DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Designing Variational Autoencoders for Image Retrieval

SARA TORRES FERNÁNDEZ

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

The explosive growth of acquired visual data on the Internet has raised interest in developing advanced image retrieval systems. The main problem relies on the search of a specific image among large collections or databases, and this issue is shared by lots of users from a variety of domains, like crime prevention, medicine or journalism. To deal with this situation, this project focuses on variational autoencoders for image retrieval.

Variational autoencoders (VAE) are neural networks used for the unsupervised learning of complicated distributions by using stochastic variational inference. Traditionally, they have been used for image reconstruction or generation. However, the goal of this thesis consists of testing variational autoencoders for the classification and retrieval of different images from a database.

This thesis investigates several methods to achieve the best performance for image retrieval applications. We use the latent variables in the bottleneck stage of the VAE as the learned features for the image retrieval task. In order to achieve fast retrieval, we focus on discrete latent features. Specifically, the sigmoid function for binarization and the Gumbel-Softmax method for discretization are investigated. The tests show that using the mean of the latent variables as features gives generally better performance than their stochastic representations. Further, discrete features that use the Gumbel-Softmax method in the latent space show good performance. It is close to the maximum a posteriori performance as achieved by using a continuous latent space.

Sammanfattning

Den explosiva tillväxten av förvärvade visuella data på Internet har ökat intresse för att utveckla avancerade bildhämtningssystem. Huvudproblemet är beroende av sökandet efter en specifik bild bland stora samlingar eller databaser, och det här problemet delas av många användare från olika domäner, som brottsförebyggande, medicin eller journalistik. För att hantera denna situation fokuserar detta projekt på Variations autokodare för bildhämtning.

Variations autokodare (VAE) är neurala nätverk som används för oövervakat lärande av komplicerade fördelningar genom att använda stokastisk variationsinferens. Traditionellt har de använts för bildrekonstruktion eller generation. Målet med denna avhandling består emellertid i att testa olika autokodare för klassificering och hämtning av olika bilder från en databas.

Denna avhandling undersöker flera metoder för att uppnå bästa prestanda för bildåtervinning. Vi använder de latenta variablerna i flaskhalsstadiet i VAE som de lärda funktionerna för bildhämtningsuppgiften. För att uppnå snabb hämtning fokuserar vi på diskreta latenta funktioner. Specifikt undersöks sigmoidfunktionen för binärisering och Gumbel-Softmax-metoden för diskretisering. Testerna visar att med hjälp av medelvärdet av latenta variabler som funktioner ger generellt bättre prestanda än deras stokastiska representationer. Vidare visar diskreta funktioner som använder Gumbel-Softmax-metoden i det latenta utrymmet bra prestanda. Det ligger nära det maximala prestanda som uppnås genom att använda ett kontinuerligt latent utrymme.

Resumen

El exponencial crecimiento del número de imágenes digitales en internet ha elevado el interés en desarrollar sistemas avanzados de recuperación de imágenes. El principal problema reside en la búsqueda de una imagen específica entre grandes colecciones o bases de datos, hecho que afecta a grandes muchos usuarios de distintos sectores, como la prevención de crímenes, la medicina o el periodismo. Bajo el objetivo de proporcionar una nueva solución para la recuperación de imágenes, este trabajo emplea autocodificadores variacionales.

Los autocodificadores variacionales (VAE) son redes neuronales utilizados para el aprendizaje no supervisado de distribuciones complicadas, mediante el uso de inferencia variacional estocástica. Tradicionalmente se han empleado en el contexto de reconstrucción o generación de imágenes. Sin embargo, el objetivo de este trabajo consiste en la clasificación y recuperación de imágenes de una base de datos.

En este trabajo se han investigado varios métodos distintos con el objetivo de conseguir el mejor rendimiento posible relativo a aplicaciones de recuperación de imágenes. Para estas tareas de recuperación de imágenes se han empleado las variables latentes aprendidas en la capa que supone el cuello de botella del VAE. Más específicamente, se han investigado también tanto la función sigmoide como el método de discretización de Gumbel-Softmax. Las pruebas muestran que los mejores resultados se obtienen generalmente empleando la media de estas variables latentes en lugar de sus propias representaciones estocásticas. Además, los resultados obtenidos con la discretización mediante el método de Gumbel-Softmax muestran un buen desempeño, próximo al máximo a posteriori conseguido con un espacio continuo.

Contents

List of Figures

List of Tables

Notations

$\mathbf{X} \in \mathbb{R}^{\mathrm{D}}$ - Input data

$\mathbf{Z} \in \mathbb{R}^{\mathrm{N}}$ - Latent space

$\hat{\mathbf{X}} \in \mathbb{R}^{\mathrm{D}}$ - Reconstructed data

D - Input data dimensionality

N - Latent space dimensionality

B - Batch size

n2 - Size of the second layer of the encoder

n3 - Size of the third layer of the encoder

K - Number of images used for top-K closest

N_c - Number of classes for the categorical reparametrization

N_d - Number of categorical distributions for the categorical reparametrization

Chapter 1

Introduction

Humanity has tried to capture images from its very beginning, starting in the Paleolithic with cave paintings and following after with different techniques and styles through the different eras of human history. The evolution of human art at its beginning, specially referred to painting, can be summarize as the search of new techniques to show, how the world looked, as exactly as possible and with the biggest amount of details. This search was finally fulfilled when Louis Daguerre developed the daguerreotype process in 1839: the world could be immortalized exactly as it was.

The daguerreotype process can be seen as the precursor of what is nowadays known as photography, although many other different technical discoveries led to the invention of this art and to the first cameras. The photography has been evolving through the years, highlighting for example the development of color images or the change from analog to digital cameras. All this evolution, joint to the fact that the embedded cameras in the mobile phones are almost as good as a proper camera device, has allowed that each person is able to take great pictures to reflect the world that he or she is experiencing.

Thanks to the evolution of telecommunications, which led to the invention of the Internet, along with the previously mentioned development of the photography and the cameras, many image-based social media environments have been developed, such as Facebook, Instagram or Snapchat. This social networks have been widely adopted by most of the population, and just, in Instagram, more than 95 million of pictures and videos are uploaded everyday. All these facts contribute to the matter of having a colossal image library on the Internet.

One of the main problems that these enormous library presents is about image retrieval: how to select only images related to a desired topic. Through the years, many alternatives have been developed to solve this problem, but, with the boom of machine learning and pattern recognition algorithms in the latest years, this problem has been approached in that way. The main objective of this thesis is to provide a good and not too complicated solution for the image retrieval problem, by means of variational autoencoders and machine learning.

1.1 Applications

It looks clear that image retrieval represents a problem , and this section is focused in some of the possible applications for it. For it, the applications menctioned in [11] has been taken into account.

Firstly, one of the most important applications consists in medicine, since digital devices are being employed for taking medical images more and more. Even in the smallest hospitals, many different procedures generate medical images such as, for example, radiography or tomography. These procedures create a huge amount of gigabytes in a small time lapses, increasing the database of the hospital. This database requires a huge effort to be processed and classified for its different uses, which makes it a good application for image retrieval.

Another application consists of Biodiversity Information Systems (BIS). BIS features all kinds of data gathered by biologist all over the world for biodiversity studies, such as images of living beings or spatial data. The objective of this database is to help the researchers to complete their studies and enhance they knowledge about the different species and their habitat. With a good image retrieval algorithm, it would be easier to find all kind of images about a different specie, just by describing it.

A pretty interesting application as well is crime prevention. Image retrieval could be used for fingerprint identification, for example. In that case, the algorithm would successfully determine the closest fingerprints to one obtained at a crime scene, saving the detectives quite a lot of time for finding possible suspects. Another application in this area would be retrieving face images similar to one taken by a camera on a crime scene, and the procedure would be the same.

Image retrieval could be useful as well for digital libraries which support services based on image content. A great example of a digital library is The New York Public Library Digital Collections[1]. This library contains 743199 items digitalized from many different collections of photography, as well as manuscripts, fashion or nature collections, among others. These digital libraries could be used as well for historical research, since they also feature manuscripts which may be useful for that purpose.

Other possible application could be geolocation. In this aspect, the pictures could be a complement to GNSS systems to achieve a more accurate location. Along with this application it also comes tourism. In that way, the tourists could travel to a city, take pictures of the buildings and monuments and later, at home, be able to remember what these buildings were.

As it has been seen along this section, the applications are countless, which makes image retrieval a kind of necessity for the society. It is clear that a good method has to be designed and, once this has been done, there is no doubt that image retrieval would be part of the daily lives of humanity.

[1]https://digitalcollections.nypl.org/

1.2 Motivation

As it was previously mentioned, the number of users on the Internet uploading pictures has increased notably in the last few years. Along with this, the collections of digital images have experienced also a growth in their data. Having that enormous quantity of images on the Internet requires a good mechanism to browse, search and retrieve a particular element from such vast databases. Image retrieval consists in doing all those things, which makes it a necessity.

Another problem of those collections consists of the complexity of the data: each image can be interpreted in various ways. Therefore, another solution that image retrieval can provide would be making good classification systems to catalog the data correctly.

In the last few years, with the boom of machine learning and pattern recognition, different neural networks have been proved to correctly work for performing image retrieval. Many different systems and algorithms have been tested to achieve the best results, such as Generative Adversarial Networks or Convolutional Neural Networks. In this thesis, however, a variational autoencoder is presented to perform the image retrieval.

The main advantage that variational autoencoders present compared to other networks consists of its simplicity, which will be explained in more detail in Chapter 3. Another advantage that these networks present is that they are unsupervised learning algorithms since they are a kind of autoencoders. Furthermore, variational autoencoders have been proved to provide exceptional results for image generation so they may work as well for image retrieval.

1.3 Project Statements

There have been done many different tasks during this Master Thesis:

1. Study carefully of the previous literature related to image retrieval as well as to variational autoencoders, understanding the theory behind them so one could be created.

2. Implementation of a variational autoencoder and an algorithm to perform image retrieval with it, using Python and TensorFlow.

3. Adjustment of the settings of the image retrieval to improve the performance of the method propose.

4. Application of different variations of the method trying to achieve a better performance.

5. Writing of this report which includes the theoretical framework, the work done, the experimental results achieved and the conclusions extracted.

1.4 Outline

This thesis is structured as follows:

- **Chapter 2: Related Work and Background.** This chapter firstly focus on a brief literature review of related work. After that, the background required to understand this thesis is shortly explained in order to familiarize the reader with the concepts treated in this thesis. This background includes firstly a brief review of image retrieval, followed by a short description of neural networks. Then, both autoencoders and, more specifically variational autoencoders are presented.

- **Chapter 3: Variational Autoencoders Design for Image Retrieval.** Firstly, this chapter features the description of the variational autoencoder developed. Later, the different methods and approaches followed during the progress of this work are explained in order to ease their understanding.

- **Chapter 4: Experimental Results.** As its title indicates, in this chapter the experimental results of the variational autoencoder implemented as exposed, commented and compared. It also features an explanation of the settings employed for performing the image retrieval, as well as the database utilized.

- **Chapter 5: Conclusions and Future Work.** In the last chapter, a summary of all the content of this master thesis can be found, and the conclussions of the work are stated. Furthermore, some possible ways to improve the obtained results can be seen.

Chapter 2

Related Work and Background

The aim of this chapter is to provide a brief description of the background required to understand this thesis. It firstly includes a summary of the state-of-the-art approaches similar to this thesis, regarding image retrieval applications. After that, a description of image retrieval is found in order to familiarize the reader with the main objective of the thesis. Following it, an outline of the main concepts regarding neural networks and autoencoders can be find. Finally, an overview of the theory behind variational autoencoders can be find.

2.1 Related Work

As it was mentioned in Chapter 1, image retrieval has been widely researched in the environment of machine learning. There are mainly two different approaches when trying to perform image retrieval: using Generative Adversarial Networks and Convolutional Neural Networks and both will be exposed in this sections.

Generative Adversarial Networks or GANs are generative models: they learn how to copy the data distribution of the input data so they can generate images similar to that. They involve two competing networks: the discriminator and the generator. The first approach using GANs for image retrieval was [1], where a GAN architecture was presented with different design features to allow the system to correctly retrieve images.

Another approach was performed later using Binary GAN [2]. The employment of binary networks allows the retrieval process to use Hamming distance, which is simpler and faster than other distance metrics such as Euclidean distance. In order to perform this approach, Song introduced a new sign-activation strategy and a new loss function consisting of an adversarial loss, a content loss and a neighbourhood structure loss.

On the other hand, Convolutional Neural Networks or CNNs are neural networks composed by several filters which convolve the data, producing a feature map. The most interesting feature about CNNs however, is that they are spatially invariant since the filter weights do not change at different parts of the image. In this area, one of the first approaches can be found in [4], where they employed a Deep CNN achieving good results. Following with this approach, in 2014 a paper proposed exploiting multi-scale schemes for extracting local features using ConvNets [7]. A different approach consisted in aggregating deep convolutional features [6]. This method provided compact global descriptors in an efficient way. As well as with GANs, a binary approach employing CNNs was explained in [3], where Guo et al. introduced a hash layer in order to simplify the latter retrieval.

A completely different approach regarding CNNs can be found in [9], where a Siamese network is proposed. A siamese network consists of two or more identical subnetworks, with the same parameters and weights. The two networks are given two inputs, and the output module process their outputs to generate the final output of the network.

Finally, CNNs were used on [10] for trying to employ fine-tuning instead of training from scratch. This method was proven to achieve good results even for 3D models, without the need of human annotation.

2.2 Image Retrieval

In the last decades, image-related technologies have evolved extraordinarily, from old analog cameras to the digital ones which nowadays are broadly used, including those embedded in our mobile telephones. This evolution has been accomplished along with the development of the Internet, which also features the expansion of image-based social media applications such as Facebook or Instagram. With this tools the digital image collection available currently on the Internet is huge, and it keeps growing more and more every day. As an immediate consequence, it seems clear that the available collection is almost impossible to manage, which makes image retrieval a necessity for many areas such as medical imaging or advertising.

Image retrieval consists in the browse, search and retrieval of images from a vast database of digital images. It can be exact or relevant [12]. Exact image retrieval requires the images to be matched exactly whilst relevant is based on the contents on their contents and can be more flexible, depending on the scale of relevance required.

There are two main frameworks in the context of image retrieval: text-based and content-based [13, 14]. Text-based image retrieval resides in manually annotate the images and then use a database management system (DBS) to retrieve them. It can be easily seen that this approach presents two main disadvantages: the need of an enormous human labour and the inaccuracy caused by human perception. On the other hand, content-based image retrieval (CBIR), consists in classifying the images based on their visual content, like colour, texture, shapes... The main difference between this two frameworks lies in the need or not of human interaction. In the first one, as human interaction is required, the images will be labeled with high-level features or concepts such as keywords or text descriptions whereas in CBIR those features will be low-level concepts like the ones described previously.

With the growth and development of machine learning, and specially deep learning techniques, many of the networks and systems developed had been tested in the environment of image retrieval. In this thesis, variational autoencoders will be tested in order to achieve a good image retrieval system.

2.3 Artificial Neural Networks

Artificial Neural Networks (ANN) are computing systems firstly inspired by biological neural networks [15]. The biological neural networks comprise the neurological system that forms the brains of many animal species. Therefore, ANN tries to export this system to the computer area. Although they were firstly described in the 1940s [16], ANNs have experienced an increasing interest due to machine learning algorithms.

The main objective of ANNs is to "learn" how to perform a determined task, and, by means of several iterations, the backpropagation and loss functions optimize the learning process by minimizing the latter. They can be formed by one input layer, one output layer and one or more hidden layers layers. When the ANN has more than one hidden layer it is called a Deep Neural Network (DNN).

The model of feed-forward neural networks can be described as a series of transformations. The first transformation consists on performing M linear combinations of the input vector, described as $x = (x_1, x_2, .., x_D)$, where M is the size of the hidden layer. This transformation can be seen on Eq. 2.1, where $w_{j,i}$ are the different weights of the network which are updated after each iteration in the case of deeper networks. Each one of this layers contains one or more "neurons".

$$a_j = \sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}, \forall j = 1, ..., M \tag{2.1}$$

Then, the network reaches the first hidden" layer $z = (z_1, ..., z_M)$. This hidden layer consists of a transformed a_j by means of a differentiable, non-linear transformation function: $z_j = h(a_j)$. If the network was deeper, there would be more hidden layers and they would be iterated as the example above. If there are more hidden layers, the proccess would repeat until the last one is activated.

After the last hidden layer, the components of z are used to create K linear combinations along the decoder layer, where K denotes the dimensionality of the output vector $y = (y_1 ... y_K)$. This transformations are done accordingly to Eq. 2.2.

$$a_k = \sum_{j=1}^{M} w_{k,j}^{(2)} z_j + w_{k,0}^{(2)}, \forall k = 1, ..., K \tag{2.2}$$

The predicted output of the network would be $\hat{y} = a$, and the objective loss function that requires to be minimized would be $L = \sum_{i=1}^{K} (y_i - \hat{y}_i)^2$. An example of a neural network with 1 hidden layer, extracted from [17], can be seen on Fig. 2.1.

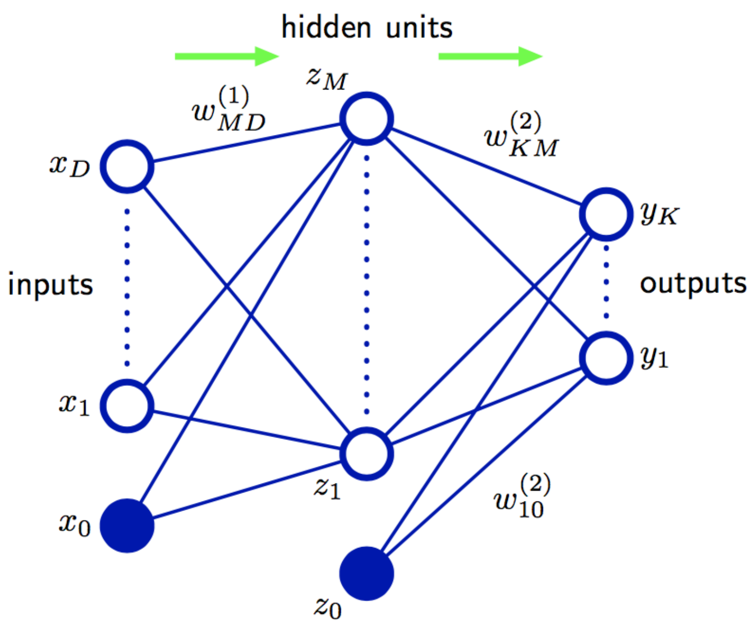

Figure 2.1: Structure of a neural network

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks or CNNs, are a kind of deep and feed-forward neural networks. They have been used usually to solve computer vision and machine learning problems such as image classification. Their main advantage in this area is that CNNs require less preprocessing than other algorithms. Another advantage consists of their shift-invariant property, which allows them to find patterns in different parts of the image.

As well as regular ANNs, convolutional neural networks are formed by an input layer, an output layer and one or more hidden layers.CNNs involve many different operations to form the hidden layers which process the image at the input to generate the expected output: convolution, activation and pooling.

The convolution function is a discrete operation which convolves the input x with a filter w. The output of this operation is called a feature map, and represents the cross-correlation between the filter's pattern and the local features of the input. Since this operation is translation invariant, the same features can be detected in different parts of the image.

On the other hand, the activation function introduces a non-linear property which allows the CNN to learn complex mappings from the input. Both the convolution and the activation functions are usually joint in a so called convolutional layer.

Finally, the pooling function composes the pooling layer, which is located after the convolutional layer. The most used pooling function is max-pooling, which extracts the maximum values in each N x N block (where N is the size of the filter) from the output of the convolutional layer. An example of max-pooling with a 2x2 filter and stride 2 can be seen on Fig. 2.2.

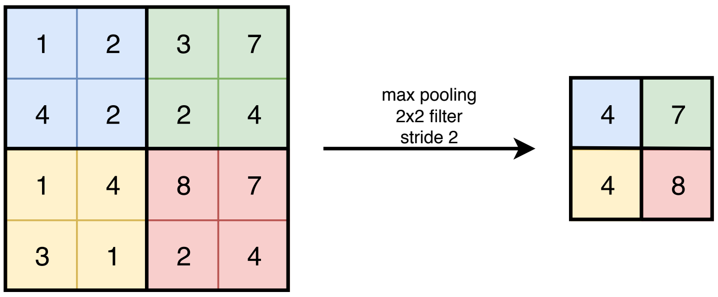

Figure 2.2: Example of max-pooling

2.4 Autoencoders

Autoencoders are unsupervised learning algorithms where the expected output is an exact copy of the input. In order to achieve this, autoencoders find a latent representation of the data which tries to learn good representations from the data. This hidden or latent layer has smaller dimensions than the input data, to prevent the autoencoder from learning the identity transform which might be a trivial solution to make the input and the output identical.

Basically autoencoders consist of two symmetric neural networks: an encoder and a decoder. The encoder maps the input x into a latent space z also known as bottleneck layer due to its smaller dimensions compared with the input data. Then, the decoder will take this latent space to reconstruct and generate the input data $\hat{x}$.

It seems clear that the goal of the reconstruction is to minimize the difference between x and $\hat{x}$, also known as reconstruction loss. To ensure it, the latent space must learn the most important feature variations of the original data so the reconstruction would be similar enough to the original data. In order to achieve a minimum reconstruction loss, the whole network (encoder and decoder) is trained jointly.

Autoencoders can have a single layer in both the encoder and the decoder but commonly this networks are formed with two or more layers. In this case, the autoencoders are known as deep autoencoders (2.3).

There are four main types of autoencoders: denoising autoencoders, sparse autoencoders, variational autoencoders and contractive autoencoders.

Denoising autoencoders [21], as their name specifies, try to denoise or recover a clean image from a randomly partially corrypted input. The idea behind this kind of autoencoders is basically to force the hidden layer to learn robust features, and preventing it from just learning the identity function which would result in another corrupted image.

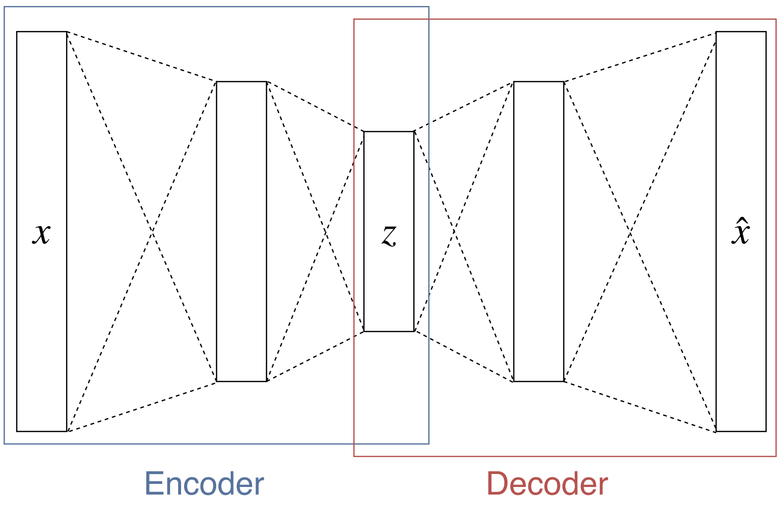

Figure 2.3: Structure of a deep autoencoder

Sparse autoencoders [22] are characterized by having a hidden layer with higher dimensions than the input. The problem of avoiding the network to learn the identity function is solved by only allowing a few number of the hidden neurons to be active at the same time.

Contractive autoencoders or CAEs [23] are slightly more complex since they add a new term to their loss function in order to achieve a model more robust to slight variations in the input values.

Finally, variational autoencoders or VAEs have the architecture of autoencoders (Fig. 2.3) but they use a variational approach for the learning. In the following section variational autoencoders will be explained more deeply.

2.5 Variational Autoencoders

The main difference between variational autoencoders and the other kinds of autoencoders consists in using a variational approach for latent representation learning. VAEs have proved to ensure great systems for image generation but they will be tested for image retrieval in this thesis.

2.5.1 Advantages and Disanvantages of Variational Autoencoders

As it was mentioned in Chapter 1, the main advantages of VAEs rely on being algorithms for unsupervised learning. Unsupervised learning is the natural procedure that cognitive mammals, i.e. human beings use for learning, which is makes it an interesting alternative for machine learning and artificial intelligence. This consists on the network discovering the features of the data on its own, using later those features to classify the data. In this way, there is no need to define beforehand an input and output dataset, like in supervised learning.

It was also mentioned that VAEs have simple structures, which is an advantage compared to Generative Adversarial Networks. In this way, they are easier to train, joint to the fact that VAEs have a clear objective function to optimize (log-likelihood).

Another advantage that variational autoencoders present against GANs is that the quality of their models can be evaluated by means of the log-likelihood (explained in the following sections), whilst GANs cannot be compared except by visualizing the samples.

However, VAEs present a drawback in terms of reconstruction since the generated images are blurred when compared from the ones generated by GANS. This blurred is caused by the imperfect reconstruction achieved by variational autoencoders.

2.5.2 Problem Formulation

Variational autoencoders are probabilistic generative models: both the input and the latent space are supposed to be random variables characterized by probability distributions.

The problem formulation can be seen from a graphical model perspective, using graph theory to show the dependency between random variables. There is a dataset $\mathbf{X} = \{x_i\}_{i=1}^{N}$, composed with N samples from a random variable $\mathbf{x}$, which can be continuous or discrete. This dataset is related with the hidden continuous random variable $\mathbf{z}$ by means of the probabilistic graphical model (PGM) showed in Fig. 2.4, whose joint probability can be noted in Eq. 2.3.

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2.3}$$

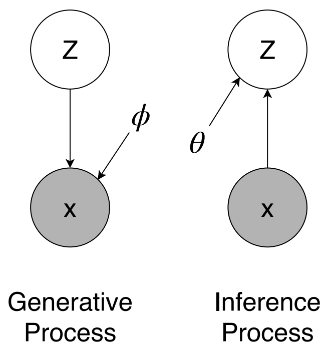

Figure 2.4: Graphical model representation in the VAE. a) Generative process. b) Inference process

According to the generative process of the PGM, the latent variables are generated by sampling a random variable z_i from a prior distribution $p(\mathbf{z})$, whereas the datapoints x_i are obtained afterwards from a conditional distribution over $\mathbf{z}$, $p(\mathbf{x}|\mathbf{z})$. Both the prior and the likelihood are usually defined as Gaussian distributions:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \tag{2.4}$$

and

$$p_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{x}|f(\mathbf{z},\theta)), \sigma^2\mathbf{I}), \tag{2.5}$$

where $f(\mathbf{z},\theta)$ represents a neural network.

However, the objective is to achieve a correct latent space $\mathbf{z}$ given the observed data, i.e. calculating the posterior probability $p(\mathbf{z}|\mathbf{x})$. According to Bayes,

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}, \tag{2.6}$$

where

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x},\mathbf{z}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz. \tag{2.7}$$

Nevertheless, this last equation requires exponential time to compute since it requires all the possible configurations of $\mathbf{z}$. The solution proposed is to approximate it with a simpler distribution according to the Variational Bayesian Inference Method, for example, with a Gaussian distribution like the one described in Eq. 2.8. Yet, by applying this approximation, the total loss of the system is increased, and a new term is added to the previously mentioned reconstruction loss: the latent loss. This latent loss is calculated by means of the Kullback-Leibler divergence, which will be explained as follows.

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}(\mathbf{x},\phi), \boldsymbol{\sigma}^2(\mathbf{x},\phi)\mathbf{I}) \tag{2.8}$$

2.5.3 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence is a non symmetrical measure of the similarity or difference between two different probability functions. It can be defined accordingly to Eq. 2.9, and the result shows the information in nats lost when approximating a function q to approximate p.

$$KL(p(x)||q(x)) = -\sum q(x)\log\frac{q(x)}{p(x)} \tag{2.9}$$

KL divergence has two main properties: the first is that $KL(p||q) \geq 0$ since, when p and q are equal, $KL(p||q) = 0$. The second property is, as it was mentioned before, that the KL diveregence is assymetric, i.e., $KL(p||q) \neq KL(q||p)$.

In the case of the problem stated before, KL divergence is calculated as in Eq. 2.10.

$$\begin{aligned} KL(q_\phi(\mathbf{z}|\mathbf{x})||p(q_\phi(\mathbf{z}|\mathbf{x}))) = \mathbf{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right] - \mathbf{E}_{\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z})\right] \\ + \log p(\mathbf{x}) \end{aligned} \tag{2.10}$$

The goal then is to minimize the KL divergence by finding the optimal variational parameters λ. However, it can be noted that the unknown $p(\mathbf{x})$ appears in that divergence. In order to solve this problem, the posterior inference can be approximated by combining this KL divergence with the Evidence Lower Bound or ELBO.

2.5.4 Evidence Lower Bound

To define the evidence lower bound or ELBO the first step consists in factorizing the marginal likelihood as in Eq. 2.11.

$$\log p(\mathbf{X}) = \log \prod_{i=1}^{N} p(x_i) = \sum_{i=1}^{N} \log p(x_i) \tag{2.11}$$

For each one of the datapoints, this likelihood can be defined accordingly to Eq. 2.7, and multiplying and dividing by the posterior approximation,

$$\begin{aligned} \log p_\theta(x_i) = \log \int p_\theta(x_i, \mathbf{z}) d\mathbf{z} = \log \int \frac{q_\phi(\mathbf{z}|x_i) p_\theta(x_i, \mathbf{z})}{q_\phi(\mathbf{z}|x_i)} d\mathbf{z} = \\ = \log \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\frac{p_\theta(x_i), \mathbf{z})}{q_\phi(\mathbf{z}|x_i)} \right] \end{aligned} \tag{2.12}$$

By applying Jensen's inequity (Eq. 2.13), a lower bound can be obtained in the previous equation (Eq. 2.14), and in that way the ELBO can be defined as in Eq. 2.15.

$$\psi(\mathbf{E}[x]) \geq \mathbf{E}[\psi(x)] \tag{2.13}$$

$$\log \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\frac{p_\theta(x_i), \mathbf{z})}{q_\phi(\mathbf{z}|x_i)} \right] \geq \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\log \frac{p_\theta(x_i), \mathbf{z})}{q_\phi(\mathbf{z}|x_i)} \right] \tag{2.14}$$

$$ELBO(x_i) = \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\log p_\theta(x_i|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|x_i) \right] \tag{2.15}$$

Identifying terms with Eq. 2.10, the ELBO for each point can be written as:

$$ELBO(x_i) = \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\log p_\theta(x_i|\mathbf{z}) \right] - KL(q_\phi(\mathbf{z}|x_i)||p(\mathbf{z})) \tag{2.16}$$

Finally, the ELBO for the whole dataset would be

$$ELBO(\mathbf{X}) = \sum_{i=1}^{N} \mathbf{E}_{q_\phi(\mathbf{z}|x_i)} \left[\log p_\theta(x_i|\mathbf{z}) \right] - KL(q_\phi(\mathbf{z}|x_i)||p(\mathbf{z})) \tag{2.17}$$

The objective of the model is to maximize the objective function, optimizing it using stochastic gradient descend. However, it is not possible to take derivatives of a distribution with respect to its parameters. For this purpose, a "reparametrization trick" was proposed in [29].

2.5.5 Reparametrization Trick

The samples $\mathbf{z}$ is obtained from the distribution $q_\phi(\mathbf{z}, \mathbf{x})$ but, as it was mentioned before, it is not trivial how to take the derivatives of a function of $\mathbf{z}$ with respect to ϕ.

The solution could be reparametrizing this $\mathbf{z}$ so the stochasticity is independent on the parameters of the distribution as it is possible for some distributions. It can be done with an auxiliary noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$:

$$\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}, \phi) + \boldsymbol{\sigma}(\mathbf{x}, \phi)\boldsymbol{\epsilon} \tag{2.18}$$

By taking Monte Carlo estimates, the expectation would be:

$$\tilde{ELBO} = \sum_{i=1}^{N} \left[\frac{1}{L} \sum_{l=1}^{L} [\log p_\theta(x_i|z_{i,l})] - KL\left(q_\phi(\mathbf{z}|x_i)||p(\mathbf{z})\right) \right] \tag{2.19}$$

where $z_{i,l} = \boldsymbol{\mu}(x_i, \phi) + \boldsymbol{\sigma}(x_i, \phi)\boldsymbol{\epsilon}_{i,l}$.

As the two distributions in the KL-divergence term are Gaussian distributions, it can be calculated as:

$$KL\left(q_\phi(\mathbf{z}|x_i)||p(\mathbf{z})\right) = -\frac{1}{2}\sum_{k=1}^{K}(1 + \log(\boldsymbol{\sigma}_k^2(x_i, \phi) - \boldsymbol{\mu}_k^2(x_i, \phi) - \boldsymbol{\sigma}_k^2(x_i, \phi)) \tag{2.20}$$

The Gaussian likelihood reconstruction term is:

$$\log p_\theta(x_i|z_{i,l}) = -\frac{1}{2\sigma^2}(x_i - f(z_{i,l}, \theta))^2 + constant \tag{2.21}$$

Finally, the estimation of the ELBO from a random data batch of size B would be:

$$ELBO(\mathbf{X}) = \tilde{ELBO}(\mathbf{X}^B) = \frac{N}{B}\sum_{i=1}^{B} \tilde{ELBO}(x_i) \tag{2.22}$$

2.6 Gumbel-Softmax Trick

As a final addition to this work, the latent space is quantized to test its performance without a continuous space. For this purpose, the Gumbel-Softmax Distribution is applied. Therefore, in this section, a brief introduction to this distribution and the literature regarding it will be exposed.

This distribution was firstly defined in 1954 by E.J. Gumbel [38], and has the advantage that, by means of the Gumbel-Max trick [39, 40, 41, 42, 43], can be deformed into a discrete distribution. In this way, discrete values can easily be extracted from a continuous space, which allows the system to work with a discrete latent space rather than the continuous one used until this point.

The trick works as follows: firstly, the different states considered are vectors $d \in 0, 1^n$ of bits. These vectors are one-hot i.e. $\sum_{\{} j = 1\}^n d_j = 1$.

An unnormalized parametrization is considered $(\alpha_1, ..., \alpha_n$, where $\alpha_j \in (0, \infty)$, from a discrete distribution $D \sim \text{Discrete}(\alpha)$, with 0-probability states excluded.

The Gumbel-Max trick then consists in sampling $U_j \sim \text{Uniform}(0, 1)$ or, in other words, find the j that maximizes $\log \alpha_j - \log(-\log U_j)$, having set $D_j = 1$ and $D_i = 0$ for $i \neq j$. After this,

$$P(D_j = 1) = \frac{\alpha_j}{\sum_{i=1}^{n} \alpha_i} \tag{2.23}$$

The name of this trick has its explanation since $-\log(-\log U)$ has a Gumbel distribution.

Chapter 3

Variational Autoencoders Design for Image Retrieval

As it was stated before, the aim of this thesis is to show the performance of variational autoencoders for image retrieval applications. This chapter is focused in the description of the variational autoencoder implemented, as well as the different methods and approached tested for image retrieval.

3.1 Implementation of a Variational Autoencoder

The VAE utilized along this thesis was implemented with Python, using Tensorflow [24] for the training of the models. The main objective is to train an end-to-end system so the latent space z characterizes the differences between the different classes or numbers in the database. After that, these vectors z can be compared to determine the closest images and therefore, detect the class of the image by checking their labels.

The variational autoencoder designed consists of two layers in both the encoder and the decoder. The first step is to initialize the encoder weights and biases by means of a Xavier initialization [25], to be able to calculate the mean and the standard deviation of the Gaussian distribution in the latent space (μ_z and σ_z). Later, z is calculated as in Eq. 2.18. This latent space will be updated after each training epoch, and the training is performed incrementally with mini-batches of the input data.The trained model obtained can be used to reconstruct the input, to generate new samples and to map inputs to the latent space. This third application is the one exploited in this thesis.

The loss function is composed of two terms as it was stated in Section 2: the reconstruct loss and the latent loss. The reconstructions loss can be seen as the difference between the input and the reconstruction given by the decoder. On the other hand, the latent loss is defined as the KL-divergence between the distribution in the latent space and the input data. The objective of the system is to try to optimize this loss function by minimizing it using ADAM optimization algorithm [26]. The structure of the variational autoencoder implemented is shown in Fig. 3.1

Once the VAE is implemented, the first step consists in extract the features and queries from the latent space. This features are the vector z for the 50000 images of the training set whilst for the queries, the 10000 images of the test set were the ones utilized.

3.2 Latent Space

The latent space z is a space in N dimensions, where N can be any natural number defined by the user. It is calculated according to Eq.2.18; i.e., $z = \mu_z + \sigma_z \epsilon$, and both μ_z and σ_z are obtained by training the VAE. It consists in characterizing the features of each one of the classes like, for example, shape of the digit, its angle or the stroke width among others.

A way to see the latent space is to use the generator network to plot reconstructions of the images in the latent space for which they were generated (Fig. 3.2). It can be seen that each one of the classes is generated in a different area of the space, ensuring that the latent space correctly detects the differentiating characteristics of the ten distinct classes studied.

The latent space is continuous, and the values for each one of its elements is in the range [-4-4]. For example, a 2D latent space can be seen on Fig. 3.3. This latent space was generated using 10000 elements from the training set of MNIST, choosing N=2 as the dimensionality of the latent space. In this figure the difference between the different classes of the MNIST database can be spotted since each class occupies a different place in the representation. However, it needs to be mentioned that some of the classes can be confused, i.e. they are too close to each other and their representation is mixed. One example of this could be between classes 4 and 9 or 3 and 5 and it is explained by the similitude of those numbers. By seeing this figure it is clear that taking the K-closest images can be a great tool to detect the class of each image.

3.3 Methods and Approaches Developed

Throughout the development of this thesis, many approaches were tested to determine the best configuration for image retrieval applications. In this section, all the studied methods and approaches are briefly described so the results shown in Chapter 4 can be fully understood.

The first step, however, consisted in training the variational autoencoder. An scheme of the training process can be seen on Fig. 3.4. In red, there are marked the two components of the loss function employed: the latent loss (Kullback-Leibler divergence) and the reconstruction loss (difference between x and $\hat{x}$). This terms are intended to be minimized along the training process. On the other hand, in blue there are highlighted the latent space variables generated, that will be used in the following steps for performing the image retrieval.

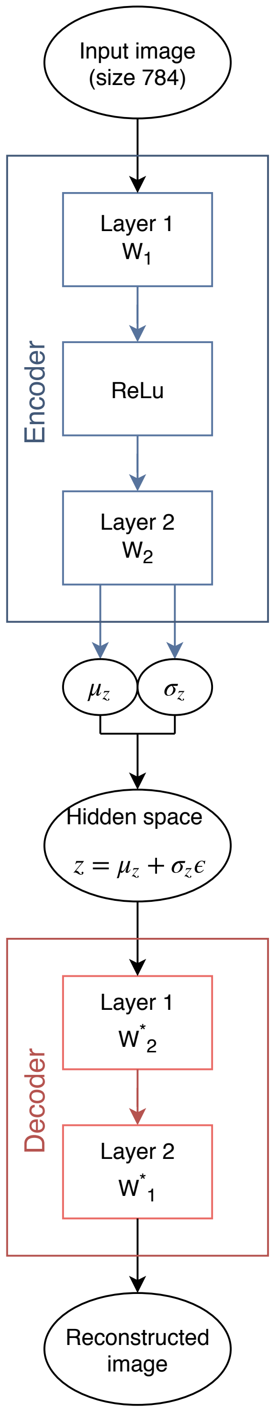

Figure 3.1: Structure of the variational autoencoder implemented

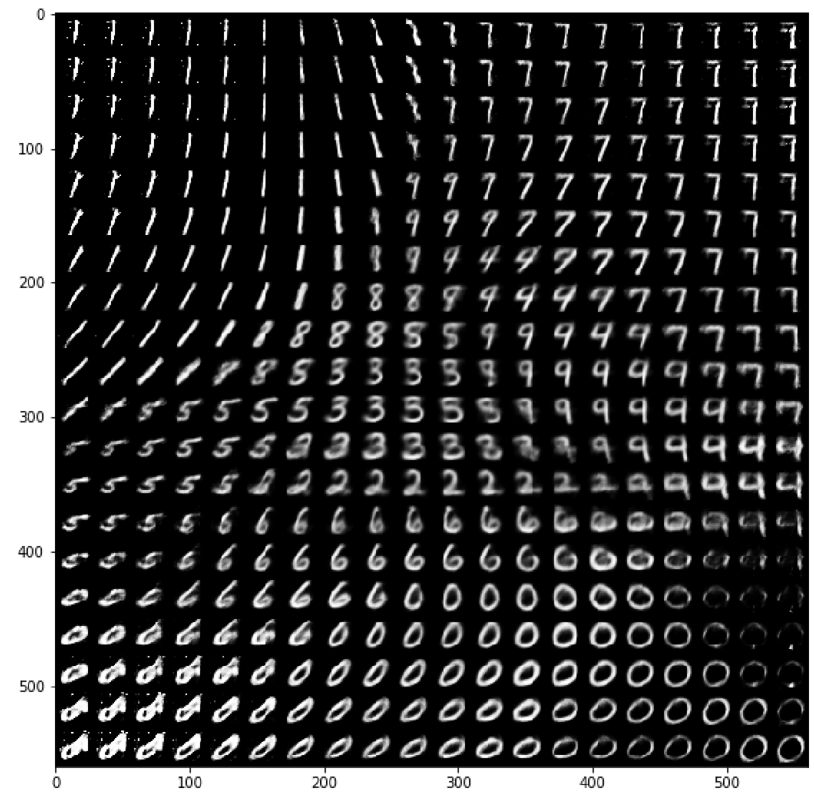

Figure 3.2: Latent space reconstructions

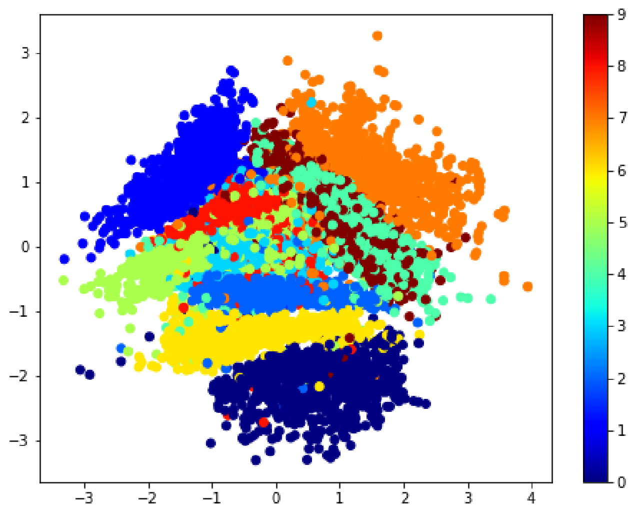

Figure 3.3: 2D latent space

3.3.1 Default Approach

As a first approach, knowing that the latent space is continuous and that each one of the ten classes is primarily concentrated in a different region, comparing the distance between the points in the latent space seems to be useful to determine each class. Indeed, to determine the class of a particular element i from the test set, the method chosen consists of comparing its latent space z_i with all the latent spaces from the training data. This can be easily done calculating the Euclidean distance between the two spaces, since the elements z are all vectors of size N.

Once all the distances have been calculated, the easier way to determine the class of the element is taking the images with smaller distance, checking the top K items in the ranking list and then looking at their labels. In this way, the most repeated label will ideally correspond to the element from the test that was compared. A diagram showing this steps can be seen on Fig. 3.6.

3.3.2 Binarization

It must be noted that, if instead of being continuous the latent space was binary (the vector z would only consist of zeros and ones), a simpler approach could be made. This approach would be calculating the Hamming distance between the vectors, and taking the K elements with smaller distance, as well as it was done in the environment of a continuous space.

Hamming distance is a measure employed fundamentally in information theory, to calculate the difference between two codewords of the same length. It is defined as the number of digits that should change to transform one codeword into the other, i.e., the number of digits that differ between the two codewords.

This binary approach was also considered along the development of this thesis as an experiment to obtain better and faster results. It was implemented in two different ways: the first one consists of the binarization of the vectors z obtained after the training, and the second in embedding the binarization so the obtained z vector is already binary.

For the first approach, as z values are within the range $[-4, 4]$, the floating point values were transformed into -1s or 1s after the training, according to the following criteria:

- If the value is smaller or equal to 0, it will become a -1.
- If the value is greater than 0, it will become a 1.

On the other hand, for the embedded binarization the method considered consists in directly binarizing the output of z, employing the function $sgn(\cdot)$, defined in Eq. 3.1. However, this method can present issues in the backpropagation (vanishing gradient problem): This is caused since the sign function is non-smooth and non-convex, and due to the fact that the gradient of this function is zero for all nonzero inputs. This problems can be seen in Fig. 3.5.This figure shows that all the 10 different classes from the database are merely compressed in 4 points. As the classes are overlapping in the space, the retrieval does not seem to be effective.

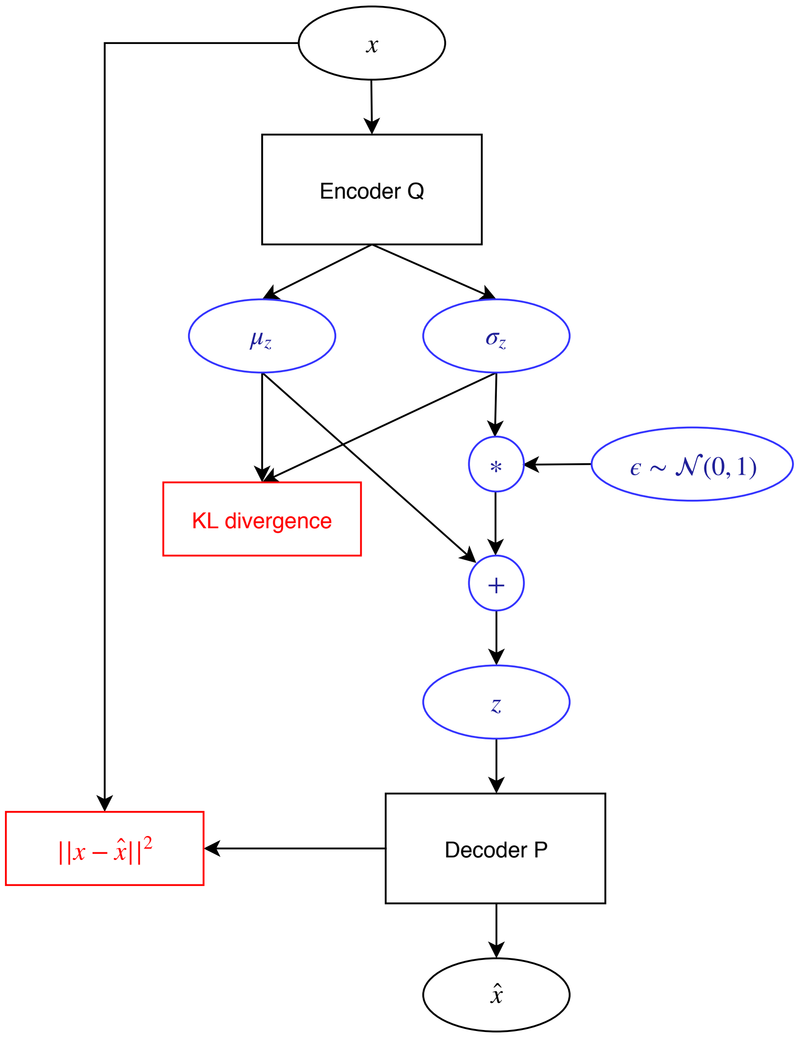

Figure 3.4: Diagram of the training process

$$sgn(z) = \begin{cases} +1, & \text{if } z \geq 0 \\ -1 & otherwise \end{cases} \tag{3.1}$$

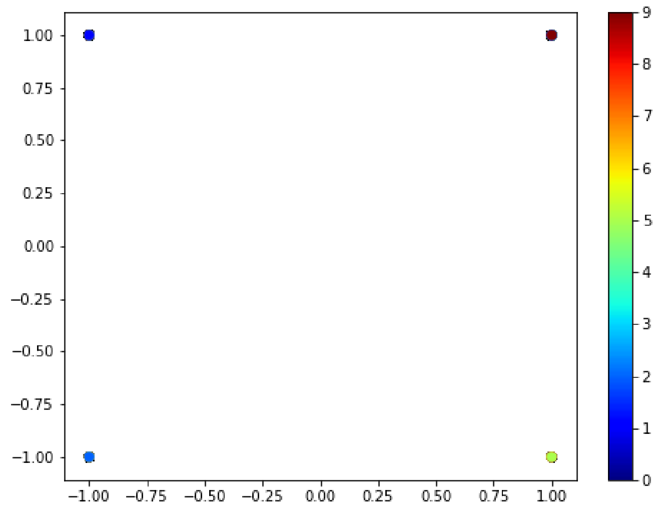

Figure 3.5: Latent space for the first embedded binarization

3.3.3 Different Latent Representations

In order to find the best results, several latent representations were tested. For this purpose, the main variations were done to the size of the vector z.

This variations in the dimensions of the latent space (N) are useful to determine the variation of how good the representation of the data is depending on the number of dimensions that features the latent space.

Another method employed consists in reducing the dimensions of the latent space, as it was previously exposed in [2]. In that paper Song designed a hashing layer to "binarize" the hidden space. The solution propose consists in approximating the $sign(\cdot)$ function with a new function called $app(\cdot)$, which is defined in Eq. 3.2. With this function, the latent space obtained (Fig. 3.7) is more similar to the one achieved without the binarization (Fig. 3.3), but compressed in the range $[-1, 1]$. Nevertheless, it can be seen that the overlapping issues between similar classes (like 4 and 9 or 5 and 3) are still existent. The best characteristic of this method would consist in a faster retrieval than the one using the continuous space with values between -4 and 4.

$$app(z) = \begin{cases} +1, & \text{if } z \geq 1 \\ z, & \text{if } 1 > z > -1 \\ -1, & \text{if } z \leq -1 \end{cases} \tag{3.2}$$

Number of levels	**Possible values**
2	-1, 1
4	-3, -1, 1, 3
8	-3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5
16	-3.75, -3.25, -2.75, -2.25, -1.75, -1.25, -0.75, -0.25, 0.25, 0.75, 1.25, 1.75, 2.25, 2.75, 3.25, 3.75
32	-3.875, -3.625, -3.375, -3.125, -2.875, -2.625, -2.375, -2.125, -1.875, -1.625, -1.375, -1.125, -0.875, -0.625, -0.375, -0.125, 0.125, 0.375, 0.625, 0.875, 1.125, 1.375, 1.625, 1.875, 2.125, 2.375, 2.625, 2.875, 3.125, 3.375, 3.625, 3.875

Table 3.1: Possible values for the different number of levels in the quantization

The last change in the latent space that was studied consists in using an approximation to z as $z = \mu_z$ instead of the one stated in Eq. 2.18, i.e, $z = \mu_z + \sigma_z \epsilon$. In this case, the results might be better due to the suppression of the random component in z and because of considering only the centroid of the distributions. The latent space for this approximation can be seen in Fig. 3.8, and it can be seen that it is pretty similar to the one obtained with the default method (Fig. 3.3).

3.3.4 Quantizing the latent space

Following with the binarization approach, it could be interesting to test the variational autoencoder with a quantized latent space. This could be useful for example if this method was desired to be implemented in a DSP environment. The quantization was done by quantizing the latent space after the training.

For this purpose, linear uniform quantizer was used, varying the number of levels to see the performance of having a more or less accurate representation. Linear uniform quantizers have as many intervals as the number of levels previously defined, and all those intervals have the same size. One example of a quantizer can be seen on Fig. 3.9. The possible levels tested were 2 (binarizing), 4, 8, 16 and 32, as well as ∞ (no quantization), and the possible values determined by those levels are stated on Table 3.1.

As all this options have more than 2 possible values, hamming distance is no longer an option besides the case with only 2 levels. Therefore, the distance employed would be again the Euclidean distance, as in most of the experiments performed. The only difference with the considered cases before is that, as all the levels are quantized, the distances have fixed values, making it easier to determine the closest images.

3.3.5 Discretization of the latent space

Another approach tested consisted in embedding a discretization of the latent space, while doing the training. This was done following the method proposed in [45], called Categorical reparametrization with Gumbel-Softmax distributions. It consists of a simple technique which allows to train neural networks with discrete latent variables.

Categorical reparametrization with Gumbel-Softmax Distribution

In the cited work [45], Yang et. al. proposed a ”reparametrization trick” similar to the one explained in section 2.5.5, but for the categorical distribution: they smoothly deform the Gumbel-Softmax distribution into the categorical distribution desired.

The first step consists of using the Gumble-Max trick [38, 43], which efficiently draws samples z from the categorical distribution with class probabilities π_i, as in Eq. 3.3.

$$z = one\_hot(\arg\max_t[g_t + \log \pi_t]) \tag{3.3}$$

Since arg max is not differentiable, the next step resides on using the softmax function as a continuous approximation for it (Eq. 3.4), calling it the Gumble-Softmax distribution. This distribution was discovered at the same time in [46], naming it the Concrete distribution.

$$y_i = \frac{exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} exp((\log(\pi_j) + g_j)/\tau)} \tag{3.4}$$

In the previous equation, τ is a temperature parameter which allows to control how closely the samples from the Gumbel-Softmax distribution approximate the ones from the categorical distribution. When $\tau \to 0$, these distributions become the same. However, in order to allow the backpropagation to compute gradients, $\tau > 0$, so finally the authors, after many experiments defined a value of $\tau = 1$.

After performing this categorical reparametrization, the latent space obtained differs widely compared with the one obtained without it. As it was explained through Section 3.2, the latent space without categorical reparametrization (for the default experiment) consists of a vector formed by continuous values between [-4, 4]. This latent space was the one considered unless it was stated otherwise througout this chapter.

However, since it was mentioned before, the Gumbel-Max trick yields as an output one-hot vectors, i.e., only one of all the possible positions in the vector equals to one whilst all the rest is equal to zero. With this in mind, it could be seen that this latent space represents now an index of all the different classes in the database.

Therefore, using the Euclidean distance at the retrieval phase would no longer have any sense since the latent space represents the index of the classes and not just its position in the hidden space. For this purpose, since the vector can only have the values 0 or 1, Hamming distance can be useful to determine if the images tested belong to the same class or not.

In this case, only if the Hamming distance equals to 0 the images are considered to belong to the same class and, on the contrary, if Hamming distance is bigger than 0, the images are discarded since they should not be from the same class as the tested one. This mechanism also helps to improve the speed at the retrieval, as it was mentioned before.

Another difference between using this reparametrization and not using it relies on the KL divergence. While in the default experiment the KL divergence was calculated compared to a Gaussian distribution, in this case it would no longer be possible, since the latent space does not approximate to this kind of distribution but to a Categorical one.

If the Gaussian distribution was still used to calculate the KL divergence, the results would widely differ from the ones expected since the approximation would no longer be accurate. Since the latent space in this embedded discretization method correspond to a Categorical or Gumbel-Softmax distribution, the latent loss term is required to adapt to this.

Therefore, the KL divergence which corresponds to the latent loss is calculated between two categorical distributions. In this way, the difference at the output drops to have similar values than the ones achieved without the embedded discretization.

This Kullback-Leibler divergence can be seen on eq. 3.5, where $C^{(i)} = \sum_{j=1}^{N_c} e^{C_j^{(i)}}$ is a constant and $\mathbf{z}^{(i)} = \left[z_1^{(i)} ... z_{N_c}^{(i)} \right]$ is the i-th component of the output of the encoder.

$$KL_{discrete} = \sum_{i=1}^{N_d} \sum_{j=1}^{N_c} \frac{e^{z_j^{(i)}}}{C^{(i)}} \left(\log \frac{e^{z_j^{(i)}}}{C^{(i)}} - \log \frac{1}{N_c} \right) \tag{3.5}$$

3.3.6 Training a deeper network

The last of the experiments performed consisted in training a deeper network, with one more layer on the encoder and the decoder. This approach was initially thought to overcome the problems with the embedded binarization.

With this purpose, a ReLu was added after the second layer in the encoder and, after that, a third layer, with a variable size as the second layer. In Fig. 3.10, the encoder structure can be seen, whilst the decoder's one is Fig. 3.11.

The rest of the variational autoencoder structure remained unchanged, being the mean and the variance of the latent space calculated after the third layer instead than after the second.

Due to time constraints, instead of trying this network for all the different methods proposed before, it was only tested for the best one as well as for the embedded binarization.

3.4 Evaluation Criterion

In order to test the performance of the method propose, an evaluation criterion is required. To have a first insight into the accuracy of the different methods, the first criterion employed consisted in calculating the mean of correct detection of the K-top images. However, this criterion doses not reflect correctly the efficiency of the method. To solve this problem, finally the MAP metric was chosen to be the evaluation criterion of this thesis. A deeper insight on this criterion would be exposed in Chapter 4.

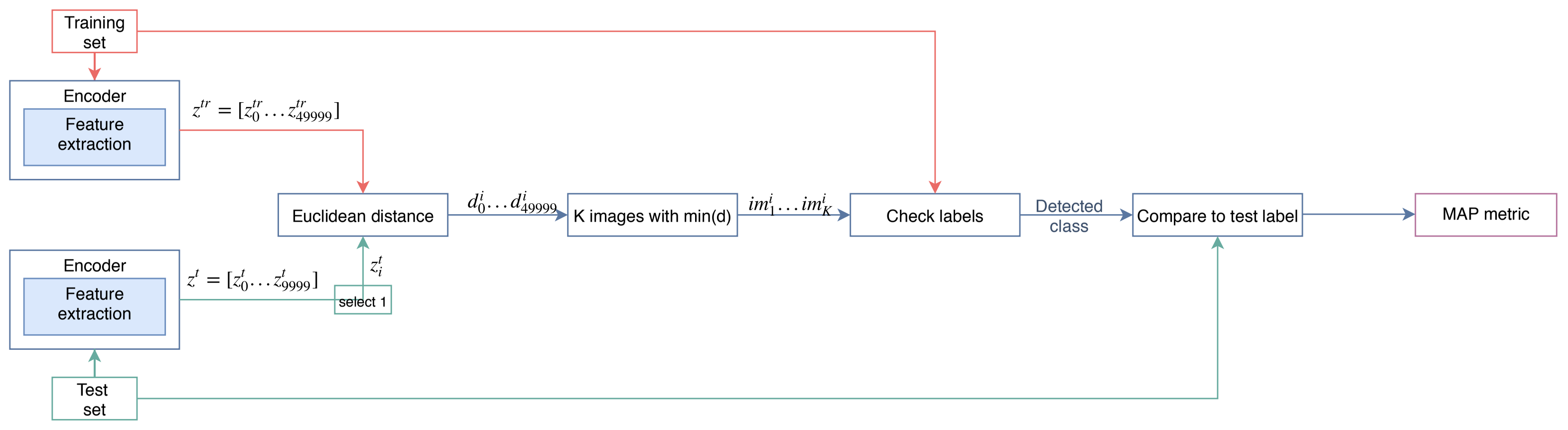

Figure 3.6: Diagram of the process employed for the image detection

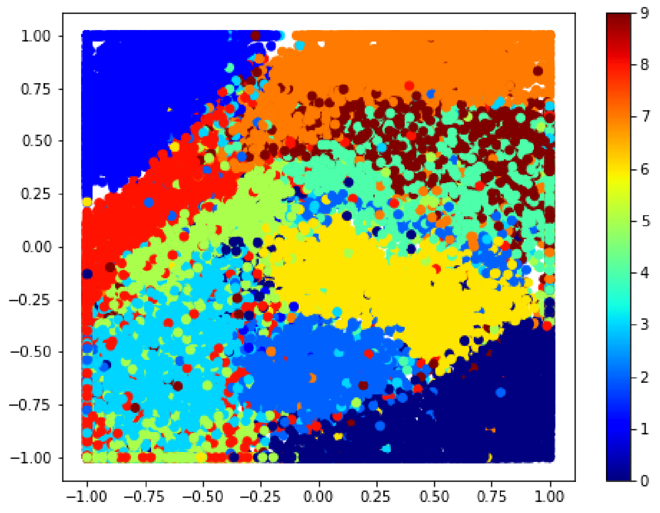

Figure 3.7: Latent space for the reduced latent space

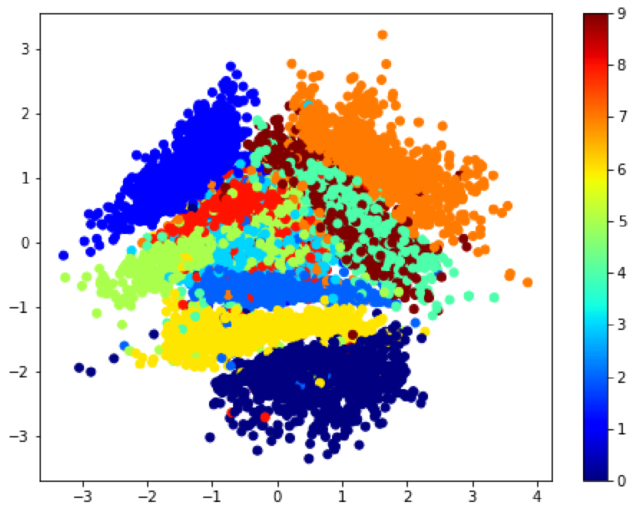

Figure 3.8: 2D latent space for the approximated method

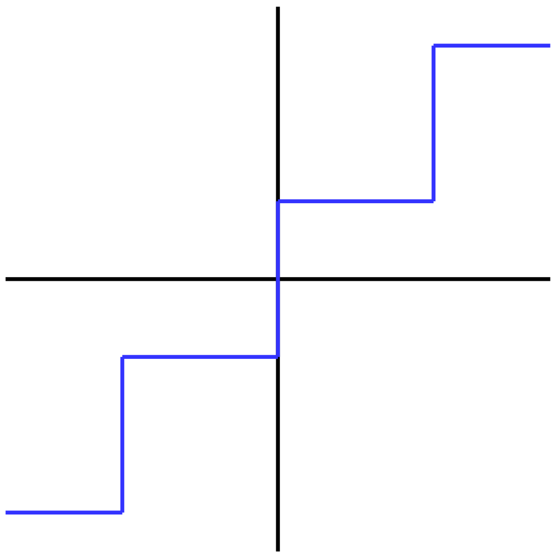

Figure 3.9: Example of a linear uniform quantizer

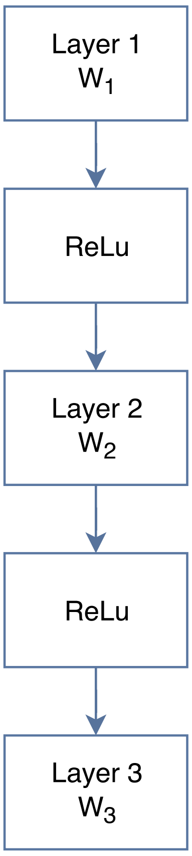

Figure 3.10: Structure of the encoder with 3 layers

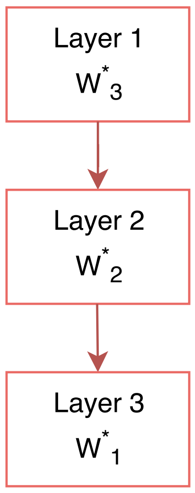

Figure 3.11: Structure of the decoder with 3 layers

Chapter 4

Experimental Results

Along the development of this thesis, many experiments were performed to evaluate the parameters of the variational autoencoder implemented. The obtained results are presented in this chapter, following this structure: firstly, the experiment settings employed, such as the database and the evaluation criterion, are stated. Secondly, the different parameters of the VAE are briefly presented and finally, the performances of all the experiments are presented and compared.

4.1 Experiment Settings

In this section the database and the evaluation criterion are briefly explained in order to describe the settings utilized for the development of the experiments.

4.1.1 MNIST Database

The chosen database for the development of this thesis was the MNIST database [28]. It consists of a set of hand-written digits, from 0 to 9 and is widely used for training and testing machine learning techniques and pattern recognition methods, as well as for image processing. It is composed by a training set of 60000 examples and a test set of 10000 images, all of them labeled to ease the goodness of the tested method.

The images of the digits are in grey-scale, with a size of 28x28 pixels, with values in the range [0-1]. As it can be seen on Fig. 4.1, the background of the image is represented by low intensity values (0) whilst the foreground (the digits) have high intensity values (around 1). The number of examples in each class in the training dataset goes from 5421 examples for number 5 to 6742 in class 1.

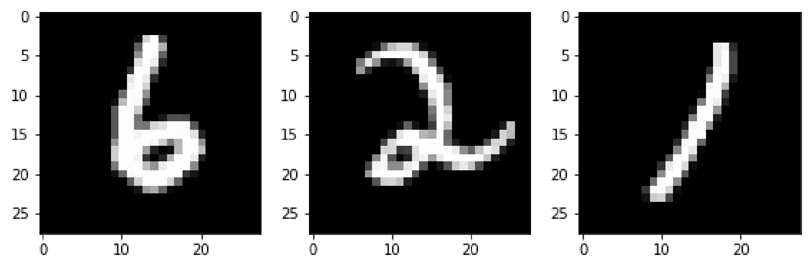

Figure 4.1: Example of MNIST database

4.1.2 Mean Average Precision Metric

The Mean Average Precision metric or MAP metric for short is a good way to evaluate the results obtained before. It consists in taking the mean for the APs of all the queries analyzed (Eq. 4.1).

$$MAP = \frac{\sum_{q=1}^{Q} AP_q}{Q} \tag{4.1}$$

Average Precision does not only consider how many correct detections the system has achieved, but besides, it penalizes the wrong ones, and the order in which they are. This means that, in the case of some incorrect detections in the K-nearest neighbours, it is preferred that these are not the closest to our input but the ones further from it.

To calculate the AP for a given input the first step would be determining whether the detections are correct or not. If they are incorrect they will contribute 0 to the calculation of the AP. If the answer is correct, then it will add one to the count of correct images, but dividing this sum between the number of images. After having evaluated all the answers, the AP is calculated as the sum of all them between the number of correct answers.

An easy example would be having as input two images from the test set with label 5. The 5-closest neighbors for the first one are images from the training set with labels 5, 3, 5, 3, 8, and for the second 3, 3, 5, 5, 5. At fist sight it could be said that the second one has a better detection since 3 of the labels are correct (60% accuracy) whereas for the second only 2 were correct (40% accuracy). However, calculating the APs for both Image 1 and Image 2 gives the following results:

$$AP_1 = \frac{1/1 + 0 + 2/3 + 0 + 0}{2} = \frac{5}{6} = 0.8\overline{3} \tag{4.2}$$

$$AP_2 = \frac{0 + 0 + 1/3 + 2/4 + 3/5}{3} = \frac{43}{90} = 0.4\overline{7} \tag{4.3}$$

This happens because Average Precision penalizes that the two first labels for the second image were incorrect, showing that the most similar images for this one are incorrect.

Size	**MAP metric**
n2 = 100	0.8896
n2 = 500	0.85764
n2 = 784	0.8468
n2 = 1000	0.84918
n2 = 10000	0.84912

Table 4.1: MAP metrics for different size of the second layer of the encoder

4.2 Parameters of the Variational Autoencoder

The variational autoencoder implemented in Chapter 3 using TensorFlow had some non-fixed parameters such as the length of the vector z, which represents the latent space, or the number of epochs. In this section, all the parameters used along the development of this thesis are be presented.

Firstly, the training is done using mini-batches of size 100, along 75 epochs to avoid an overfitting to the data. Both the encoder and the decoder are implemented using Gaussian distributions, and the first layer has a fixed size of 500. In the encoder a ReLu activation function [27] is located between the two layers featured. This activation function is simpler and faster compared to the sigmoid and tanh functions due to its linearity. On the other hand, the pooling selected is max pooling, which convolves the input with multiple filters, selecting the highest response of all them as the representative for each path, as in Fig. 2.2.

In order to select the size of the second layer of the encoder an experiment was performed varying it. In this experiment, the top K=5 closest images from the training set compared with the test one were selected. It considered both sizes smaller and larger than 784 (the dimension of the input), as well as that one with the objective of determining the size that yields the best results. Furthermore, it allows to determine how affects the size of this layer to the overall performance. The obtained results can be seen in Table 4.1, where it makes clear that the best performance is obtained whit the smaller size for this layer. For this reason, the chosen value for this parameter through all the simulations studied will be 100.

Experiment	MAP metric
Default	0.95005
Binary	0.65445
Embedded binarization	0.14465
Range [-1, 1]	0.9577
μ_z	0.96435
μ_z and range [-1, 1]	0.96355

Table 4.2: MAP metrics for the different experiments performed, using K=2

4.3 Performance of the Experiments

As it was introduced at the beginning of the chapter, the results of the different experiments are presented in this section. In Table 4.2 there is an overview of all these results, with its MAP metric, the evaluation criterion considered along this thesis. Through this section, all the studied experiments will be explained and compared more deeply.

4.3.1 Default Experiment

The first experiment tested, or default experiment, was initially designed as a first indicator of how good a variational autoencoder could be for image retrieval purposes. Its main characteristics are that it employs a latent space with N=20 dimensions, in the previously described range of [-4, 4], without taking any approximation.

As it was explained in 3.3.1, the algorithm selects the top K-images within the whole training set, meaning the K images with the smaller Euclidean distance to the test image. Therefore, the first step of this experiment should be determining the which value of K would suit better for the purpose of image retrieval.

For that purpose, different values were considered, both small and big, and the MAP results obtained for them were compared. That comparison can be seen in Table 4.3, showing that all the considered cases yielded more than 0.7 MAP precision, which may be considered as very promising outcomes. However, the peak result along the values considered is obtained when K=2, so all the following experiments will be studied with this value.

However, it must be noted that this is the parameter that yielded better results in this framework, and it could be change depending on the experiments or the methodology.

Different Latent Representations

The next step consisted in varying the size N of the latent space z for achieving smaller and bigger representations and checking its performance for the image retrieval problem. The only constraint referring to the size of z is that it must always be smaller than the size of the input image, i.e., 784. In the contrary, if N was equal or bigger than 784, the variational autoencoder would learn the

K	**MAP metric**
K = 1	0.8594
K = 2	0.9006
K = 5	0.8896
K = 10	0.8637
K = 20	0.82609
K = 50	0.78502
K = 100	0.74512

Table 4.3: MAP metrics for different K-top images

N	**MAP metric**
$N = 2$	0.7729
$N = 5$	0.9356
$N = 10$	0.95005
$N = 20$	0.9006
$N = 50$	0.65865
$N = 100$	0.43885
$N = 500$	0.23805

Table 4.4: MAP metrics for different dimensions of the latent space, for K=2

identity function in the training, so the output would always be identical to the input.

The different values for N can be seen on Table 4.4, as well as the MAP obtained with each one of them. It looks clear that the best number of dimensions for z is 10, since it yields 0.95005, quite better than the following one, N=5, that only obtained 0.9356. Therefore, the rest of the experiments will be done by having hidden spaces of 10 dimensions, and will be compared to this result. One explanation for N=10 being the better option of all the tested could be that there are exactly 10 possible classes to be classified. However, further experiments should be performed in order to assure this.

Another interesting conclusion can be extracted from this experiment, consisting of the drop in the performance when the size of z approaches the size of n2 = 100. This decrease is specially noted on the last dimension tested, N = 500 and it is caused by expanding the space compared to the previous layer. In this case, some approximations are performed and therefore, the results obtained are worse than the ones obtained when $N < n2$.

An example of the image retrieval performed with the previous parameters for the VAE (n2=100 and N=10) can be seen on Fig. 4.2. The only difference between this figure and the final method employed is that it shows the K=5 closest images instead of only the K=2 determined previously in this section.

In order to test more the performance of the variational autoencoder designed, it was used to reconstruct some images, as it can be seen on Fig. 4.3. It makes clear that the reconstructions are pretty similar to the input images, proving that this variational autoencoder works fine.

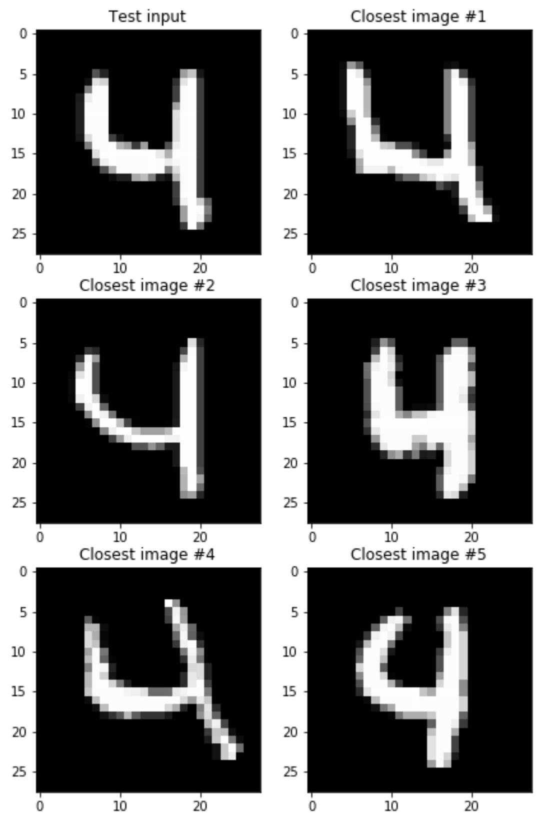

Figure 4.2: Example of image retrieval

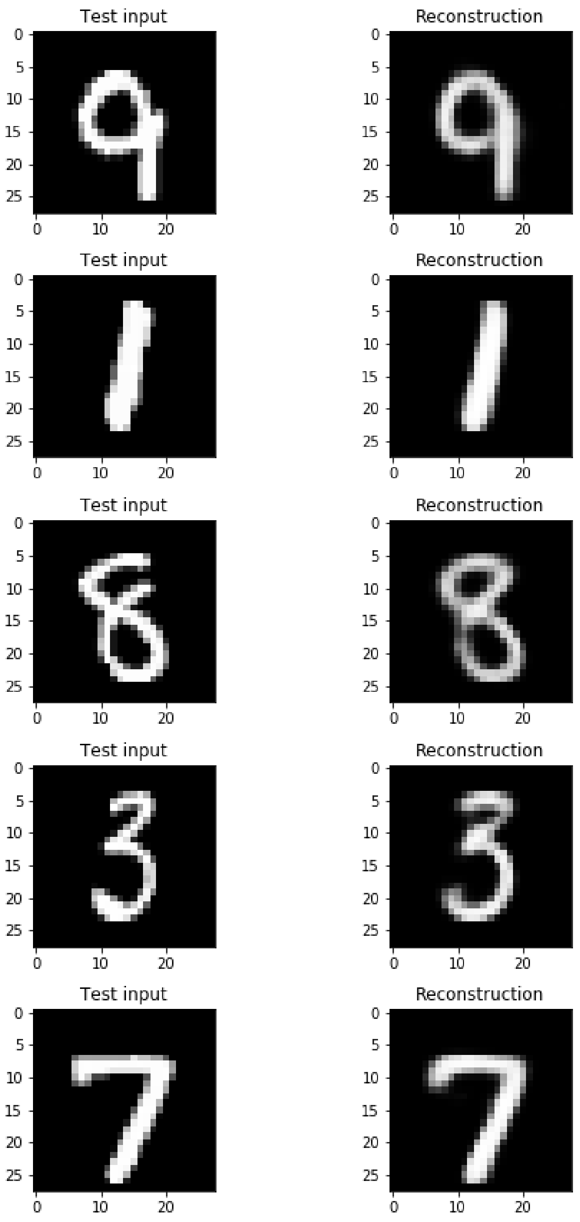

Figure 4.3: Reconstruction of some images

4.3.2 Binarizing the Latent Space

As it was mentioned in the previous chapter, binarizing the latent space could be an easier and faster solution for the image retrieval. Therefore, the second experiment considered consisted in binarizing the latent space z after the training, using a size of N=10 as it was determined in the previous experiment. However, despite the fact of having a faster retrieval, the accuracy of this method decrease considerably compared to the one achieved without the approximation: 0.65445 versus the previous 0.8896.

This drop in the MAP metric may be caused since the latent space is approximated, not the original one. This implies that many information is lost when transforming the latent space from a continuous space between the values [-4, 4] to a binary one with only the values -1 and 1.

Nevertheless, further experiments were performed concerning a binary latent space by means of embedding that binarization in the training, so the output latent space would be binary a priori. This experiments however, did not report a good performance at all, with a mere MAP of 0.14465. This result is explained by the vanishing gradient problem, as it was mentioned in the previous chapter. An alternative solution to this problem would be making the network deeper. Unfortunately, due to time constraints, this issue should be continued after this thesis is presented.

4.3.3 Reducing the Range of the Hidden Space

The following approach consisted in reducing the range of the hidden space to [-1, 1]. This could be considered as a kind of binarization since it constraints the values to that range. However, it shows better results compared to the proper binarization and even to the default method: a MAP of 0.9577.

This gain in the performance compared to the binary case can be explained since this approach reduces the approximation to the original latent space. However, it also shows that reducing the range of the latent space provides better results. This can be seen also as a normalization of the latent space.

4.3.4 Considering only the Centroids of the Representations

The next experiment was done considering only the centroids of the different representations in the latent space for the ten classes available. This could also be seen as taking $z = \mu_z$ instead of the prevoiusly defined $z = \mu_z + \sigma_z \epsilon$. This later approach reports the best result of all the examined ones: a MAP metric of 0.96435. The rise in this value compared to the previous one is caused since, when considering a distribution as only one point (its centroid), the different classes are more separated in the space. Therefore, the Euclidean distance between different classes rises, enhancing the retrieval accuracy.

Since this method proved to be better than the default, as well as the one exposed in Section 4.3.3, an experiment was done by combining both of them. The obtained result, however, decreased the accuracy to 0.96355 so it was discarted as the best alternative.

Number of levels	MAP metric
2 levels	0.65445
4 levels	0.7769
8 levels	0.9088
16 levels	0.95285
32 levels	0.96235
∞ levels	0.96435

Table 4.5: MAP metrics for different quantization levels, N=10, and $z = \mu_z$

4.3.5 Quantizing the latent space

As it was mentioned in Chapter 3, after the previous experiments were performed, it was proposed to perform a quantization of the latent space since the results for only binarizing it proved to be quite disappointing, as it was previously mentioned in this chapter. For this purpose, the quantization was performed over the results of considering the centroids of the representations, since that experiment was the one that yielded the best results.

The approach taken consisted in quantizing the hidden space z after the training, with 2, 4, 8, 16 and 32 levels. The results of this experiment, joint to the result without quantization which would correspond to quantize with ∞ levels, can be seen on Table 4.5.

It seems clear that the results improve along with the increase of the number of levels, since the approximation of the latent space is smaller, i.e., the approximated values are more similar to the original ones when increasing the number of levels. One interesting fact that can be observed is that the quantization of 16 levels yields better results than the default approach with the same parameters (N=10 and K=2), and whose map resulted on 0.95005 versus the 0.95285 obtained with the quantization.

Another highlight of this quantization would be that with 32 levels, the result is pretty similar to the one without quantization: 0.96235 versus 0.96435. This makes clear that the values could be quantized without supposing a great decrease in the performance of the experiments

$\mathbf{N_c}$	$\mathbf{N_d}$	**MAP metric**
10	10	0.95135
5	10	0.935
30	10	0.9453
10	5	0.92658
10	30	0.95248

Table 4.6: MAP metrics for configurations in the embedded discretization, with $z = \mu_z$

4.3.6 Discretization of the latent space

On the other hand, another method that was performed consisted in an embedded discretization, which was tried following the method described in 3.3.5. This method gives a latent space z already discrete as an output of the training process. This discrete latent space, as it was mentioned resulted on being labels for each one of the ten different classes from the MNIST database, i.e, one-hot vectors with a different position depending on the class.

For this approach, the temperature value suggested in [45] was taken, being it $\tau = 1$. Regarding the other parameters which can be varied among this method (the number of classes, N_c, and the number of categorical distributions, N_d), some different configurations were tested to see how do those parameters affect to the final performance.

The configurations tested were:

- As a first approach, $N_c = 10$ and $N_d = 10$
- Keeping $N_d = 10$ constant and varying N_c:
  - Reducing its value: $N_c = 5$
  - Increasing it: $N_c = 30$
- Keeping $N_c = 10$ constant and varying N_d:
  - Reducing its value: $N_d = 5$
  - Increasing it: $N_d = 30$

The results of these configurations can be seen on Table 4.6. The first conclusion that can be extracted from those results is that increasing and reducing the size of the number of classes, N_c, does not yield any improvement on the performance of the retrieval. By taking all the values with $N_d = 10$, it can be seen that the best value for N_c is 10. Therefore, it was kept for the following experiments regarding the value of N_d.

On the other hand, once N_c was kept constant with a value of 10, it can be clearly seen that, by increasing the number of categorical distributions, the performance grows as well. However, it must be noted that the processing time increases as well with N_d, so an extremely big value for this parameter would result in a pretty slow method. Finally, from all the tested values, the one which achieved the better performance would consist of the one with $N_c = 10$ and $N_d = 30$, with a MAP of 0.95248.

Size	MAP metric
n3 = 200	0.9576
n3 = 100	0.96455
n3 = 50	0.96475

Table 4.7: MAP metrics for different size of the third layer of the encoder, for $z = \mu_z$

The best result ($N_c = 10$ and $N_d = 30$) proved to be still slightly better than the one obtained with the default approach (0.95248 versus 0.95005), but they were still worse than the ones obtained without having a categorical latent space (0.96435). This is due to the fact that, although this method has been proven to be better than many others in the literature for quantizing the latent space [47, 48, 49], it still does not yield as good results as the ones obtained with a continuous z. However, further experiments could be done changing the parameters previously commented.

4.3.7 Training a deeper network

As it was mentioned in Chapter 3, the last experiment performed regarded changing the structure of the variational autoencoder in order to achieve better results, specially on the embedded binarization. However, due to time constraints and that the new network required more time for the training, not many experiments were done with this new structure.

The first experiment with this network was performed in order to decide the size of the third layer, n3. For this, three possible values were tested: 200, 100 and 50. The size of the first layer remained constant with a value of 500, as in all the previous experiments. On the other hand, whilst the size of the second layer, n2, could be varied, it was determined to let it fixed at 100 in order to reduce the number of possible experiments. However, it could be varied too in order to achieve the best combination which results in the higher MAP for the retrieval.

In Table 4.7, the results of the experiments regarding the size of this third layer are stated. It must be clarified also that these values were calculated for the best method of all the previous ones: the method that considers only the centroids of the latent space.

As it happened with the size of the second layer, the performance gets increased when reducing the size n3 of the third layer of the encoder and the decoder. However, it can be seen that, when the size of the third layer is bigger than the size of the second one (n3 = 200 > n2 = 100), the performance drops to an even smaller value than the one achieved with only two layers (0.9576 vs 0.96355).

Another significant conclusion regarding the size of n3, is that there is almost not significant difference between having n3 = 100 and n3 = 50. Nevertheless, as the latter is better, this was the chosen value for the following experiment which considered the embedded binarization.

Method	MAP metric
2 layers, Default experiment	0.14465
2 layers and $z = \mu_z$	0.14755
3 layers and $z = \mu_z$	0.16235

Table 4.8: MAP metrics for different embedded binarization configurations

When comparing the result for n3 = 50, 0.96475, with the one achieved with only two layers, 0.96355, it is clear that the difference is quite small. Therefore, since this difference in the accuracy is insignificant and the computation time increases noticeably, this new architecture has not been proven to be quite effective. Yet, since this deeper network was inspired by a need for increasing the accuracy in the embedded binarization, it must to be checked with that method.

Embedded binarization with the 3 layers network

Once the size of the last layer was decided (n3 = 50), the embedded binarization was tried again. For this purpose, since previously this method was performed over the default one, the first step was to determine the performance of the embedded binarization when considering the centroids of the distribution, since this method was the one with the best performance.

All the tested configurations for the embedded binarization can be seen on Table 4.8. The second configuration, an encoder with 2 layers approximating z as its mean, shows that there is a slightly improvement in the result (0.14755 vs. 0.14465 achieved with the default method), as well as happened previously in the case without binarization.

Regarding the new VAE with a third layer, which obtained a MAP of 0.16235, the results shown that it slightly outperforms the result comparing it to the original VAE which only featured 2 layers, that had a MAP of 0.14755. This little improvement shows that making the network deeper actually helps the retrieval to be more accurate but with only 3 layers it is not enough.

To conclude, although it seems to be a slight improvement in the results, specially regarding the embedded binarization, the results were not very encouraging. This method still does not perform good enough to compare to the other ones considered along this chapter. Furthermore, by adding a third layer the training time grew quite much, so, adding this to the results obtained, the conclusion is that a network with 3 layers is not necessary.

However, by trying other configurations with the sizes of the second and the third layer the result may improve a little more. Moreover, an even deeper network may achieve better performances, as it was seen that the result was better than the one obtained with only two layers.

4.4 Discussion

Through this chapter the performance of all the different methods employed for the image retrieval have been commented and compared, as well as the best parameters for the variational autoencoder designed and implemented.

As it was stated previously in this chapter, a few parameters of the network were changed in order to achieve the best performance. This parameters were the size of the second layer of the encoder and the decoder ($n2$), which resulted on being 100. Once this size was determined, the size of the hidden layer was tested too, providing that the best alternative was $N = 10$. This size, as it was mentioned before, may be caused by having the same number of classes and therefore, classifying each one of them in one of the 10 positions available. However, further experiments should be done before affirming this.

Once the network parameters were fixed, the settings for the image retrieval were varied too, finding out that K=2 provided the best results.

Then, many different experiments were tested, and the one which resulted in having the higher accuracy was the one considering only the centroids of each one of the different classes. After this, a quantization experiment was performed. The results of that experiment proved that a quantization with a relatively high number of levels (32) did not suppose a great loss in the accuracy of this method. When embedding the discretization in the training, the result still did not suffer from a high loss in the accuracy when comparing it to the better one.

In order to finally test the accuracy of this method, it would have been interesting to implement some different algorithms proposed in the literature (siamese networks, GANs...). With those methods implemented, a comparison of the accuracy would have been possible but, due to time constraints, this comparison is one of the future work possibilities after this thesis.

Chapter 5

Conclusions and Future Work

In this last chapter, a brief summary of all the work done through the development of this thesis is presented. This chapter also features some possible ways to continue with the objective of the thesis.

5.1 Conclusions

Through the development of this thesis we have proposed a new method to perform image retrieval by means of using a variational autoencoder.

The variational autoencoder implemented was a quite simple structure, with only a 2 layers depth in both the encoder and the decoder. All the parameters of this VAE were trained achieve the best performance regarding the applications of this work.

The obtained models were trained and tested on the training and test sets of the database MNIST. The training provided a latent space able to capture the differentiating characteristics of the ten classes featured in this database and, thus, able to reconstruct correctly the input with the decoder.

However, the variational autoencoder implemented was finally employed to perform image retrieval. For this, a method was proposed, consisting in selecting the K-closest images in the training set for each one of the test images. Then, the MAP metrics of this retrieval was calculated to determine the performance of the method.

After that, many different variations in the method were tested with the objective of simplifying the procedure and achieving better results. Within all this methods, one outperformed the rest: the approximation of the latent space to $z = \mu_z$.

To conclude, it has been proved that this method employs a simple variational autoencoder which yielded pretty good results. With a more sophisticated network, it could be tested to be a simpler alternative to the state-of-the-art methods employed for image retrieval, such as Generative Adversarial Networks.

5.2 Future Work

Although the results of this thesis seemed to have a good performance, it can always be outperformed. In this section there are a few ways for improving the designed variational autoencoder.

- Firstly, it would be interesting to test the presented method with different databases. The database chosen stands out by its simplicity: only 10 different classes of handwritten digits. Therefore, more complex databases would be needed to determine the performance of the method presented in more realistic environments.
- As it was mentioned in Section 4.3.2, in order to binarize correctly the hidden space during the training it would be interesting to use an even deeper network. Thus, the encoder may be able to differentiate between the classes using only a binary space. In this case, a simpler and faster solution may be achieved.
- More experiments regarding the categorical reparametrization: change the values of the temperature, etc.
- Perform more experiments with the variational autoencoder that features three layers at its encoder and decoder.
- Compare the proposed method with different ones from the literature.
- Finally, many other methods could also be applied for achieving a better image retrieval. For instance, instead of using the Euclidean distance for the continuous space, a different one can be implemented.

Bibliography

[1] A. Creswell and A. A. Bharath, "Adversarial Training For Sketch Retrieval". In: *arXiv preprint arXiv:1607.02748*, 2016.

[2] J. Song, "Binary Generative Adversarial Networks for Image Retrieval". In: *arXiv preprint arXiv: 1708.04150*, 2017.

[3] J. Guo, S. Zhang and J. Li, "Hash Learning with Convolutional Neural Networks for Semantic Based Image Retrieval". In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp. 227–238, 2016.

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Neural Information Processing Systems*, 2012

[5] A. Babenko, A. Slesarev, A. Chigorin and V. Lempitsky, "Neural Codes for Image Retrieval". In: *arXiv preprint arXiv:1404.1777*, 2014.

[6] A. Babenko and V. Lempitsky, "Aggregating Deep Convolutional Features for Image Retrieval". In: *arXiv preprint arXiv:1510.07493*, 2015.

[7] A. S. Razavian, J. Sullivan, S. Carlsson and A. Maki, "Visual Instance Retrieval with Deep Convolutional Networks". In: *arXiv preprint arXiv:1412.6574*, 2014.

[8] A. Gordo, J. Almazan, J. Revaud and D. Larlus, "End-to-end Learning of Deep Visual Representations for Image Retrieval". In: *arXiv preprint arXiv:1610.07940*, 2016.

[9] E.-J. Ong, S. Husain and M. Bober, "Siamese Network of Deep Fisher-Vector Descriptors for Image Retrieval". In: *arXiv preprint arXiv:1702.00338*, 2017.

[10] F Radenovic, G. Tolias and O. Chum, "Fine-tuning CNN Image Retrieval with No Human Annotation". In: *arXiv preprint arXiv:1711.02512*, 2017.

[11] J.Ã. da Silva Júnior, R. E. Marçal and M. A. Batista, "Image Retrieval: Importance and Applications", *X Worshop de Visão Computacional*, pp. 311-315, 2014.

[12] M. Yasmin, S. Mohsin and M. Sharif, "Intelligent image retrieval techniques: A survey", *Journal of Applied Research and Technology*, vol. 12, pp. 87-103, February 2014.

[13] Y. Liu, D. Zhang, G. Lu and W. Ma, "A survey of content-based image retrieval with high-level semantics", *Pattern Recognition*, vol. 40, pp. 262-282, 2007.

[14] J. Eakins and M. Graham, "Content-based image retrieval", *Technical Report*, University of Northumbria at Newcastle, 1999.

[15] A. K. Jain, J. Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial", *Computer*, vol. 29, Isuue. 3, pp. 31-44, March 1996.

[16] W. S. McCulloch and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[17] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[18] Y. Bengio, A. Courville and P. Vincent, "Representation learning: A review and new perspectives", *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798-1828, 2013

[19] K. Gregor, I. Danihelka, A. Mnih, C. Blundell and D. Wiersta, "Deep AutoRegressive Networks". In: *arXiv preprint arXiv: 1310.8499*, 2013.

[20] Q. Xu and L. Zhang, "The effect of different hidden unit number of sparse autoencoder", *The 27th Chinese Controdrl and Decision Conference (CCDC)*, pp. 2464-2467, 2015.

[21] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.

[22] A. Makhzani and B. Frey, "k-sparse autoencoder". In: *arXiv preprint arXiv: 1312.5663*, 2013.

[23] S. Rifai, P. Vincent, X. Muller, X. Glorot and Y. Bengio, "Contractive Autoencoders: Explicit Invariance During Feature Extraction", *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011.

[24] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv: 1603.04467*, 2016.

[25] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", *Proceedings of Machine Learning Research*, vol. 9, pp. 249-256, 2010.

[26] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv: 1412.6980*, 2014.

[27] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit", *Nature*, vol. 405, pp. 947-951, 2000.

[28] Y. LeCun et al., "The MNIST Dataset Of Handwritten Digits (Images), 1999. In: http://yann.lecun.com/exdb/mnist/

[29] D. Kingma and M. Welling, "Auto-Encoding Variational Bayes", *Proceedings of the International Conference on Learning Representations*, 2014, in: *arXiv preprint arXiv:1312.6114*.

[30] D. Kingma, T. Salimans and M. Welling, “Improving variational inference with inverse autoregressive flow”, 2016. In: *arXiv preprint arXiv:1606.04934*.

[31] C. Doersch, “Tutorial on variational autoencoders”. In: *arXiv preprint arXiv: 1606.05908*, 2016.

[32] Y. Burda, R. Groose and R. Salakhutdinov, “Importance Weighted Autoencoders”. In: *arXiv preprint arXiv: 1509.00519*, 2015.

[33] Q. Xu, Z. Wu, Y. Yang and L. Zhang, “The Difference Learning of Hidden Layer between Autoencoder and Variational Autoencoder”, *The 29th Chinese Control and Decision Conference (CCDC)*, pp. 4801-4804, 2017.

[34] T. Salimans, D. Kingma and M. Welling, “Markov hain Monte Carlo and variational inference: Bridging the gap”, *ICML*, 2015

[35] T. D. Kulkarni, W. F. Whitney, P. Kohli and J. Tenenbaum, “Deep convolutional inverse graphhics network”, *NIPS*, 2015.

[36] D. P. Kingma, S. Mohamed, D. Jiménez Rezende and M. Welling, “Semi-supervised learning with deep generative models”, *NIPS*, 2014.

[37] K. Gregor, I. Danihelka, A. Graves, D. Rezende and D. Wierstra, “Draw: A recurrent neural network for image generation”, *ICCV*, 2015.

[38] E. J. Gummbel, “Statistical theory of extreme values and some practical applications: a series of lectures”. Number 33. *US Govt. Print. Office*, 1954.

[39] R. D. Luce, “Individual Choice Behavior: A Theoretical Analysis”. New York: *Wiley*, 1959.

[40] J. I. Yellott, “The relationship between Luce’s choice axiom, Thurstone’s theory of comparative judgment, and the double exponential distribution”, *Journal of Mathematical Psychology*, vol. 15, pp. 109-144, 1977.

[41] G. Papandreou and A. L. Yuille, “Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models”, *ICCV*, 2011.

[42] T. Hazan and T. Jaakkola, “ On the partition function and random maximum a-posteriori perturbations”, *ICML*, 2012.

[43] C. J. Maddison, D. Tarlow and T. Minka. “A* sampling”, *Advances in Neural Information Processing Systems*, pp. 3086-3094, 2014.

[44] T. Hazan, G. Papandreou and D. Tarlow *Perturbation, Optimization and Statistics*, Chapter 7, MIT Press, 2016.

[45] E. Jang, S. Gu and B. Poole, “Categorical Reparametrization with Gumbel-Softmax”, 2016. In: *arXiv preprint arXiv:1611.01144*.

[46] C. Maddison, A. Mnih and Y. Teh, “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”, 2016. In *arXiv preprint arXiv:1611.00712*

[47] Y. Bengio, N. Leonard and A. Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation", 2013. In *arXiv preprint arXiv:1308.3432*.

[48] A. Mnih and D. J. Rezende. "Variational inference for monte carlo objectives", 2016. In *arXiv preprint arXiv:1602.06725*.

[49] D. J. Rezende, S. Mohamed and D. Wierstra. "Stochastic backpropagation and approximate inference in deep generative models", 2014. In *arXiv preprint arXiv:1401.4082*.

TRITA TRITA-EECS-EX-2018:454
ISSN 1653-5146