest embedding vector to be close to each other [19].

Credit: Adapted from Aäron van den Oord et al. [19].

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 subsection 2.5.1 [19].

2.3 Semi-Conditioned VAEs

Semi-conditional VAEs were first introduced in 2020 [5]. Semi-conditional VAEs (SC-VAEs) are a variant of VAEs that were first designed for the reconstruction of non-linear dynamic processes based on sparse observations. The semi-conditional VAEs extend the standard VAEs framework 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 information.

The additional information m can be any type of information that is available at the time of the reconstruction and can be used to improve the reconstruction of the data. In the original paper, the authors used the SCVAEs to reconstruct fluid flow data. The additional information m was the sparse measurements of the flow data. The method was showcased on two different datasets, velocity data from simulations of 2D flow around a cylinder, and bottom currents [5].

Natural applications for the SCVAEs are related to domains where there are often sparse measurements, such as environmental data. However, the SCVAEs can also be used, for instance, in computer vision to generate new images based on sparse pixel representations [5].

The semi-conditional property of the SCVAEs could also be applied to the VQ-VAEs, which has not been explored yet in the literature. Also, the potential of combining non-conditioned and semi-conditioned VAEs through multitask learning has not been explored yet.

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 traditional single-task

learning, where each task is learned independently, multitask learning allows taking advantage of task relationships and learning a shared representation that is useful for all tasks [3].

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 in scenarios where the data is limited, which is often the case for medical data [1].

For example, in medical image analysis, in a scenario where the task is to develop a system for identifying and classifying different types of abnormalities in medical images. Traditionally this could be done by training a separate model for each type of abnormality. However, this approach is not optimal due to limited data and the shared features that could be used to identify and classify multiple types of abnormalities [3].

Multitask learning is an approach in which a single model is trained to perform multiple tasks. In the context of medical image analysis, this could involve training multitask VAE or VQVAE models capable of simultaneously learning to reconstruct images, whilst also learning to classify different types of conditions.

Multitask learning has been shown to be most effective when the tasks are related and data is limited. The potential and flexibility of multitask learning for improving the model performance make it a very valuable technique in machine learning. It has an unexplored potential in the context of VAEs and VQ-VAEs, which is the main motivation for this thesis.

2.5 Additional concepts

2.5.1 PixelCNN

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 pixel, 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 [18, 17].
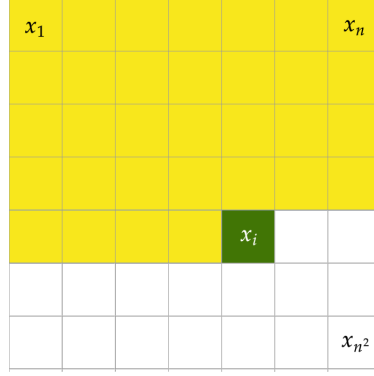
Figure 2.4: Conditional generation in autoregressive models. The model generates the pixels of the image one at a time, conditioning the previous pixels. The model is autoregressive because the distribution of each pixel is conditioned on the previous pixels.

Autoregressive models treat an 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 sequentially from the conditional distribution of the pixel given the previous pixels, which is shown in the figure 2.4 [18].

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 the 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 [18, 16, 2].

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 [19].

2.5.2 Random number generation using Power law distribution

Power law distribution in probability theory and statistics is a distribution in which one variable is proportional to the power of the other, i.e., $p(x) \propto x^\alpha$, where x is the random variable and α is the exponent parameter, where α determines the shape of the distribution. The power law distribution is a heavy-tailed distribution, which means that the probability of the large values is higher than in the normal distribution. The power law distribution is used in various fields, such as physics, biology, economics, and computer science [21].

One example of the power law distribution could be the function $f(x) = \alpha x^{\alpha-1}$ for $0 \leq x \leq 1$, $a > 0$. One advantage to this distribution is that this distribution has a finite range and is easy to scale to any range. This fact will be used when power law distribution is used to sample the number of pixels to be conditioned on in the semi-conditional VAEs.

2.5.3 SoftAdapt: Adaptive loss weighting

SoftAdapt is a novel approach to address the challenge of dynamically adjusting weights for multipart loss functions in deep learning. In machine learning, multipart loss functions are common, wherein the loss function is composed of a sum of loss terms. Traditionally, the loss terms of multipart loss function are weighted equally, or their weights are determined through heuristic methods. Whilst this approach can work well in practice, it is not optimal because the importance of the loss terms can vary during the training process and the optimal weights can be hard to determine [7].

The approach of the SoftAdapt algorithm which was published in 2020 is to mathematically determine the optimal weights of the loss terms given a short loss history. By

evaluating the rate of change for each loss term, it tries to dynamically learn the optimal weights for the loss terms and adapt them during the training process [7]. The SoftAdapt algorithm since its publication has been shown to be effective in various machine learning applications, such as generative models, model compression and physics-informed neural networks.

Chapter 3

Methods

This chapter will outline the methods of how SCVAEs and non-conditioned VAEs are integrated using multitask learning in this thesis, which will be explored in the context of images. The chapter will be divided into three sections. The first section will describe the conditioning strategies, which will be used to condition the VAEs. In the following section, I will describe the method of merging SCVAEs with non-conditioned VAEs using two decoders, where one is conditioned, and the other is not. This method will be referred to as Multidecoder method. In the following section, I describe another method of merging SCVAEs with non-conditioned VAEs by utilizing the same encoder and decoder for both models by employing novel training strategies, which will be referred to as Single decoder method. Both methods will be applied to both Gaussian VAEs and VQ-VAEs.

3.1 Conditioning information

In this thesis, I will obtain the conditioning information m from the input image x by sampling pixels some pixels from the input image x. In this process, the first step is to obtain a mask which will be used to sample the pixels from the input image x. The mask will be a binary matrix of the same size as the input image x, where a value of 1 represents the sampled pixels and a value of 0 represents the pixels that are not sampled. The decision of how many pixels to sample and which pixels to sample will be described in the following sections for each method.

After the mask is obtained, the mask is applied to the input image x to obtain the masked image, which is then used to obtain the conditioning information m by concatenating the mask with the masked image, which can be seen in Figure 3.1.
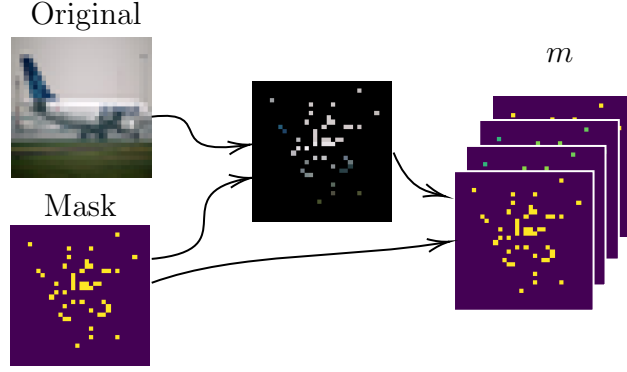
Figure 3.1: The illustration of the process of obtaining the conditioning information m from the input image x and the mask. The process to obtain the conditioning information m consists of two steps: the first step is to obtain the masked image from the input image x and the mask, and the second step is to add the mask to the masked image to obtain the conditioning information m. The conditioning information m is then used to condition the decoder.

In both proposed methods, the conditioning information m will be incorporated into the decoder by first flattening it and then concatenating it with the input of the decoder and then using a fully connected layer before passing it through the transposed convolutional layers of the decoder.

3.2   Multidecoder method

In the first method, the VAE framework is expanded with a second decoder $p_\xi(x|z, m)$, which is conditioned on an additional property m of the input data. This approach involves utilizing two decoders within the VAE framework: one for reconstructing the input data solely based on a trained neural network with Multidecoder method latent variable z, and the other for reconstructing the input data based on both the latent variable z and the additional conditioning information m.

3.2.1   Conditioning strategy

In this method, the conditioning information m is acquired by sampling a constant number of pixels from the input data. To sample the pixels from the input image x, I will use a pixel sampling operation, which will be described in the following section.

Pixel Sampling operation

I will explore two-pixel sampling types for this method, which can be seen in Figure 3.6 and are described as follows:

1. **Exact Same Sampling:** In this sampling type, the conditioning information m is sampled from the input image x by sampling the same sparse pixels from the input image x. In this work, the pixels that will be sampled from the input image will be a sparse grid of pixels.

2. **Uniform Random Sampling:** In this sampling type, the conditioning information m is sampled from the input image x by sampling the exact number of pixels from the input image x uniformly at random.

The sampling process will be done every time the input image x is passed through the model. This means that in the case of the Uniform Random Sampling, the conditioning information m will be different for each time the image is passed through the model. This is done to ensure that the model learns to generalize and handle various cases.

3.2.2   Application to Gaussian VAEs

In Gaussian VAEs, the integration of a second conditioned decoder $p_\xi(x|z,m)$ follows a similar approach as outlined in the general method description. However, there are specific adjustments and considerations that need to be taken into account when applying this method to Gaussian VAEs.

The integration of the second conditioned decoder $p_\xi(x|z,m)$ into the Gaussian VAEs framework involves flattening on concatenating the latent variable z and the conditioning information m and passing it through a fully connected layer before passing it through the transposed convolutional layers of the second decoder. Although a new latent variable could have been sampled for the second decoder, the same latent variable z is used instead of sampling a new one. This is done to ensure that the latent variable z is the same for both decoders and ensures that it is easier to compare the reconstructions of the input data. The architecture of the Gaussian VAE framework with the second conditioned decoder is illustrated in Figure 3.2

Although the training strategy could have been to train decoders by alternating between them, in this thesis, I will explore the training strategy where both decoders are trained simultaneously.

In Gaussian VAEs, the loss objective consists of two components: the reconstruction loss and the KL divergence term. However, with the inclusion of the second conditioned decoder, the loss objective is extended to include the reconstruction loss of the second conditioned decoder. The overall loss objective to be **minimized** becomes

$$L = W_1 \frac{1}{D} \sum_{i=1}^{D} ||x_i - \hat{x}_{1_i}||^2 + W_2 \frac{1}{D} \sum_{i=1}^{D} ||x_i - \hat{x}_{2_i}||^2 + \beta \frac{1}{2} \sum_{i=1}^{Z} \left(-\log \sigma_\phi^2(x)_i - 1 + \mu_\phi^2(x)_i + \sigma_\phi^2(x)_i \right),$$

where D is the number of pixels in the input image, x_i is the i-th pixel of the input image, $\hat{x}_1$ is the output of the first decoder, $\hat{x}_2$ is the output of the second decoder, β is the KL divergence weight, Z is the dimension of the latent space, $\mu_\phi(x)$ and $\sigma_\phi(x)$ are the mean and the standard deviation of the Gaussian distribution produced by the encoder, respectively. The resulting loss has also extra weight parameters W_1 and W_2 to balance and control the importance of reconstruction loss, which will be hyperparameters of the model.

During the training process, the encoder is trained to produce an accurate mean and standard deviation of the Gaussian distribution, and the decoders are trained to produce accurate reconstructions of the input data. The KL divergence term is used to regularize the latent space to be close to a standard Gaussian distribution. The optimizer used to minimize the overall loss objective is the AdamW optimizer [11].

As a result of the integration of the second conditioned decoder, the model has the ability to reconstruct the input data based just on the latent variable z and as well as the conditioning information m.
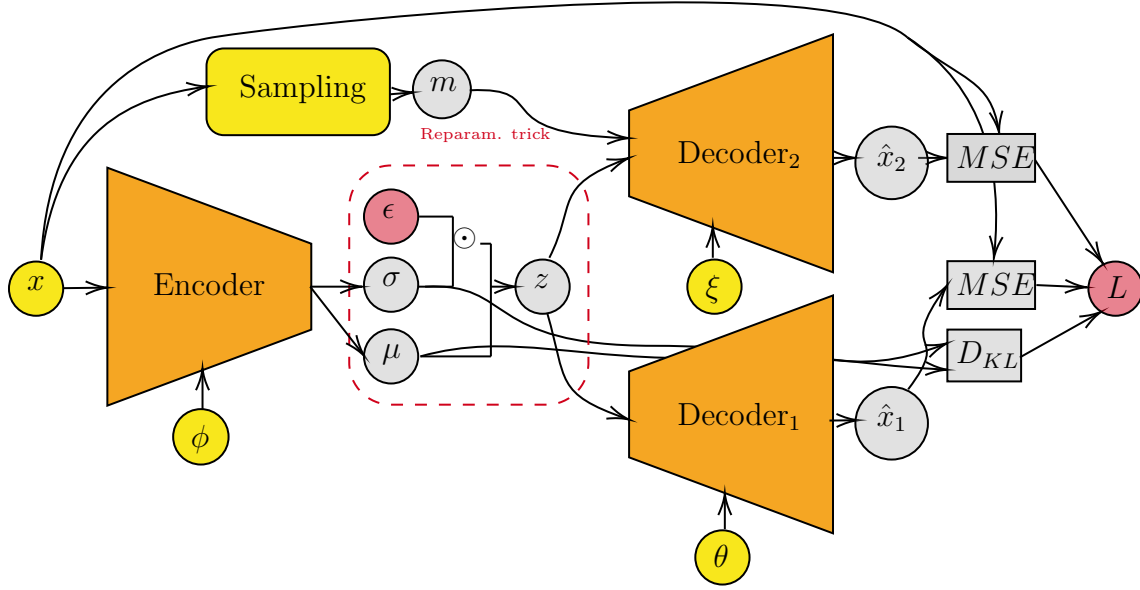
Figure 3.2: Multidecoder methodapplied to Gaussian VAEs. The Gaussian VAEs framework is extended by adding a second conditioned decoder $p_\xi(x|z,m)$. The input x is passed through the encoder with parameters ϕ producing the mean μ and the standard deviation σ of the Gaussian distribution. The random variable ϵ is sampled from a standard Gaussian distribution and is used to sample $z = \mu + \sigma \odot \epsilon$. The sampled z is then used as input to both decoders. As a result, the loss function consists of three components: the MSE reconstruction loss of the first decoder, the MSE reconstruction loss of the second decoder and the KL divergence regularization loss.

3.2.3 Application to VQ-VAEs

The approach of integrating the second conditioned decoder $p_\xi(x|z,m)$ into the VQ-VAEs framework is similar to the approach used for Gaussian VAEs. However, some differences need to be taken into account when applying this method to VQ-VAEs.

One of the main differences is that the VQ-VAEs framework uses a discrete latent space and the Vector Quantization which is described in section 2.2. This means that the latent variable z is a discrete variable that represents the indexes of the embedding table vectors. This means that first the corresponding embedding table vectors $z_q(x)$ must be computed for the latent variable z before flattening and concatenating it with the conditioning information m and then using a fully connected layer before the transposed convolutional layers of the second decoder. The architecture of the VQ-VAEs framework with the second conditioned decoder is illustrated in Figure 3.3.

The training strategy is the same as in Gaussian VAEs, where both decoders are trained simultaneously. This means that the loss objective must be extended to include the reconstruction loss of the second conditioned decoder. In VQ-VAEs, the loss objective consists of three components: the MSE reconstruction loss the commitment loss and the codebook loss. However, with the inclusion of the second conditioned decoder, the loss objective is extended to include the reconstruction loss of the second conditioned decoder. The overall loss objective to be **minimized** becomes

$$L = W_1 \frac{1}{D} \sum_{i=1}^{D} ||x_i - \hat{x}_{1_i}||^2 + W_2 \frac{1}{D} \sum_{i=1}^{D} ||x_i - \hat{x}_{2_i}||^2 + \frac{1}{Z} \sum_{i=1}^{Z} \left(||sg(z_e(x)_i) - e_{k_i}||^2 + \beta ||z_q(x)_i - sg(e_{k_i})||^2 \right),$$

where D is the number of pixels in the input image, x_i is the i-th pixel of the input image, $\hat{x}_1$ is the output of the first decoder, $\hat{x}_2$ is the output of the second decoder, Z is the number of latent space vectors, $z_e(x)$ is the output of the encoder, $z_q(x)$ is the mapping output after Vector Quantization, e_k is the k-th embedding table vector, β is the commitment loss weight, sg is the stop gradient operation, which was previously defined in the background section 2.2.

After the first stage of the training, the model has the ability to reconstruct the input data based just on the latent variable z and as well as the conditioning information m. To generate the latent variable z, one must be trained a separate autoregressive model, which will be used to generate the latent variable z.
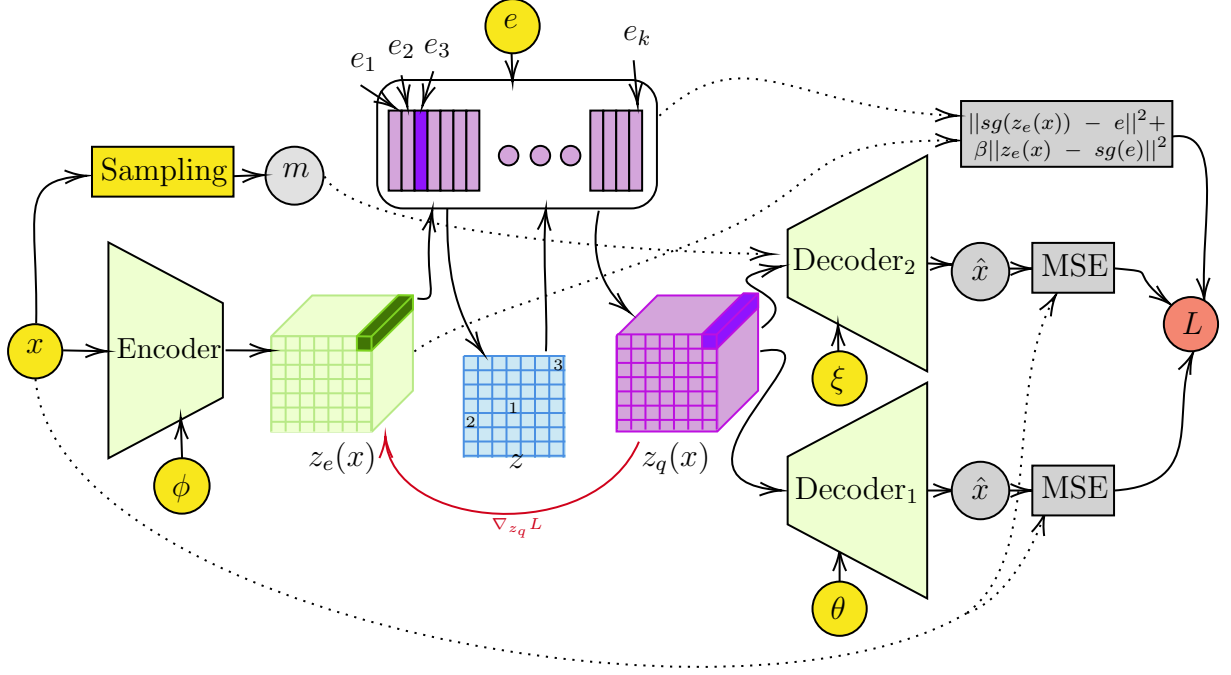
Figure 3.3: Multidecoder method applied to VQ-VAEs. The VQ-VAEs framework is extended by adding a second conditioned decoder. Instead of using a fully connected layer to merge the latent variable z and the conditioning information m, the corresponding embedding table vectors $z_q(x)$ must be computed for the latent variable z before merging it with the conditioning information m. As a result of adding the second conditioned decoder, the loss function requires the addition of the MSE reconstruction loss of the second decoder.

3.3 Single decoder method

In the Single decoder method, the idea is to employ a single decoder within the VAE architecture that is capable of reconstructing the input data under variable conditioning. Variable conditioning means that the conditioning information m can be a variable amount of information or just an empty mask. This method differs from the Multidecoder method in that it uses a single decoder for both tasks and dynamically adjusts its behavior based on the amount of information present in the variable m.

This method utilizes a single decoder that dynamically adjusts its behavior based on the amount of information present in the variable m. When the conditioning information

is available, the decoder incorporates it into the reconstruction process and takes advantage of it to get the best reconstruction. Otherwise, it operates solely based on the latent variable z.

To achieve this, in this method, I will use different conditioning strategies, which will be described in the following subsection.

3.3.1 Conditioning strategy

Similar to the previous method, the conditioning information m is acquired by sampling pixels from the input image x. However, in this method, the process of sampling the conditioning information m is different in that the conditioning information m is sampled by selecting a variable number of pixels from the input image x. This is done to ensure that the decoder of the model learns to handle various cases where there is variable conditioning information.

To achieve this, I will implement a two-step sampling process:

1. **Sampling the number of pixels:** In this step, the number of pixels to be sampled from the input image x is sampled from a power law distribution. The power law distribution is scaled to the size of the input image x and is used to sample the number of pixels to be sampled from the input image x.

2. **Sampling the pixels:** In this step, the conditioning information m is sampled from the input image x by sampling the number of pixels sampled in the previous step from the input image: uniformly at random or from a Gaussian distribution. Both of these sampling types can be seen in Figure 3.6.

The sampling two-step process will be done every time the input image x is passed through the model. This means that the conditioning information m will be different for each time the image is passed through the model. This is done to ensure that the model learns to generalize and handle various cases.

During training, the number of pixels to be sampled from the input image x varies every time the input image x is passed through the model and is sampled from a power law distribution. The power law distribution is chosen because it can have a finite range and scalability, which makes it suitable for sampling the number of pixels from the input image. The exponent of the power law distribution will be chosen to be high so that the decoder can learn to handle also the cases where the conditioning information is not available. The exponent will be a hyperparameter of the model.

3.3.2 Application to Gaussian VAEs

In Gaussian VAEs, applying the second method involves modifying the decoder to be capable of reconstructing the input data under variable conditioning. The modified decoder $p_\xi(x|z, m)$ is capable of reconstructing the input data based on the latent variable z and the conditioning information m or just the latent variable z.

This is achieved by using a fully connected layer to merge the latent variable z and the conditioning information m before passing it through the transposed convolutional layers. The architecture of the Gaussian VAEs framework with a single decoder capable of variable conditioning is illustrated in Figure 3.4.
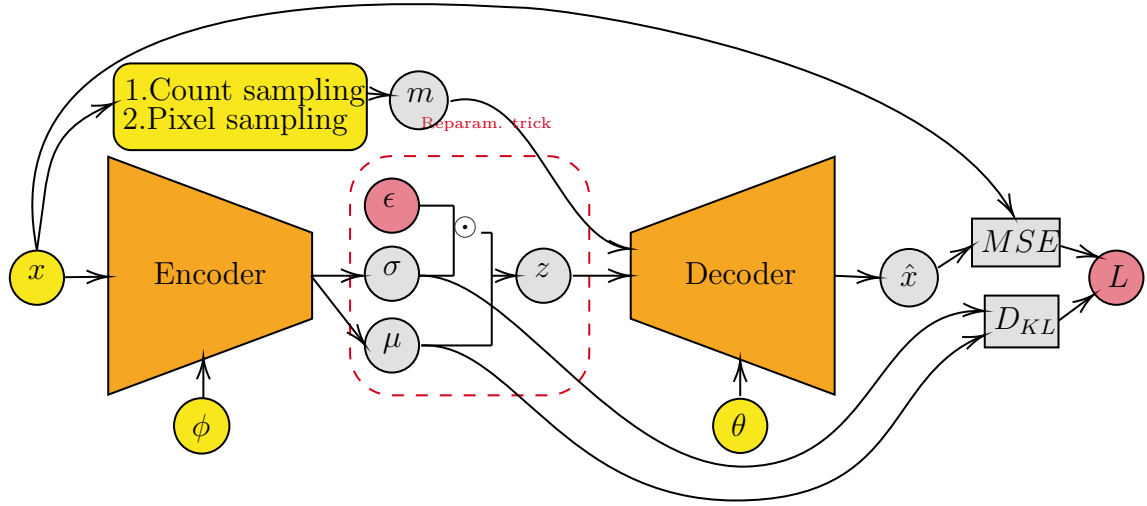

Figure 3.4: Single decoder methodapplied to Gaussian VAEs. The Gaussian VAEs framework is modified by allowing the decoder to be conditioned on a variable amount of information and to dynamically adjust its behavior based on the amount of information present in the variable m. The input x is passed through 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 used as input to the decoder as well as the extra conditioning information m. With this approach, there is no need for a second decoder, as the single decoder is capable of reconstructing the data based on the latent variable z and the conditioning information m or just the latent variable z.

3.3.3 Application to VQ-VAEs

In VQVAEs, the same as in Gaussian VAEs, the second method involves modifying the decoder to be capable of receiving variable conditioning information. Unlike Gaussian

VAEs, the VQ-VAEs use Vector Quantization to discretize the continuous latent space, which is described in section 2.2. Since the latent variable z is just a representation of the indexes of the embedding table vectors, the latent variable z must be first mapped to the corresponding embedding table vectors resulting in $z_q(x)$ representation. Then the same process as in Gaussian VAEs is applied where the latent variable $z_q(x)$ and the conditioning information m are merged using a fully connected layer before passing it through the transposed convolutional layers of the decoder. The architecture of the VQ-VAEs framework with a single decoder capable of variable conditioning is illustrated in Figure 3.5.
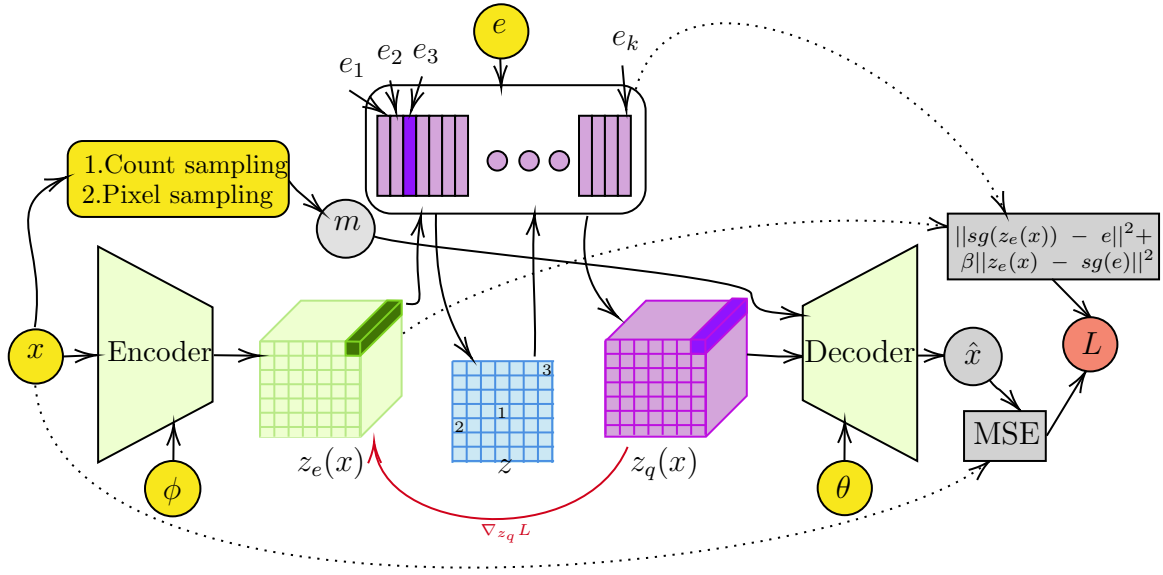

Figure 3.5: Single decoder methodapplied to VQ-VAEs. The VQ-VAEs framework is modified by allowing the decoder to be conditioned on a variable amount of information and to dynamically adjust its behavior based on the amount of information present in the variable m. The input x is passed through the encoder producing the output $z_e(x)$. The output $z_e(x)$ is then used as input to the autoregressive model to generate the latent variable z. The latent variable z is then used as input to the decoder as well as the extra conditioning information m. With this approach, there is no need for a second decoder, as the single decoder is capable of reconstructing the data based on the latent variable z and the conditioning information m or just the latent variable z.
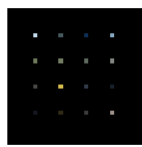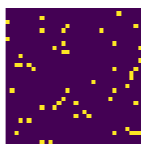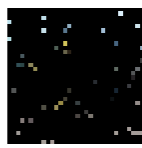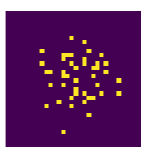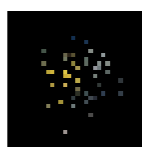
Figure 3.6: Table of pixel sampling types for conditioning. The table has 3 rows each representing a different sampling type. The first row represents the Exact Same Sampling type, the second row represents the uniform random sampling type and the third row represents the Gaussian sampling type where the pixels are more likely to be sampled from the center of the image. The first column represents the original image, the second column represents the mask and the third column represents the result of the sampling operation.

3.4 Experimental setup

In this section, I will describe the experimental setup that will be used to evaluate the proposed methods. The full code for experiments was implemented in Python using Pytorch and is available on GitHub[1]

3.4.1 Datasets

The proposed methods will be evaluated on the MNIST, CIFAR-10 and CelebA datasets. The MNIST dataset consists of 60,000 images of the size 28x28 pixels with 10 classes. The CIFAR-10 dataset consists of 60,000 colored images of the size 32x32 pixels with 10 classes. The CelebA dataset consists of 202,599 images of celebrities with 10,177 identities and 40 binary attributes. In these experiments, the CelebA dataset images will be resized to 64x64 pixels to reduce the computational complexity. The datasets will be split into training, validation and test sets, where the training set will be used to train the models and the validation set will be used to evaluate.

3.4.2 Network architecture

The encoder and the decoder of the models will be convolutional neural networks. The number of layers and the number of filters in each layer will be hyperparameters of the model. Both the encoder and the decoder will have a similar architecture of convolutional layers followed by batch normalization layers and LeakyReLU activation functions. However, the encoder and the decoder for the Gaussian VAEs will be more shallow compared to the VQ-VAEs, because the Gaussian VAEs suffer from the posterior collapse if the neural network is too deep.

For Gaussian VAEs experiments were conducted in 2 configurations:

- Configuration 1: latent space dimension of 16, with hidden dimensions of 32, 64, 128, 256
- Configuration 2: latent space dimension of 64, with hidden dimensions of 32, 64, 128, 256

---

[1]https://github.com/EdvardsZ/MastersThesis

For VQ-VAEs experiments were conducted in 3 configurations:

- Configuration 1: 128 embeddings and 16 the embedding dimension, hidden dimensions for both encoder and decoder were set to 32 and 64 and 0 residual layers.
- Configuration 2: 128 embeddings and 32 the embedding dimension, hidden dimensions for both encoder and decoder were set to 128 and 256 and 2 residual layers.
- Configuration 3: 256 embeddings and 64 the embedding dimension, hidden dimensions for both encoder and decoder were set to 128 and 256 and 4 residual layers.

3.4.3   Training

The models will be trained using the AdamW optimizer with a learning rate of 10^{-3} and a batch size of 128 for 100 epochs. The models will be trained on the training set and evaluated on the validation set. The best model will be selected based on the validation loss. The models will be trained on a single GPU.

Chapter 4

Results

This chapter presents the findings of the experiments conducted in this master's thesis. The chapter is divided into two sections. The first section presents the results of Multidecoder method, and the second section presents the results of Single decoder method, where both methods will be evaluated in the context of both Gaussian VAEs and VQ-VAEs. In the final section, I will present the cross-validation results of both methods.

4.1 Results of Multidecoder method

In this section, I will present the results of Multidecoder method on both Gaussian VAEs and VQ-VAEs.

4.1.1 Results on Gaussian VAEs

The experiments showed similar results for both exact sampling and uniform sampling. The results showed that Multidecoder method can be used not only to obtain a conditioned decoder but also to improve the quality of non-conditioned decoder reconstruction when compared to non-conditioned Gaussian VAEs. However, KL divergence loss of the latent space increased when Multidecoder method was applied. This can be explained by the fact that there is a trade-off between the quality of the reconstruction and the KL divergence of the latent space. An example of this can be seen in figure 4.2.

The reconstruction loss of the conditioned decoder was improved when compared to the non-conditioned decoder 4.1.

To validate and verify that the results are not influenced due to the architecture of the neural network, I applied a range of loss-balancing techniques such as coefficient balancing and the SoftAdapt algorithm. After applying these techniques it could still be observed that the quality of the reconstruction was improved at the expense of a slightly higher KL divergence loss of the latent space. When comparing the results of the Exact Same Sampling and Uniform Sampling it showed little difference to each other.

When trying to use deeper neural networks on the Gaussian VAEs, I observed that the posterior collapse was more likely to happen, which is a common problem in Gaussian VAEs [20].
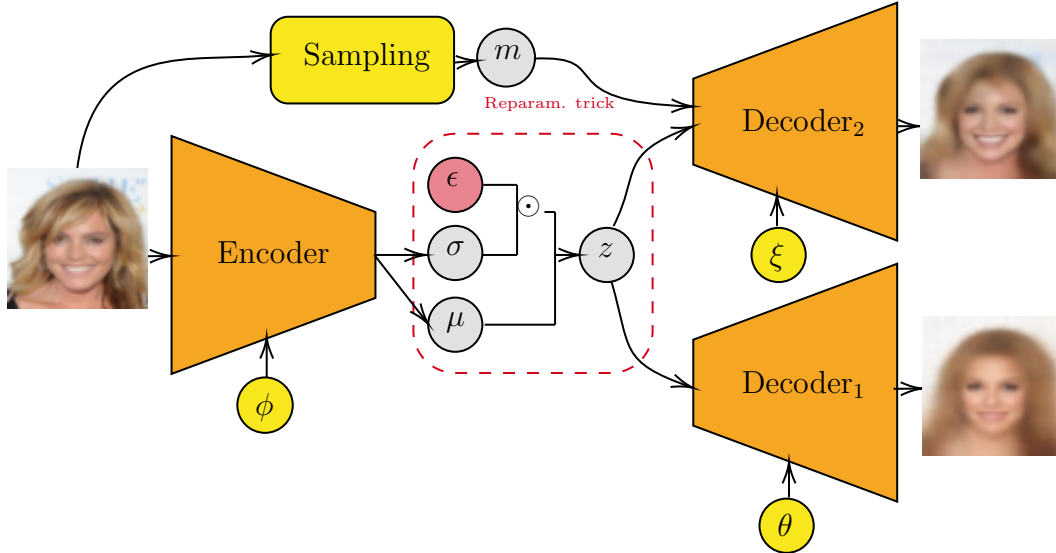
Figure 4.1: Trained neural network with Multidecoder method with Exact Same Sampling applied to a Gaussian VAE on CelebA dataset and latent space 16. On the left side as input is the original image, on the right side there are two outputs of the decoders. The image from the conditioned decoder is reconstructed with higher quality compared to the non-conditioned decoder because the conditioned decoder $Decoder_2$ uses conditioning information m to improve the quality of the reconstruction.
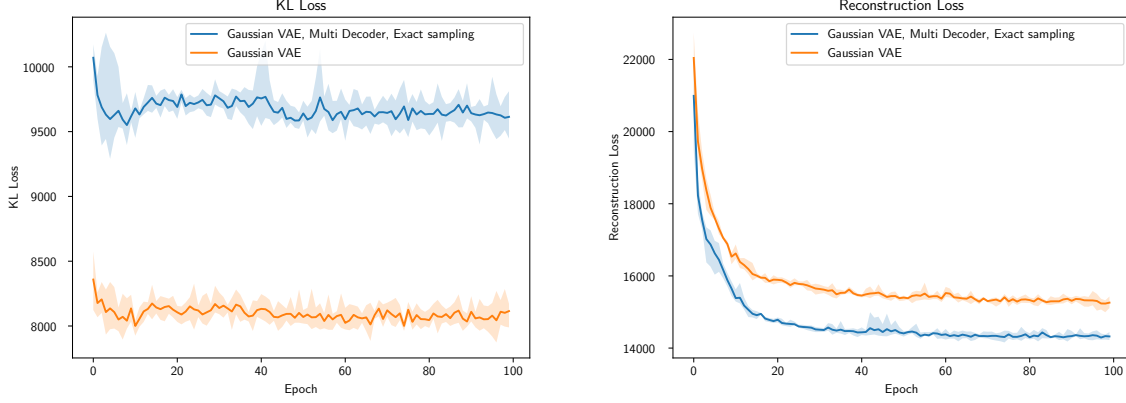
Figure 4.2: Validation losses during training with and without Multidecoder method applied on Gaussian VAE. The results are shown for the non-conditioned decoder - $Decoder_1$, and Exact Sampling. The dataset used is CelebA. Left: KL divergence loss of the latent space comparison. Right: Reconstruction loss comparison of the $Decoder_1$ - non-conditioned decoder.

4.1.2 Results on VQ-VAEs

When applying Multidecoder method to VQ-VAEs, the results showed that both the quality of the reconstruction and the VQ objective loss improved, which could be observed for both Exact Same Sampling and Uniform Sampling, as shown in figure 4.4. This could be explained by the fact that the VQ-VAEs are more robust and stable compared to Gaussian VAEs.

The conditioned decoder was able to reconstruct the image with a higher quality compared to the non-conditioned decoder, however, the difference observed was not as significant as in the Gaussian VAEs. The reasoning behind this lies in the fact that the VQ-VAEs already have a high-quality reconstruction, which makes it harder to improve the quality of the reconstruction.

Same as previously mentioned the method was applied with the SoftAdapt loss balancing technique and without it. The results showed little to no difference between the two. However, the training stability was improved when using SoftAdapt with fewer fluctuations in the loss. When looking at the differences between the Exact Same Sampling and Uniform sampling it showed very little difference between the two, however, the Uniform sampling showed slightly better results.
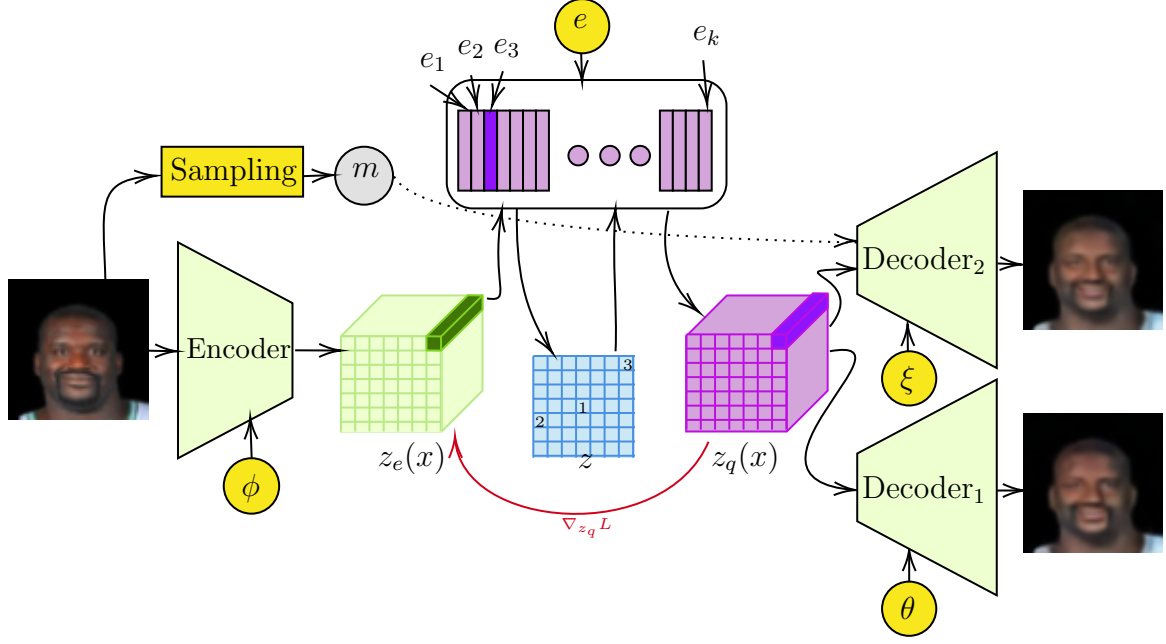
Figure 4.3: Trained neural network with Multidecoder method and with the Exact Same Sampling applied to a VQ-VAE on the CelebA dataset. The neural network has a latent space of 32 and a table of 128 embeddings, with both the encoder having 4 residual layers and the decoder having 4 residual layers. On the left side as input is the original image, on the right side there are two outputs of the decoders. The image from the conditioned decoder is reconstructed with higher quality compared to the non-conditioned decoder because the conditioned decoder $Decoder_2$ uses conditioning information m.
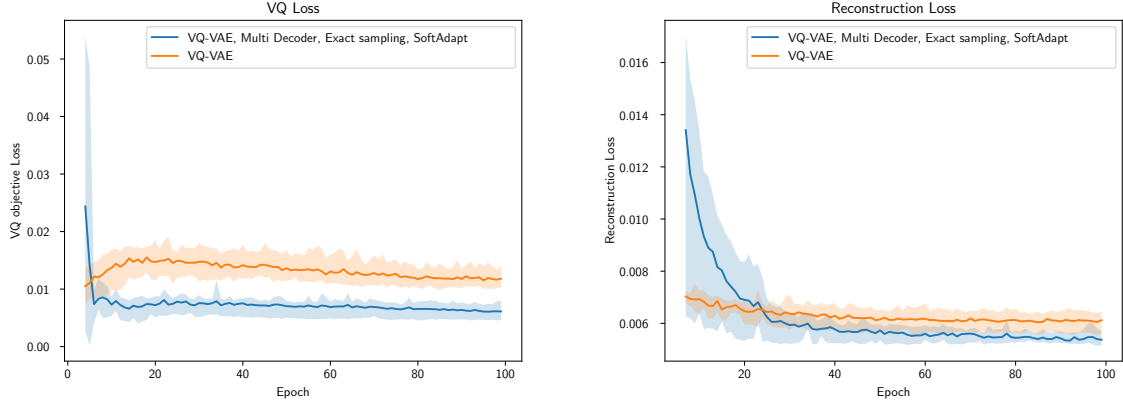
Figure 4.4: Validation losses during training with and without Multidecoder method applied on VQ-VAE Configuration 3. The results of Multidecoder method in the plot shown for the non-conditioned decoder - $Decoder_1$, and Exact Sampling is used with SoftAdapt. The dataset used is CelebA. Left: VQ objective loss comparison. Right: Reconstruction loss comparison.

4.2   Results of Single decoder method

In this section, I will present the results of Single decoder method on both Gaussian VAEs and VQ-VAEs. The method was applied to the same datasets as the previous method. For this method, I ran the experiments with both Uniform and Gaussian sampling. The count sampling was done with Power law distribution with different exponent values.

4.2.1   Results on Gaussian VAEs

The results showed that Single decoder method can be used for unifying two decoders into one. The experiments were conducted with both Uniform and Gaussian sampling. The results showed that for both methods this comes at a cost of slightly lower quality of the reconstruction when no conditioning information is given compared to a standard Gaussian VAE. However, the findings showed that this approach can substantially reduce the KL divergence loss of the latent space, which can be seen in figure 4.5.

Although, for both Uniform and Gaussian sampling findings showed that this method results in a slightly lower quality of the reconstruction compared to a standard Gaussian VAE. This approach reduced the KL divergence loss of the latent space for both of the sampling methods. For Uniform sampling, the results showed that the KL divergence loss of the latent space was reduced more compared to Gaussian sampling. One possible explanation for this could be that with Gaussian sampling, there is a higher probability of sampling the same pixel multiple times, which can be less informative for the decoder.

Experiments were also conducted with a range of different exponent values for the power-law distribution. The findings showed that the higher the exponent value the more the model was able to reduce the reconstruction loss of the scenario, where no conditioning information is given. However, the KL divergence loss of the latent space increased with higher exponent values.

Figure 4.5: Validation losses during training with and without Single decoder method applied on Gaussian VAE Configuration 2. Left: KL loss comparison. Right: Reconstruction loss comparison of the $Decoder_1$ - non-conditioned decoder with a standard Gaussian VAE.

4.2.2 Results on VQ-VAEs

The results showed that Single decoder method worked very well with VQ-VAEs. The results showed it can be used to substantially reduce the VQ objective loss and at the same time to improve the quality of the reconstruction when no conditioning information is given. This showed to be the case for both Uniform and Gaussian sampling with a high enough exponent value for the power-law distribution.

The experiments were run with a range of different exponent values: 20, 30, 40, 50, 60. The results showed that the higher the exponent value the better the reconstruction in the scenario where no conditioning information is given, which is expected since the higher the exponent value the less number of pixels are given to the decoder as conditioning information on average.

When comparing Uniform and Gaussian sampling, the results showed that Gaussian sampling performed slightly better than Uniform sampling. This can be explained by the fact that the information around the center of the image is more important for the reconstruction, which is more likely to be sampled with Gaussian sampling.
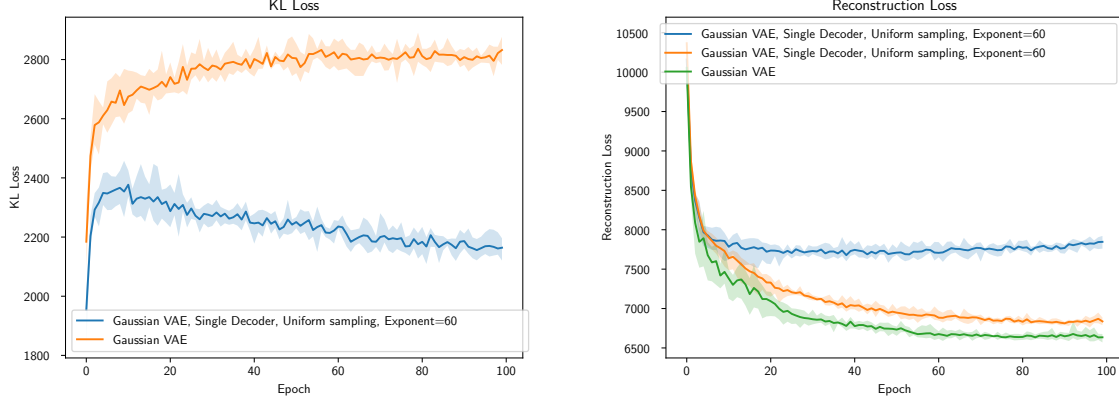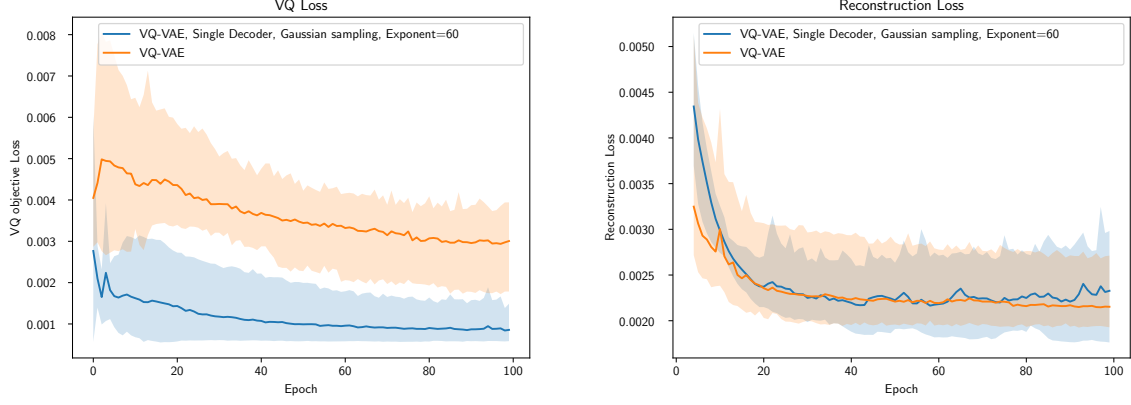
Figure 4.6: Validation losses during training with and without Single decoder method applied on VQ-VAE Configuration 3. Left: VQ objective loss comparison. Right: Reconstruction loss comparison of the $Decoder_1$ - non-conditioned decoder with a standard VQ-VAE.

4.3 Cross-validation results

In this section, I will present the cross-validation results of both methods. The cross-validation was conducted on both Gaussian VAEs and VQ-VAEs. The cross-validation was conducted with 5 folds, where the data was split into 80% training and 20% validation. The final cross-validation results are presented in tables 4.1, 4.2, and 4.3.

WIP: there should be a table but results are not just in yet

Table 4.1: Cross-validation results of Multidecoder method and Single decoder method on the CelebA dataset.

	Config.	Method	Parameters	Reconstruction loss	VQ/KL loss
		-	-	0.0061 +- 5.7e-07	0.0108 +- 2.2e-06
		Multi Decoder	Exact sampling	0.0076 +- 4.1e-07	0.0071 +- 8.7e-07
			Exact sampling, SoftAdapt	0.0300 +- 5.2e-04	0.0038 +- 2.1e-05
			Uniform sampling	0.0068 +- 8.1e-07	0.0103 +- 8.5e-06
			Uniform sampling, SoftAdapt	0.0278 +- 4.3e-04	**0.0016 +- 4.6e-06**
	2		Gaussian sampling, Exponent=40	0.0071 +- 4.5e-07	0.0053 +- 3.6e-06
			Gaussian sampling, Exponent=40	0.0077 +- 2.5e-06	0.0032 +- 9.7e-07
			Gaussian sampling, Exponent=60	0.0065 +- 5.9e-07	0.0037 +- 5.9e-07
		Single Decoder	Gaussian sampling, Exponent=60	**0.0060 +- 1.3e-07**	0.0024 +- 1.4e-06
			Uniform sampling, Exponent=40	0.0070 +- 1.1e-06	0.0047 +- 3.4e-06
			Uniform sampling, Exponent=40	0.0078 +- 2.9e-06	0.0039 +- 2.3e-06
			Uniform sampling, Exponent=60	0.0062 +- 1.5e-07	0.0032 +- 2.1e-06
			Uniform sampling, Exponent=60	0.0075 +- 4.2e-06	0.0044 +- 6.6e-07
		-	-	0.0089 +- 2.2e-06	0.0074 +- 6.6e-06
		Multi Decoder	Exact sampling	0.0064 +- 6.5e-07	0.0088 +- 3.1e-06
			Exact sampling, SoftAdapt	0.0292 +- 7.4e-04	0.0031 +- 9.7e-06
			Uniform sampling	**0.0063 +- 6.1e-08**	0.0099 +- 4.0e-07
VQ-VAE	3		Uniform sampling, SoftAdapt	0.0277 +- 4.5e-04	0.0026 +- 6.8e-06
		Single Decoder	Gaussian sampling, Exponent=40	0.0065 +- 2.7e-07	0.0028 +- 2.5e-06
			Gaussian sampling, Exponent=60	0.0064 +- 4.9e-07	0.0026 +- 2.5e-06
			Uniform sampling, Exponent=40	0.0063 +- 1.7e-07	**0.0026 +- 2.0e-06**
			Uniform sampling, Exponent=60	0.0072 +- 3.2e-07	0.0043 +- 2.3e-06
		-	-	0.0076 +- 1.1e-07	**0.0029 +- 4.2e-07**
		Multi Decoder	Exact sampling	0.0071 +- 1.2e-07	0.0034 +- 5.4e-07
			Exact sampling, SoftAdapt	0.0067 +- 2.0e-07	0.0090 +- 1.7e-06
			Uniform sampling	0.0076 +- 6.6e-08	0.0082 +- 7.1e-07
	1		Uniform sampling, SoftAdapt	0.0129 +- 1.2e-04	0.0070 +- 7.0e-06
		Single Decoder	Gaussian sampling, Exponent=40	0.0071 +- 2.1e-06	0.0033 +- 1.9e-06
			Gaussian sampling, Exponent=60	0.0075 +- 3.8e-06	0.0037 +- 7.4e-07
			Uniform sampling, Exponent=40	**0.0063 +- 1.6e-07**	0.0041 +- 2.1e-06
			Uniform sampling, Exponent=60	0.0075 +- 3.2e-06	0.0036 +- 3.6e-07
		-	-	0.0170 +- 4.7e-03	0.0071 +- 2.5e-03
		Multi Decoder	Exact sampling	0.0142 +- 3.3e-03	0.0108 +- 3.5e-03
			Exact sampling, SoftAdapt	0.0137 +- 7.5e-04	0.0115 +- 1.1e-02
			Uniform sampling	0.0141 +- 2.6e-03	0.0106 +- 6.7e-03
	2		Uniform sampling, SoftAdapt	**0.0136 +- 2.9e-02**	0.0114 +- 8.1e-02
		Single Decoder	Gaussian sampling, Exponent=40	0.0172 +- 6.1e-03	0.0056 +- 3.1e-03
			Gaussian sampling, Exponent=60	0.0172 +- 3.3e-03	0.0057 +- 1.4e-03
			Uniform sampling, Exponent=40	0.0173 +- 4.0e-03	0.0055 +- 6.2e-04
Gaussian VAE			Uniform sampling, Exponent=60	0.0173 +- 2.6e-03	**0.0055 +- 1.3e-03**
		-	-	0.0190 +- 1.5e-03	0.0060 +- 1.3e-03
		Multi Decoder	Exact sampling	0.0182 +- 4.9e-03	0.0075 +- 1.0e-03
			Exact sampling, SoftAdapt	**0.0177 +- 6.1e-03**	0.0091 +- 3.2e-03
			Uniform sampling	0.0183 +- 1.3e-03	0.0072 +- 2.6e-03
	1		Uniform sampling, SoftAdapt	0.0179 +- 1.8e-03	0.0089 +- 1.5e-03
		Single Decoder	Gaussian sampling, Exponent=40	0.0181 +- 5.8e-03	0.0050 +- 1.6e-03
			Gaussian sampling, Exponent=60	0.0180 +- 8.2e-03	0.0050 +- 2.4e-03
			Uniform sampling, Exponent=40	0.0183 +- 4.9e-03	**0.0048 +- 1.7e-03**
			Uniform sampling, Exponent=60	0.0182 +- 4.3e-03	0.0048 +- 6.2e-04

Table 4.2: Cross-validation results of Multidecoder method and Single decoder method on the CIFAR10 dataset.

	Config.	Method	Parameters	Reconstruction loss	VQ/KL loss
VQ-VAE	1	-	-	0.0028 +- 8.8e-09	0.0101 +- 4.7e-07
		Multi Decoder	Exact sampling	0.0025 +- 6.5e-09	0.0082 +- 1.9e-07
			Exact sampling, SoftAdapt	0.0123 +- 3.9e-04	0.0035 +- 2.3e-06
			Uniform sampling	**0.0023 +- 2.4e-09**	0.0076 +- 2.8e-07
			Uniform sampling, SoftAdapt	0.0024 +- 3.6e-08	0.0042 +- 2.4e-07
		Single Decoder	Gaussian sampling, Exponent=40	0.0028 +- 7.4e-08	0.0028 +- 2.0e-07
			Gaussian sampling, Exponent=60	0.0029 +- 7.2e-08	0.0029 +- 1.3e-07
			Uniform sampling, Exponent=40	0.0029 +- 6.4e-08	0.0030 +- 4.2e-07
			Uniform sampling, Exponent=60	0.0030 +- 7.2e-08	**0.0024 +- 2.4e-08**
	3	-	-	0.0019 +- 1.3e-08	0.0024 +- 4.3e-08
		Multi Decoder	Exact sampling	0.0017 +- 8.4e-09	0.0028 +- 1.2e-07
			Exact sampling, SoftAdapt	0.0173 +- 6.9e-04	0.0011 +- 2.2e-07
			Uniform sampling	**0.0017 +- 2.4e-08**	0.0024 +- 1.2e-07
			Uniform sampling, SoftAdapt	0.0178 +- 6.5e-04	0.0009 +- 2.8e-07
		Single Decoder	Gaussian sampling, Exponent=40	0.0038 +- 8.7e-08	0.0008 +- 2.4e-07
			Gaussian sampling, Exponent=60	0.0037 +- 5.7e-08	**0.0005 +- 1.1e-07**
			Uniform sampling, Exponent=40	0.0039 +- 1.1e-07	0.0007 +- 1.8e-08
			Uniform sampling, Exponent=60	0.0038 +- 2.4e-08	0.0007 +- 4.0e-08
	2	-	-	0.0017 +- 1.6e-09	0.0022 +- 2.0e-08
		Multi Decoder	Exact sampling	**0.0014 +- 2.8e-09**	0.0028 +- 3.3e-08
			Exact sampling, SoftAdapt	0.0155 +- 7.9e-04	0.0013 +- 3.6e-07
			Uniform sampling	0.0015 +- 2.7e-09	0.0030 +- 2.5e-08
			Uniform sampling, SoftAdapt	0.0422 +- 1.1e-03	**0.0009 +- 3.2e-07**
		Single Decoder	Gaussian sampling, Exponent=40	0.0031 +- 1.6e-07	0.0014 +- 3.5e-07
			Gaussian sampling, Exponent=60	0.0031 +- 2.1e-07	0.0013 +- 2.6e-07
			Uniform sampling, Exponent=40	0.0029 +- 1.4e-07	0.0017 +- 5.6e-08
			Uniform sampling, Exponent=60	0.0035 +- 3.3e-07	0.0012 +- 4.3e-07
Gaussian VAE	1	-	-	0.0178 +- 1.2e-03	0.0165 +- 4.5e-03
		Multi Decoder	Exact sampling	0.0145 +- 2.3e-03	0.0222 +- 1.2e-02
			Exact sampling, SoftAdapt	0.0168 +- 1.3e-03	0.0186 +- 1.3e-02
			Uniform sampling	**0.0134 +- 5.4e-03**	0.0234 +- 2.6e-02
			Uniform sampling, SoftAdapt	0.0158 +- 3.3e-03	0.0195 +- 3.9e-03
		Single Decoder	Gaussian sampling, Exponent=40	0.0204 +- 2.7e-02	0.0126 +- 6.8e-03
			Gaussian sampling, Exponent=60	0.0203 +- 1.7e-02	**0.0124 +- 2.9e-03**
			Uniform sampling, Exponent=40	0.0201 +- 9.6e-03	0.0148 +- 2.4e-03
			Uniform sampling, Exponent=60	0.0202 +- 2.7e-02	0.0147 +- 3.0e-03
	2	-	-	0.0181 +- 8.7e-03	0.0161 +- 9.4e-03
		Multi Decoder	Exact sampling	0.0148 +- 8.5e-03	0.0219 +- 7.5e-03
			Exact sampling, SoftAdapt	0.0167 +- 1.1e-02	0.0188 +- 3.7e-03
			Uniform sampling	**0.0136 +- 8.8e-03**	0.0235 +- 2.3e-02
			Uniform sampling, SoftAdapt	0.0158 +- 2.3e-03	0.0195 +- 4.8e-03
		Single Decoder	Gaussian sampling, Exponent=40	0.0209 +- 5.4e-02	0.0125 +- 1.0e-03
			Gaussian sampling, Exponent=60	0.0207 +- 2.7e-02	**0.0125 +- 4.0e-03**
			Uniform sampling, Exponent=40	0.0204 +- 3.4e-02	0.0147 +- 4.4e-03
			Uniform sampling, Exponent=60	0.0204 +- 2.8e-02	0.0149 +- 5.3e-03

Table 4.3: Cross-validation results of Multidecoder method and Single decoder method on the MNIST dataset.

Chapter 5

Discussion

In this chapter, the results in the preceding chapter are analyzed, interpreted and discussed in the context of the objective and research questions. This chapter is divided into four sections. The first section discusses the results of Multidecoder method, and the second section discusses the results of Single decoder method. The third section provides a comparative analysis of the two methods, and the final section discusses the limitations of the research and potential future work.

5.1   Analysis of Multidecoder method

The results of Multidecoder method show that it is overall a viable method for combining semi-conditional and non-conditional VAEs. The resulting model has multitask capabilities, which allow it to improve the quality of the reconstruction and the generalization capabilities of the VAE. It was shown that the method can be applied to both VQ-VAE and Gaussian VAEs.

One of the disadvantages of this method is that it requires a second decoder, which increases the complexity of the model and the computational cost. However, the results show that in some cases, the increase in complexity is justified by the improvement in the quality of the reconstruction.

5.1.1 Findings on Gaussian VAEs

The results on Gaussian VAEs showed that a standard Gaussian VAE can be combined with a semi-conditional VAE by using Multidecoder method. This method improved the reconstruction quality of the non-conditioned decoder and added a second decoder that allows the model to reconstruct or generate images given some pixels. However, the results showed that this meant a slight increase in the KL divergence loss of the latent space. This is expected since the model has to learn two decoders instead of one, and it means it puts less emphasis on the KL divergence loss of the latent space.

It could be observed that the conditioned decoder was able to reconstruct the images with noticeably higher quality than the non-conditioned decoder. This is expected since the conditioned decoder has more information direct information about image pixels, which makes it easier to reconstruct.

When comparing Exact and Uniform sampling, I did not see any significant difference in the results. This is something that could be further investigated in future work. One possible explanation for this could be that the model is not able to fully take advantage of both sampling methods because it has already a trade-off between the KL divergence loss of the latent space and the reconstruction loss of both decoders.

5.1.2 Findings on VQ-VAEs

When applying Multidecoder method to VQ-VAEs, the results showed that both the quality of the reconstruction and the VQ objective loss can be improved. As a result, the model gets multitask properties to reconstruct and generate images given some pixels or to reconstruct images with higher quality. The improvement can be explained that even though 2 decoders are used, they still share the same encoder, which can make it better at its core task.

It was observed that the conditioned decoder was able to reconstruct the image with a higher quality compared to the non-conditioned decoder, however, the difference observed was not as significant as with Gaussian VAEs. The reasoning behind this could be that the VQ-VAEs already have a high-quality reconstruction, which makes it harder to improve the quality of the reconstruction.

The Multidecoder method was tested with both Exact Same Sampling and Uniform Sampling, and it was observed even though the results were similar, the Uniform Sampling showed slightly better results. The rationale behind this could be that Uniform Sampling gives much more diverse samples, which can be more informative for the network.

5.2 Analysis of Single decoder method

Single decoder method involves using the same decoder to unify both the conditioned and non-conditioned tasks, which is done by using variable conditioning - a technique that allows conditioning of the decoder on a variable amount of information or just an empty mask. The rationale behind this is that the model decoder can learn to do both tasks and can detect if and where the mask is empty.

One of the advantages of this method is that it does not require a second decoder, which reduces the complexity of the model and the computational cost, which can be a problem for large-scale high-resolution models.

5.2.1 Findings on Gaussian VAEs

Upon analysis, it has become clear that when Single decoder method is applied to Gaussian VAEs, the decoder of the model can reconstruct images given some pixels as conditioned information or with no information at all. This method improved substantially the KL divergence loss of the latent space, which means that it is possible to generate more accurate samples from the latent space.

However, one of the drawbacks of this method is that it reduced the quality of the reconstruction in the non-conditioned case. The cause for this could be that the decoder and the encoder are not deep enough to learn to do both tasks at the same time. In my experiments, when using a deeper encoder and decoder, it resulted in posterior collapse, which is a common problem in Gaussian VAEs.

It could also be observed that Uniform Sampling showed better results than Gaussian Sampling. The cause for this could be traced back to the fact that for Gaussian Sampling, there's a high chance to sample the same pixel multiple times, which can make the model overfit to the same pixels.

5.2.2 Findings on VQ-VAEs

When applying Single decoder method to VQ-VAEs, the results showed that the model had significantly reduced the VQ objective loss, which means that the model is more

accurate a can be used more effectively for image generation. One possible explanation for the improvement in the VQ objective loss is that the conditioning of the decoder gives the encoder more direct and accurate gradients to learn from.

The experiments showed that the reconstruction quality in the case of no conditioning could also be improved if the exponent value of the Power Law distribution was set to a higher value. This is because the higher the exponent value of the Power Law distribution, the fewer pixels on average are sampled, which means that the model more often has to reconstruct the image from scratch.

It was discovered that the Gaussian Sampling showed slightly better results than Uniform Sampling. The reason behind this could be that the Gaussian Sampling more often samples the center pixels of the image, which are more informative for the network.

5.3   Comparative Analysis

In the context of comparing the two methods, one must first and foremost consider the complexity of the model. The Multidecoder method requires a second decoder, which increases the complexity of the model and the computational cost. This can be a problem for high-resolution and large-scale models, as it could mean that the model is too slow to train or too computationally expensive to use. On the other hand, Single decoder method does not require a second decoder, which does not introduce a lot of extra complexity to the model. This makes it more suitable for real-world applications.

For both of the methods applied to Gaussian VAEs with deeper encoder and decoder networks, the training instability in the form of the posterior collapse was observed, which means that latent variables have become uninformative, leading the model to ignore them [12]. This is a common problem in Gaussian VAEs and can be hard to avoid. However, this is something that is not a problem by design in VQ-VAEs, which makes them more stable and easier to train.

Both methods brought improvements to the quality of the reconstruction and the generalization capabilities when applied to VQ-VAEs. However, the Single decoder method showed better results in terms of the VQ objective loss, while Multidecoder method showed better results in terms of the reconstruction quality. This could be explained by the fact that Multidecoder method has two decoders, which can make it better at the core task of the VQ-VAE, which is to reconstruct the image.

5.4   Limitations and Future Work

One of the limitations of this research is that the experiments could have been conducted on a much wider range of hyperparameters and datasets. This could have given a better understanding of the methods and their capabilities and limitations. Another limitation is that the experiments were conducted on relatively small datasets, which could have affected the results.

One limitation of this research is that there could have been more experiments comparing different sampling methods. This could have given a better insight into the differences between the sampling methods and how they affect the results of the model.

One possible future direction for this research could be to investigate the possibility of using a training schedule where the model is initially trained with Multidecoder method and then later switched to a regular VQ-VAE or Gaussian VAE. This could potentially improve the results of the model for unconditional generation tasks.

Chapter 6

Conclusion

Bibliography

[1] Jinbo Bi, Tao Xiong, Shipeng Yu, Murat Dundar, and R. Bharat Rao. An improved multi-task learning approach with applications in medical diagnosis. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 117–132, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-87479-9.

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

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

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

[5] Kristian Gundersen, Anna Oleynik, Nello Blaser, and Guttorm Alendal. Semiconditional 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.

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

[7] A. Ali Heydari, Craig A. Thompson, and Asif Mehmood. Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions, 2019.

[8] 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`.

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

[10] 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`.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[12] James Lucas, George Tucker, Roger Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse, 2019.

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

[14] 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`.

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

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

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

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

[19] 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`.

[20] Yixin Wang, David M. Blei, and John P. Cunningham. Posterior collapse and latent variable non-identifiability, 2023.

[21] Y. Watanabe, H. Shirai, and Y. Kamide. Geomagnetic disturbances as probabilistic nonlinear processes. In Huaning Wang and Ronglan Xu, editors, *Solar-terrestrial Magnetic Activity and Space Environment*, volume 14 of *COSPAR Colloquia Series*, pages 281–286. Pergamon, 2002. doi: https://doi.org/10.1016/S0964-2749(02)80170-6.
    URL: `https://www.sciencedirect.com/science/article/pii/S0964274902801706`.

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