# Clustering with Missing Values: a Deep Learning Approach

**Rita Tomás Caveirinha**

Thesis to obtain the Master of Science Degree in

**Information Systems and Computer Engineering**

Supervisors: Prof. Pedro Filipe Zeferino Tomás
Dr. Helena Isabel Aidos Lopes

**Examination Committee**

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Pedro Filipe Zeferino Tomás
Member of the Committee: Prof. Nuno Cruz Garcia

**May 2022**

# Acknowledgments

A todos os que me acompanharam nesta etapa.

Aos meus orientadores, por toda a disponibilidade, tempo e apoio em todas as fases desta tese. Foram sempre super incriveis e tive a maior sorte do mundo em tê-los ao meu lado.

Às instituições que me acolheram, ao ISCTE-IUL e ao Instituto Superior Técnico, e a todos os professores que me inspiraram ao longo destes anos.

À minha família, Mãe, Zé e avó, porque sem vocês nunca teria chegado aqui. À minha tia Lena, que eu adoro de coração. Ao meu avô Dimas, que me inspira todos os dias.

A todos os meus amigos, ao Tiago de quem eu tenho um orgulho enorme, ao Jorge e as minhas Marias, ao David que está longe em terras inglesas e que tenho saudades, à Ana Sofia, à Jeca, ao João, à Patrícia, ao Fred, e ao Tomás. Vão estar sempre no meu coração.

Um gigante à Carlota e à Inês, as irmãs que nunca tive e que sempre estiveram presentes em todos os sorrisos e lágrimas destes anos caóticos, sem vocês a vida não era a mesma.

A todos aqueles que estiveram comigo desde a licenciatura até aqui, a todas as companhias em salas de estudo e bibliotecas, e em festas também.

À Bia por todas as companhias em maratonas de estudo e por todas as vezes em que me ofereceu uma tablete de chocolate quando me viu triste.

À Catarina, pois nem a pandemia me impediu de fazer amizades para a vida, e agora tenho um bocadinho do meu coração em Coimbra.

E à Ana, que desde o primeiro ano sempre esteve ao meu lado em todos os trabalhos, viagens de comboio e passeios por Lisboa.

A cada um de vós, um grande obrigada. Sem vocês não teria chegado aqui.

# Abstract

This work focuses on the usage of deep clustering models, whose research has been growing in recent years due to their usefulness of its usage across many fields. The goal of this thesis was to work around the problem of clustering missing values using deep clustering, which is a fundamental problem to tackle since in real-world data, such as the profiling of patients by their progression patterns in neuro-degenerative diseases, where missing values often occur.

Yet, after exploring the state-of-the-art techniques it was concluded that the existence of missing data is an obstacle to the robustness of the best clustering methodologies since they are often developed and tested with clean data. In a similar observation, it was noticeable that the deep learning architectures with the ability to work with missing data, failed to solve the clustering part of the problem. With that in mind, this work explored the topic of clustering with missing data to find the most optimal solution, going through different base architectures that could potentially handle missing data, and the most important, introducing a variational autoencoder variation through the use of a binary mask to better handle missing data. With the usage of VAE-based architectures, this work performs clustering in its latent space by using the dimension reduction functionalities from UMAP, followed by automatic cluster detection provided by HDBSCAN. The main contributions of this work are: 1) Offer a novel approach through deep clustering with missing data, 2) A simple but effective solution for better robustness when dealing with missing data, offering similar results to complete data, 3) Automatic cluster detection, 4) Data dependant dynamic architectures.

# Keywords

Deep learning, Clustering, Missing Data, Variational Autoencoders, UMAP, HDBSCAN, Generative Models, Deep image clustering.

# Resumo

O foco desta dissertação é o uso de modelos de deep clustering, cuja pesquisa tem vindo a crescer nos últimos anos devido à versatilidade do seu uso em vários campos da ciência. O objetivo principal desta tese foi trabalhar em torno do problema de clustering de dados em falta usando deep clustering, que é um problema fundamental a ser abordado em dados do mundo real, como no caso do estudo dcd padrões de progressão em doenças neurodegenerativas, onde valores ausentes ocorrem frequentemente. No entanto, após explorar as técnicas do estado da arte, concluiu-se que a existência de dados em falta é um obstáculo à robustez de algumas das melhores metodologias existentes de clustering, uma vez que são muitas vezes desenvolvidas e testadas com dados limpos. Numa observação semelhante, foi perceptível que as arquiteturas de aprendizagem profunda com a capacidade de trabalhar com dados ausentes, focam-se mais tipicamente em tarefas de classificação. Com isto em mente, este trabalho explorou o tópico de clustering com dados ausentes para encontrar a melhor solução possivel, explorando diferentes arquiteturas de base que potencialmente poderiam lidar com dados ausentes, e a maior contribuição, a introdução de uma variante de autoencoder variacional através do uso de um máscara binária utilizada na função de custo para lidar melhor com dados ausentes. Com esse uso de arquiteturas baseadas em VAE, este trabalho realiza clustering no espaço latente utilizando as funcionalidades de redução de dimensão do algoritmo UMAP, seguida da detecção automática de clusters fornecida pelo HDBSCAN.

As principais contribuições deste trabalho são: 1) Oferecer uma nova abordagem de deep clustering capaz de lidar com dados ausentes, 2) Uma solução simples, mas eficaz para uma maior robustez ao lidar com dados ausentes, oferecendo resultados semelhantes aos dados completos, 3) Detecção automática de clusters, 4) Arquiteturas dinâmicas dependentes de dados.

# Palavras Chave

Aprendizagem profunda, Agrupamento, Não-resposta, Dados Ausentes, Variational Autoencoders, UMAP, HDBSCAN, Modelos Generativos, Agrupamento neuronal de imagens.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **CatGAN** | Categorial Generative Adversarial Network |
| **DAC** | Deep adaptive image clustering |
| **DAMIC** | Deep Autoencoder Mixture Clustering |
| **DBC** | Discriminatively Boosted Image Clustering |
| **DCN** | Deep Continuous Clustering |
| **DEC** | Deep Embedding Clustering |
| **DEPICT** | Deep Embedded Regularized Clustering |
| **GAN** | Generative Adversarial Network |
| **GMM** | Gaussian Mixture Model |
| **GMVAE** | Gaussian Mixture Variational Autoencoder |
| **HDBSCAN** | Hierarchical Density-Based Spatial Clustering of Applications with Noise |
| **InfoGAN** | Information Maximizing Generative Adversarial Network |
| **KL** | Kullback–Leibler |
| **MoE** | Mixture of Experts |
| **MSE** | Mean Squared Error |
| **N2D** | Not Too Deep Clustering |
| **PCA** | Principal Component Analysis |
| **SSIM** | Structural Similarity Index |
| **UMAP** | Uniform Manifold Approximation and Projection for Dimension Reduction |

| | |
|---|---|
| **VaDE** | Variational Deep Embedding |
| **VAE** | Variational Autoencoder |
| **WAE** | Wasserstein Autoencoder |

# 1 Introduction

Contents

## 1.1 Motivation

We live in an age of data. Throughout the recent years, there has been a remarkable growth in the global market for sensors, with some current estimations predicting this market to go from $205.2 billion in 2021 to $ 411.2 billion by 2026 [2], adding to other factors such as the decrease of data storage prices, and the emergence of cloud storage resulted in growth in the amount of data being dealt with across multiple spaces. With this growth, the need to organize and make sense of the meaning of the available information also arose, eventually leading to substantial advances in machine learning as well as in other emerging applications that make use of such techniques. Within machine learning, deep learning was fundamentally benefited, as the larger amounts of data and computational power allowed substantial improvements in this area.

At the same time, clustering methods have been improving, revealing to be very useful for the study and analysis of the progress across multiple fields. Examples include medicine, with the in the study of melanoma detection [3] [4], or even degenerative diseases such as Alzheimer's or Parkinson's. [5] [6] [7] Other cases can also include forest fire detection, [8], [9], or in the maintainance of automatic systems such as eolic turbines [10], [11].

However, the problem of missing values in the data occurs frequently across these fields, [12], [13], sensors and machines that provide exam results can have failures in sensors, or in some cases imaging data suffer from the occlusion (an artifact partially hides the image), or in cases where the data origin is survey-based, questions often get forgotten or the answers invalid. The most common ways to solve this problem rely on simple solutions such as the imputation of values.

Preprocessing of data is always a necessary step but the existence of missing values can create extra complexity to the problem since it implies a need for extra steps to deal with the problem.

Or in the case where the choice is to ignore the missingness, how much could this missingness affect the goal in mind. Previous work has been done with great results by approaching the problem of deep clustering with the usage of generative models. Yet, most of these advanced deep clustering architectures need complete datasets to achieve good results, which means that when missing data occurs the results are not as strong.

This lack of robustness to missing values was, therefore, the main motivation for the work done, in which it was proposed to explore the creation of a deep clustering model that is generative and robust to this problem.

## 1.2 Objectives

The main goal of this dissertation is to develop a clustering methodology that can directly tackle datasets with missing values without requiring the application of listwise deletion methods.

## 1.3 Contributions

The main contributions of this work are:

- **A novel approach through deep clustering with missing data**: current state-of-the-art techniques for deep clustering are mostly focused on complete data, while the deep learning works with datasets containing missing data focus mostly on other tasks, such as reconstructing and imputing the missing information and then finally apply the typical models on the reconstructed dataset.
- **A simple but effective solution for better robustness when dealing with missing data**, offering similar results to complete data: with the usage of a binary mask on the loss function.
- **Automatic cluster detection**: through the usage of Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) it is possible to discard the need for static definitions of the number of clusters, or parallel data exploration, by relying upon its features of finding the number of centroids automatically.
- **Data dependant dynamic architectures**: With the usage of a convolutional block architecture that is dependent on the type of data, the suggested solution offers the possibility of a flexible model that can easily be adapted and used with different data sizes.

## 1.4 Outline

This work starts by analyzing some of the literature on the mentioned topics, which can be seen in chapters 2 and 3. In chapter 2, the focus is on different types of deep learning architectures, and some clustering methods based on them, namely autoencoders, variational autoencoders, GANs, and Mixture of Experts. In chapter 3 the topic of missing data is explored, going through an overview of how missingness often occurs, the classic methodologies to handle missing values, recent deep learning methods that focus on handling and reconstructing data, and the clustering of missing data. Chapter 4 it is offered an overview of the main architecture achieved in this dissertation, as well as the ones explored. Chapter 5 provides an analysis and discussion of the results obtained and finally, chapter 6 presents the conclusions.

# 2

# Deep Image Clustering

## Contents

To contextualize the work done in this dissertation, this chapter introduces the concepts that serve as theoretical foundations for this work. Since the base problem being approached is deep clustering, this chapter starts by introducing the base deep learning architectures, followed by the most relevant clustering techniques and algorithms based on these architectures.

## 2.1 Autoencoder-based

### 2.1.1 Autoencoder

The first relevant concept is Autoencoders [14], which is one of the most significant algorithms in unsupervised representation learning, working by learning to efficiently compress data followed by its reconstruction. Thus, it is very useful for dimensionality reduction, which can substantially benefit the use of off-the-shelf clustering algorithms over the reduced space (as described later). The raw data is compressed into a smaller number of dimensions in the hidden layer and this allows the most salient features of data to be found. It is composed of two networks: an encoder, which is responsible for the conversion into a smaller and denser representation, and a decoder, which is responsible for the reconstruction of data back into the original input. For the learning of this model, distance metrics such as cross-entropy or mean squared error between the input data and the reconstructed image may be used.

$$L_{rec} = \frac{1}{N}||x_i - \tilde{x_i}||_2 \tag{2.1}$$

This model has a problem in instances where the data is not continuous such as when there are gaps between the clusters or when the goal is to sample an output that is not an exact copy of the input, the output will not return the ideal result, because it only knows how to replicate the same image that is inserted and will not know what to do in the case of gaps.

Since the introduction of autoencoders in the literature, a significant number of variants have emerged.

For example, with the focus on reducing overfitting and improving robustness by introducing a regularization term include the sparse autoencoder [14] and the contractive autoencoder [15]. The sparse autoencoder's key feature is that there is a greater number of hidden nodes than input nodes. It works with the usage of a sparsity penalty on the latent space, this penalty is a value close to zero but not zero and, where, this penalty allows for only a small number of units to be activated at the same time. The contractive autoencoder works by adding a term to the loss function which penalizes the model for being too sensitive, providing also better robustness to the model.

### 2.1.2 Autoencoder-based Clustering

For this category of clustering techniques, the focus is the dimensionality reduction nature of the autoencoder, which is often used in deep clustering algorithms. This type of approach often uses a pre-training scheme in which reconstruction loss is used to initialize parameters before applying/introducing clustering loss.

This category starts with **Deep Embedding Clustering (DEC)** [16], which is one of the baselines for the work that has been done on deep clustering. It is divided into two phases: In the first, an Autoencoder is trained by minimizing the reconstruction loss and initializing the parameters and after that, a cluster assignment hardening loss that is derived from K-means clustering is used and with that those parameters can be optimized. A variation of this method is **Discriminatively Boosted Image Clustering (DBC)** [17] where instead of a feedforward autoencoder, a Convolutional autoencoder is used. The training scheme, reconstruction loss, and cluster assignment hardening loss used are the same as in DEC. Naturally, this model shows good results in image datasets because of the use of CNN. **Deep Continuous Clustering (DCN)** [18] combines the autoencoder with the K-means algorithm. It starts by pre-training the autoencoder and then jointly optimizing the reconstruction loss and the K-means loss with alternating cluster assignments. When compared to other methods it has relatively low complexity and provides adequate results. Another method is **Deep Embedded Regularized Clustering (DEPICT)** [19], which adds some changes in the previously mentioned DEC and offers some improvements in the results. The first change is the usage of a convolutional layer which makes DEPICT a great choice for image clustering. Another key point that defines this architecture is the fact that it uses two encoders and one decoder, where one of the encoders is described as ”noisy” and the other ”clean”.

Another algorithm and one of the best in terms of performance and simplicity is **Not Too Deep Clustering (N2D)**. [20] An autoencoder is used here to learn low-level feature representations of the data, and the clustering is done by a shallow algorithm such as k-means or Gaussian Mixture Model (GMM). The key factor that provides the good results of this algorithm is Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [21], which is a manifold learning technique that is applied to the latent space of the autoencoder and improves the quality of the defined clusters. Hence, K-Means and Gaussian mixtures can then be applied, attaining good results.

## 2.2 Variational Autoencoder-based

### 2.2.1 Variational Autoencoder

Variational Autoencoder (VAE) [22], presented in figure 2.1, is a generative model that aims to solve the problem of traditional auto-encoders, which results in non-contiguous latent spaces. Instead, VAEs aim to learn a probability density distribution that represents the input images, thus ensuring contiguous latent spaces. By sampling points from the learned probability density function, it is possible to generate new data points that, although do not exist in the original dataset, seem realistic. Hence, it transforms the original idea into a generative model.

This idea, paired with the dimensional reduction also present in standard autoencoders and the future implementation of clustering mechanisms, provides what seems to be a good direction to achieve good results when clustering with missing data.

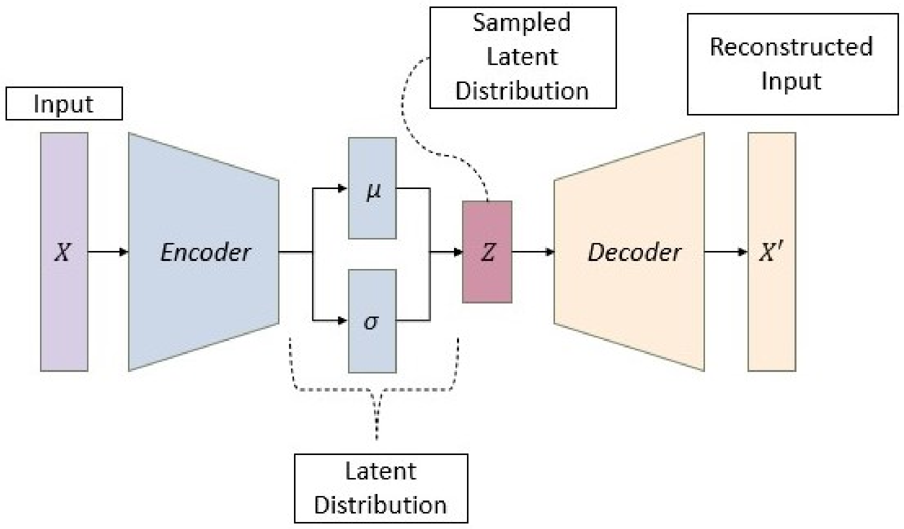


**Figure 2.1:** The architecture of a Variational Autoencoder. The encoding and decoding is a common factor with Autoencoders, yet the latent space distribution present in VAE allows for the sampling which makes a very different type of architecture.

Since the introduction of VAE's in the literature, some variations and extensions to the model have been created.

For a better comprehension of VAE's and these extensions, it is important to dive deeper into how VAE's work, especially the regularization of this model.

The loss function that is minimized when training a VAE's is composed of two terms: a ”reconstruction term” and a ”regularization term”.

$$L_{BETA}(\phi, \beta) = -\mathbb{E}_{Z \sim q_\phi(z|x)} \log p_\theta(x|z) + \beta D_{KL}(q_\phi(z|x)||p_\theta(z)) \tag{2.2}$$

The first term is the reconstruction term, where a negative log-likelihood is used concerning the encoder's distribution over the representations. This term encourages the decoder to learn to reconstruct the data. A poor reconstruction loss implies a large cost in the loss function.

In the second term, the regularization term, the organization of the latent space is regularised by making the distributions returned by the encoder close to a standard normal distribution. In this term, the Kullback–Leibler (KL) divergence is introduced, which measures the distance between two distributions and allows for an equilibrium between the cluster-forming nature of the reconstruction loss, and the dense packing nature of the KL loss. Another key component of the variational autoencoder is the parametrization trick, which introduces a stochastic node $\epsilon$ and allows for the randomness of the latent space $Z$ to be pushed into this variable, and with that backpropagation is possible.

One of the extensions made to a standard VAE is the $\beta$-VAE [23]. In this variation of the VAE the KL-divergence term of the loss function is multiplied by a hyperparameter $\beta$ which provides disentangled representations of the data [24]. Another interesting approach is the Wasserstein Autoencoder (WAE) [25]. In this variation, the regularization of the model is altered and instead of the KL divergence, the goal is to minimize the Wasserstein distance penalty, which is represented by $D_z(q_Z||p_Z)$ in the following expression.

$$L(Z) = \lambda * D_z(q_Z||p_Z) + L_{recon} \tag{2.3}$$

There are two possible variations for this penalty, GAN-based, which uses adversarial training in the latent space, and MMD-based, which uses the maximum mean discrepancy.

Since this architecture does not introduce random noise into the system and is more robust to the choice of hyperparameters in the network, WAEs have been shown to produce better quality samples compared to regular VAEs (i.e., clearer images). [25]

### 2.2.2 Clustering

Most of the autoencoder-based clustering algorithms mentioned in the section above can be adapted to receive a VAE model, but some VAE-specific can also be found in the literature.

The first is the **Variational Deep Embedding (VaDE)** [26], which works by imposing a GMM prior over the VAE. In the standard VAE version, the prior used to regularize the manifold is a one-dimensional Gaussian, with VaDE this is generalized into using a mixture of Gaussians prior instead.

The Gaussian Mixture Variational Autoencoder (GMVAE) [27] is a similar approach, which also imposes a mixture of Gaussians on the VAE latent space.

## 2.3 GAN-based

### 2.3.1 GANs

Another example of a generative model is Generative Adversarial Network (GAN) [28]. This architecture is composed of a system of two neural networks: a generator, $G$, that learns a data distribution and generates samples by sampling from the learned distribution, and a discriminator $D$ that learns to distinguish between a sample that came from the training data and a sample generated from G. The networks are simultaneously trained and compete against each other by engaging in a zero-sum game, where one agent's loss is the other agent's gain.

In a more formal manner, as explained by Goodfellow [14], a prior is defined on input noise variables $p_z(x)$is defined to learn the generators distribution $p_g$ over data x, which is then passed through a mapping function to data space through the generator function $G(z;\theta g)$. The differentiable function $G$ is represented by a neural network/multi-layered perceptron with parameters $\theta g$.,

A second neural network is defined, expressed as $D(x;\theta d)$ that outputs a scalar representing the probability of a data point coming from the actual distribution $p_x$ instead of $p_g$, i.e. the probability of a data point being real.

D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. At the same time, G is trained to fool D by minimizing $log(1 - D(G(z)))$ :

In other words, the goal is to play a minimax game between D and G which can be seen represented in the value function V (G,D):

$$min_G max_D L(D,G) = E_{x\ p_{data}(x)}[logD(x)] + E_{z\ p_z(z)}[log(1 - D(G(z))] \tag{2.4}$$

where the first term represents the maximization of D and the second term represents the minimization of G. This type of architecture, however, contains the disadvantages of being highly sensitive to hyperparameter selection, the chance of mode collapse, and other problems.

### 2.3.2 Clustering

For the category of GAN-based clustering some of the most relevant algorithms include, Information Maximizing Generative Adversarial Network (InfoGAN), Categorial Generative Adversarial Network (CatGAN) and Deep adaptive image clustering (DAC). [29]

DAC takes advantage of adversarial auto-encoder to perform clustering while using a generative approach through a tuned Gaussian mixture model.

The InfoGAN is an extension of the GAN architecture that can also be used for clustering. Its primary goal is to learn disentangled representations and allows for the control of properties of the generated images, while in the typical GAN those images are created at random.

CatGAN exploits a GAN solution that relies on a multi-class discriminator D, replacing the binary classification of real or not real into the classification of the data points into multiple categories which are defined a priori.

## 2.4 Mixture Of Experts-based

**Mixture of Experts (MoE)** is a popular technique for ensemble models, which was analyzed in this dissertation in the context of clustering with the usage of Mixture of Autoencoders, it is therefore inserted in the autoencoder-based clustering category. The idea is based on a manager which works as a balancing agent, and a set of experts, which are independent neural networks (see also figure 2.2). The goal of this approach is to divide neurons into different sections, each section representing a section of the presented data. With this technique, each expert specializes in a specific subset of the data and the manager ensures that every time the correct expert is chosen for each data point.

### 2.4.1 Clustering

Two important contributions to Mixture of Experts are: **Deep Autoencoder Mixture Clustering (DAMIC)** [30], which introduces the concept of influence (the manager determines, based on the input data, which is the most influential expert by giving a score between 0 and 1 to each expert), and uses K-means to obtain the initial clusters after the pre-training phase; **MIXAE** [31] provides some advances to the DAMIC architectures, such a different approach in the regularization and the fact that it does not require pre-training while preventing cluster collapse (which occurs in the previous architecture). However, the performance is still far from state of the art.

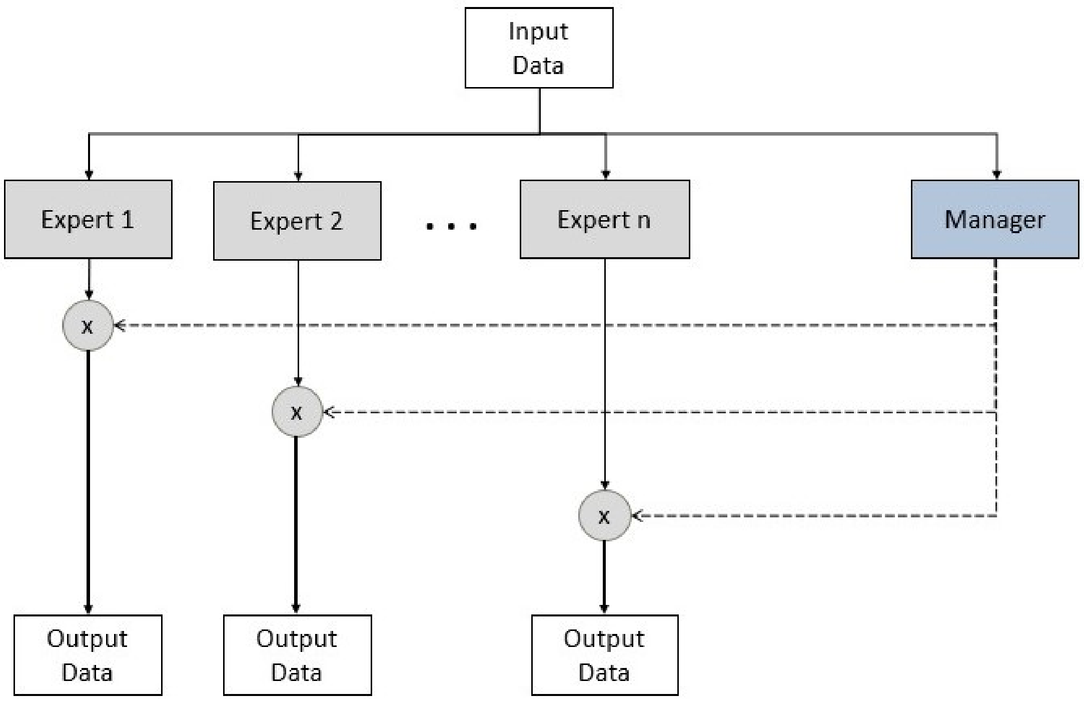


**Figure 2.2:** A diagram of the architecture of a generic MoE.

The work done by Pedro Santos [32], which offered a Mixture of Experts solution, was one of the base building blocks of this dissertation, in which the robustness to missing data was tested. The main contributions offered by this MoE solution stand in the fact that: **(i)** a generative approach was developed, just like this work intends to do; **(ii)** it has a fully data-dependant architecture, removing the need for most hyperparameter selection; **(iii)** by relying on Principal Component Analysis (PCA), it can estimate the dimensionality of the VAE bottleneck layers, simplifying the model; and **(iv)** by relying on Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [33], which is a density-based clustering non-parametric algorithm, (instead of a classical GMM, as in N2D it allows to estimate the number of clusters automatically. A general overview of this system can be seen in figure 2.3

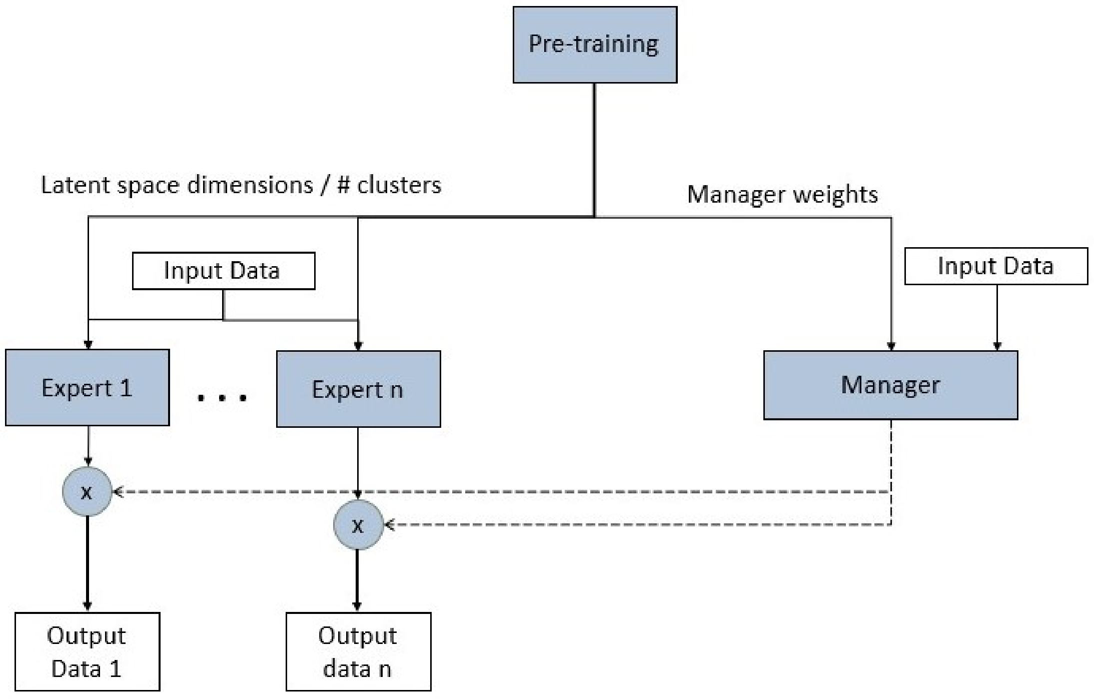


**Figure 2.3:** A diagram of the general architecture of theMoE solution that inspired the proposed work.

In detail, this architecture consists of a set of K experts which are controlled by a manager. The latter, depending on the given input, provides an importance weight to each expert of the model, in a similar way to the DAMIC MoE [34], one of the most notable architectures in that topic. But that approach lead to cluster collapse due to the experts competing against each other instead of collaborating. To attenuate that effect, this solution implemented some changes inspired by the MIXAE [35] architecture in the loss function, batch entropy, and sample entropy were added to the vanilla loss function.

The loss function is therefore composed of 3 elements. The first one is the weighted sum of the reconstruction losses of each expert averaged over the batch:

$$R(\theta) = \frac{1}{\eta}\sum^{\eta} i = 0 \sum^{K} k = 0 p_k^i * d(x_i, \hat{x}_k^i), \tag{2.5}$$

where $p_k^i$ represents the importance associated with sample $i$ for expert $k$, $d(x_i, \hat{x}_k^i)$ a loss function such as MSE or binary crossentropy coupled with a normalized KL loss term, $\eta$ is the batch size, and K is the number of clusters.

The second element of the loss function is the sample-wise entropy averaged over the batch, which forces the distribution of probabilities to follow a one-hot vector encoding:

$$S(\theta) = -\frac{1}{\eta}\sum^{\eta} i = 0 \sum^{K} k = 0 p_k^i * log(p_k^i) \tag{2.6}$$

And the third part is the batch-wise entropy that ensures that there is a balanced distribution of labels, and prevents cluster collapse in the network:

$$B(\theta) = \sum_{k=0}^{K} \hat{p}_k * log(\hat{p}_k), \tag{2.7}$$

where $\hat{p}_k$ is the mean of all predicted importances for the manager in a batch for a specific expert.

So the overall loss functionis the following:

$$L(\theta) = R(\theta) + \alpha S(\theta) + \beta B(\theta), \tag{2.8}$$

where $\alpha$ and $\beta$ are hyperparameters that need to be tuned.

However, the main innovation of this solution was the pre-training architecture. One of the goals of this procedure is to find two different values, the optimal number of clusters to cluster the data, and the optimal size for the latent dimension for each of the experts. It is also in this part that the network is trained which ensures consistent results over different iterations of the algorithm.

As seen in figure 2.4, the pre-training phase starts with the training of an autoencoder-like architecture, which can be a pure AE, a VAE, WAE, etc, in which a predetermined large latent dimension size is used. The latent dimension size finder is then used and the autoencoder is trained again with the new value for the latent dimension. After this, UMAP is used in the latent manifold to reduce dimensionality, and in the output of this step, HDBSCAN is then used, assigning clusters to each data point. Finally using, the labels obtained from HDBSCAN the managers are trained. After this, the MoE architecture can run with the trained managers, as seen in figure 2.3

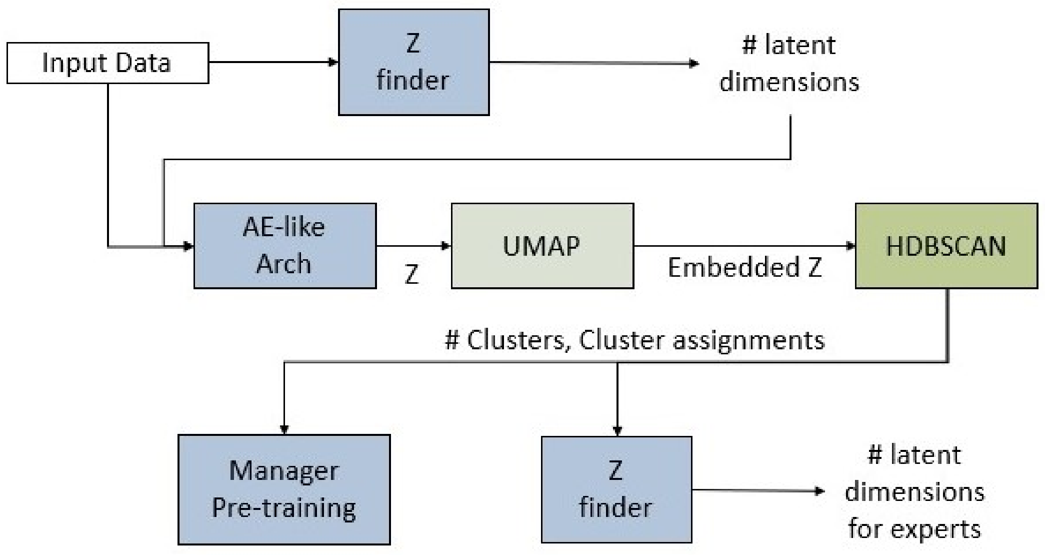


**Figure 2.4:** A diagram of the pre-training of the MoE solution that inspired this work.

## 2.5 Self supervised learning

Many of the previous approaches exploit low-dimensional space representations in order to attain a clustering methodology. At this perspective, one of the most powerful representation learning approaches is contrastive learning, where a noticeable amount of work on the subject has emerged in recent time. This is relevant to mention in this chapter because self-supervised, like unsupervised learning, does not require labeled data for its training, creating its own labels because not only it does not require labeled data for its training (like unsupervised learning, the main focus of this work), but also because a very common approach to model self-supervision architectures in computer vision is to take different parts of an image and then apply different augmentations a passing these modified images as inputs to the model. In this step, a certain connection to the works of this thesis is made, since in this case, missing data is being purposefully created and is being shown to help improve the learning task.

The main goal in contrastive learning is to obtain embeddings where similar sample pairs are kept close to each other and the ones with bigger differences are kept apart from each other.

Throughout the years, different contrastive loss objectives showed up in the literature, but in general, they have in common the idea of using positive and negative samples to obtain the embedidings.

Two very relevant works in this subject are SimCLR [36] and MoCo [37], [38] [39], which at the moment has 3 different versions, with the first two being the most relevant to mention here since v3 is an extension of the idea to work with Vision Transformers (ViT).

SimCLR proposes a simple framework for contrastive learning of visual representation.

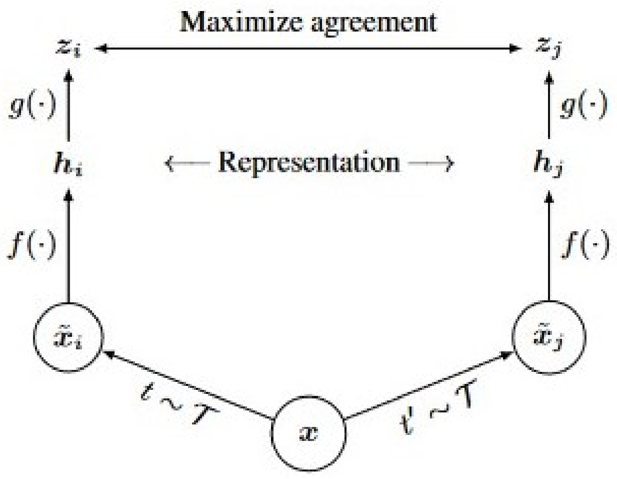


**Figure 2.5:** A simple framework for contrastive learning of visual representations.

Developed by Google, the proposed idea works by first creating different augmented views of the same sample (e.g., through flipping, rotation, noise insertion), and then using all those samples as input, while constraining the network with a contrastive loss to ensure similar space representations. which are then used as input and a representation of each is obtained, the contrastive loss can then be calculated using the latent space obtained. Figure 2.5, from [36], presents a virtual representation of such idea.

The main downside of this framework is that to achieve a good performance it is needed a quite large batch size to be able to incorporate enough negative samples.

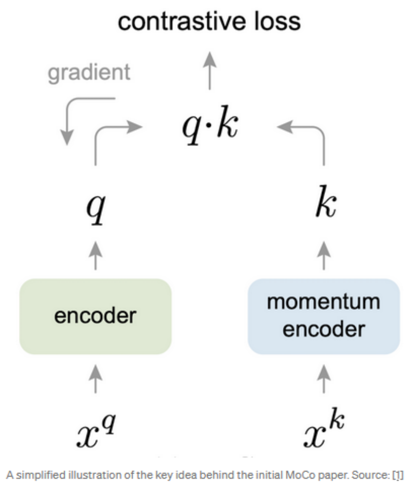

contrastive loss
gradient
$q \cdot k$
$q$
$k$
encoder
momentum encoder
$x^q$
$x^k$
A simplified illustration of the key idea behind the initial MoCo paper. Source: [1]

**Figure 2.6:** Conceptual representation of the contrastive loss mechanisms from MoCo

Figure 2.6, from [37], provides a conceptual representation behind the contrastive loss mechanims from Momentum Contrast (MoCo). In this architecture, a visual representation encoder is trained through a contrastive loss between a encoded query $q$ and a dictionary of encoded keys $k$.

MoCo is not dependent on a large batch size for enough number of negative samples, however, compared to the first version of MoCo, SimCLR proved to be more efficient through the offer of 1) an MLP projection head and 2) stronger data augmentation, which is why the second version of MoCo arrived after SimCLR offering a combination of those ideas and a better performance. The third version of MoCo is the current state of the art and applies the idea of transformers, a subject outside of the scope of this thesis.

## 2.6 Summary

This chapter offered a theoretical review of the literature regarding the categories of Autoencoders, Variational Autoencoders, Generative Adversarial Networks, and Mixture of Experts architectures, each one was explained, how and when they are often used, its advantages and disadvantages, and finally corresponding clustering methodology for each type. In this chapter, a brief overview of self-supervised learning was also provided since some current state-of-the-art works regarding clustering are being done through this learning paradigm.

# 3 Dealing with Missing Data

Contents

For complete datasets, the architectures provide good clustering results. However, their robustness against missing data is not guaranteed and how the architectures respond to missing data is an important factor. The problem arises in the fact that in a real-world scenario, the data is often incomplete.

The problem of missing data can arise from multiple sources and in different types of data. In cases where data is obtained through technical devices such as sensors, data corruption or simple failure can occur, or in image data types this can also include image occlusion derived, for example, from obstacles in automatic cameras. In figure 3.1, taken from [40], the three examples show the problem deriving from sensor failing, dead pixels, and the occlusion caused by clouds being obstacles to the image retrieval.

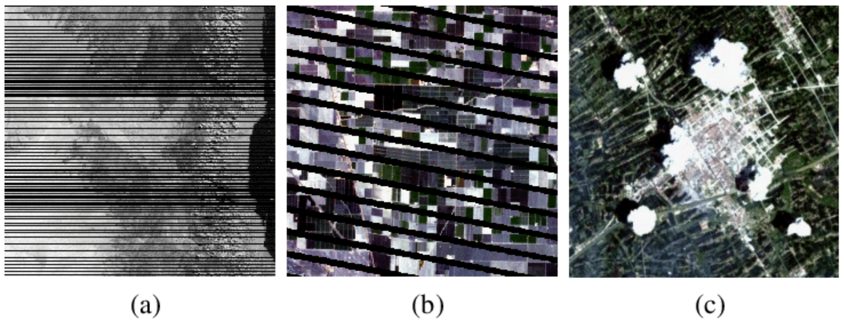


**Figure 3.1:** Examples of missing data in image obtained from sensors.

Human error can also be an influence in the existence of missing data, especially if the data origin is survey-based, something that occurs in several fields [5] as presented in Capítulo 1. It has also been shown that the existence of missing data can affect medical data analysis by inducing a bias which creates a big challenge in medical research, [41], [42]. As the data gets more complex, the influence of human error also increases, and the ways to solve this problem usually rely on simple solutions such as imputation of values. Preprocessing of data is always a necessary step. However, the existence of missing values can create extra complexity to the problem since it implies a need for extra steps to deal with the problem. But even in the cases where the choice is to ignore the missingness, how much could this missingness affect the goal in mind?

Previous work has been done with great results by approaching the problem of deep clustering with the usage of generative models. Yet, most of these advanced deep clustering architectures need complete datasets to achieve good results, which means that when missing data occurs, the results are not as strong.

This lack of robustness to missing values is, therefore, the main motivation for the work to be done, in which it is proposed to explore the creation of a deep clustering model that is generative and robust to this problem.

Before diving into ways to overcome the problem, lets first consider the three main ways that missing data can be classified into:

- **Missing completely at random (MCAR)** - when there is no relationship between the way that the missing occurs and any of the study variables (observed or missing data). This type of missing data is a quite rare phenomenon to occur.
- **Missing at random (MAR)** - in this situation the missing is not affected by the results of the data by itself but by the complete observed data.
- **Missing not at random (MNAR)** - when the reason for the data to be missing is directly related to the values of the data.

## 3.1 Missing Data Imputation

Now that the different types of missing data have been established, the next step is to analyze some basic approaches that are often used across the field to handle missing data.

One of these approaches is through **Deletion**, where any sample that contains missing data is discarded. **Listwise Deletion**, which is when all the instances with missing data are removed, and **Pairwise Deletion**, where the instance is only removed if the needed variable to the computation is missed (if any other variable is missed but not being used, it is not removed). It is important to notice that these techniques are used in the assumption that the data is missed completely at random (MCAR), since it is the only one where the deletion does not add any bias to the results. The power of the analysis is still affected by this technique since the sample size is reduced.

Another important approach is **Imputation**, which is the replacement of the missing data with a specific value. This technique can be applied to continuous variables (numeric) where simple techniques such as mean or median may be used to replace the empty values. For categorical data replacement techniques usually consider the most frequent value in that category. If there is a high number of missing values in that second scenario, a special category just for these values can be created. There are also two ways in which imputation can be performed: Single Imputation and Multiple Imputation. Single Imputation includes techniques such as single value, similarity, and regression imputation where one value at a time is imputed.

Multiple Imputation consists of calculating the average of the outcomes across multiple imputed data sets to account for this. All multiple imputation methods follow three important steps: 1) imputation, 2) analysis and 3) pooling.

- Imputation – In a similar way to single imputation, missing values are imputed. However, multiple copies of the dataset are created and an imputation is performed in each one, originating different substitutions in the missing values.
- Analysis – Each of the datasets is analyzed through different statistical methods.

- Pooling – The results from the above step will form a single imputation through averaging.

Two examples of popular approaches based on this idea include Multiple Imputation by Chained Equations (MICE) [43] and Multiple Imputation with Denoising Autoencoders (MIDAS) [44]. MICE is an older and robust technique that is still being often used. Although very efficient, it can create performance problems with more complex datasets with higher dimensions. On the other hand, MIDAS is a more recent technique that uses machine learning to provide better performance. Although MIDAS is more recent, depending on the dataset, MICE can sometimes still provide better results than MIDAS.

Many of the commonly used techniques for the handling of missing data are usually based on performing imputation before the learning and often use the whole data for the imputation of a single value. This involves a higher cost and complexity, especially in situations with high dimensional data and big data sets. In this work, the goal was therefore to find an approach using deep learning that could be able to handle the missing data and optimally perform data clustering.

## 3.2 Deep Learning Approaches

The handling of missing values in a deep learning approach has been developing at a fast pace recently, and there have been some fresh ideas quite relevant to the topic, as described in the next paragraphs.

**MIWAE** [45] is a technique for the handling of missing data with deep latent variable models. This approach is used when the training set contains missing-at-random data and is built with an importance-weighted autoencoder (IWAE) as a foundation, which proposes a generative model with a similar architecture to a VAE, but with a focus on increasing the flexibility and creating richer latent space representations. This is done with the usage of multiple samples in the recognition network which provides a strictly tighter lower bound. The MIWAE takes this base idea and applies it to missing data, solving the problem of additional computational overhead and focusing on creating missing data imputation.

**not-MIWAE** [46], is a similar approach to MIWAE, from the same authors, which was created recently, to tackle problems where the missing process is dependent on the missing data. This is particularly relevant as in such cases the missing process has to be explicitly modeled and taken into account while doing likelihood-based inference.

**Variational Selective Autoencoder (VSAE)** [47], focus on models for multimodal data imputation. It learns from partially-observed data and it works by modeling the joint distribution of observed/ unobserved modalities and the imputation mask, which results in a unified model for various down-stream tasks including data generation and imputation.

**Robust Variational Autoencoders (RVAE)** [48], has a focus on outlier detection and it is a deep generative model that learns the joint distribution of the clean data while identifying the outlier cells, and with this allows for their imputation. RVAE learns the probability of each cell being an outlier through the balancing of different likelihood models in the row outlier score, which makes this model a suitable one for detection in mixed-type datasets.

**Variational deep embedding with recurrence (VADER)** [49] is a method that relies on a Gaussian mixture variational autoencoder framework which was extended to model multivariate time series and directly deals with missing values.

Many of the state-of-the-art approaches to the missing value problem use Variational Autoencoders as a foundation for the architecture, with the appropriate changes to the respective problem, however, there is a lack of work focusing on the clustering of data, with the majority approaching either the problem of imputation or classification.

## 3.3 Clustering With Missing Data

The approaches above focus on the learning of models with missing data, often to provide either imputation or classification. However, they do not approach the clustering of this data. In this section, the focus is therefore on the clustering part of the problem.

Starting with **K-pod** [50] is a missing data approach that works by extending a K-means clustering algorithm to work with missing data, however since it is a shallow clustering method is more appropriate for smaller datasets.

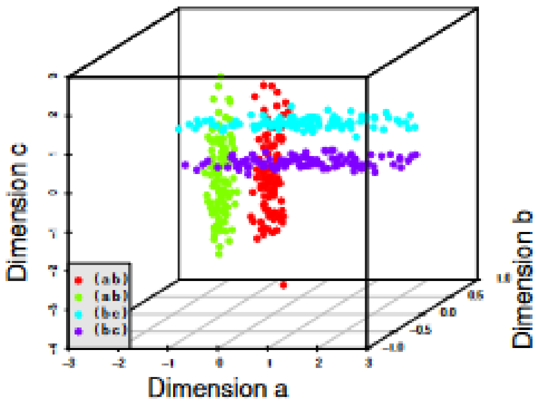


**Figure 3.2:** Subspace clustering.

**Subspace clustering** is a big topic in the literature that works well on high-dimensional data. It extends classical clustering into finding clusters within different subspaces on a dataset. By finding clusters that exist in multiple and/or overlapping subspaces, it allows the algorithms to localize the most relevant dimensions [51]. Figure 3.2, from From [51], shows a good example of a scenario containing overlapping clusters: since the points of different clusters can be quite close, the results of some traditional clustering algorithms could be easily affected, whereas subspace clustering is still able to identify the different clusters. Recently there has been some work that introduces missing data into this type of algorithm. In particular, [52] offers two methods focusing on the problem of ”partially observed” data and sees the problem of SCMD as a generalization of a low-rank matrix completion problem. Similar ideas can be seen in [53] and [54]

Some work has also been done around the topic of **graph clustering** with deep learning models. Due to the great potential of the usage of graphs across different branches of science, the merge of graph theory with deep learning lead to the emergence of Graph Neural Networks (GNNs) in the last years.

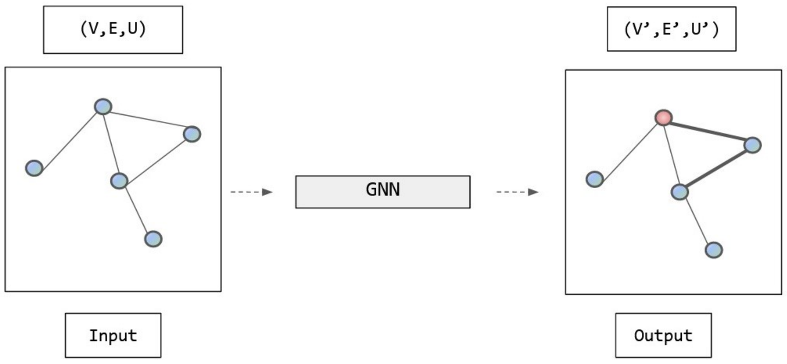


**Figure 3.3:** Generic Graph Neural Network

The figure 3.3 shows the very simple idea of a graph transformation of a graph in a network, with its input represented as *(U,V,E)* where *U* represents the node attributes, *V* the global attributes and *E* the edge attributes, the combination is used as an input into any network and applied the necessary transformations.

The usage of deep learning for multiple graph analysis has mostly been focused around tasks such as node classification and link prediction, but there as also been some work approaching the problem from a clustering point of view. The goal is to separate the nodes of the graph into different clusters, with the edge structures of the graph being taken into account, leading to a result where there are multiple edges within each clusters and a small number between different clusters.

Variational Graph Auto-Encoders are at the base of most relevant work in the graph clustering tasks, with the current state-of-the-art being based on this idea. This solution is built using a graph convolutional network (GCN) encoder and a simple inner product decoder. A key feature of this work is that some missing data was introduced in the data before the training, through the removal of a percentage of the edges of the graphs.

This could be an interesting approach to exploring the problem of missing data since graphs are flexible structures of data that could also be used to represent data types such as images or texts, which can be modeled as regularized graphs, due to their fixed number of neighbors.

The whole idea of applying deep learning to graph data structures is a whole field per se and even though this type of work aims mostly at data with more heterogeneous structures, there is great potential to use the flexible properties of using graph-structured data.

There has also been some work with **multi-view clustering** using missing data, this type of technique has a focus on multi-view data, which is very common across the field of big data. The goal of this type of task is to consider data from distinct feature sets or “views” and retrieve meaningful information in a way that considers how the data from different views complement each other and their consensus.

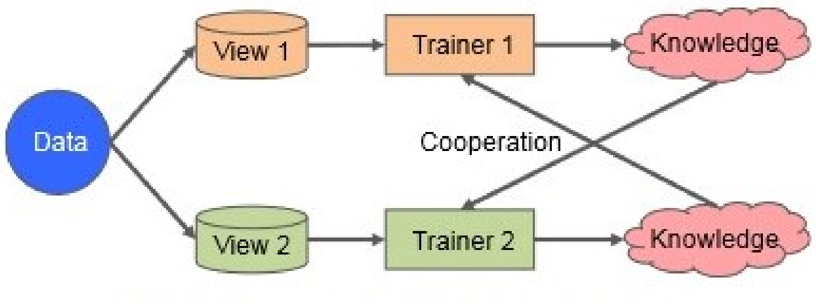


**Figure 3.4:** Generic procedure behind General procedure of co-training, one of the most most relevant multi-view clustering methods. Image from [1]

Examples of this include multimedia, where both a video and an audio signal can be used to represent a media segment, or when using image data obtained from different devices to film the same object. In the scope of missing data, the complementary information from the different views can be used to retrieve the existent missing values. An example of this type of idea is presented in 3.4

The ideas analysed in this section seem to show great potential on dealing with missing data problems, which also proves the relevancy of the work being done in this thesis. They look at the problem however in different lenses, focusing either in different types of data, or being more developed for classification tasks, such as the case with GNN’s.

## 3.4 Summary

In this chapter it was observed that there are multiple ways to approach the problem of missing data, by changing the way that data is represented, and that even though there are many works focusing on clustering, missing data and deep learning, there is a certain gap in the literature in the combination of these ideas.

Missing data can derive from multiple sources, from human to machine, and in different types of data from image to flat data. It can also have random origins such as error, or even have a factor for the lack of information. But in any origin, it can create problems by inducing bias and error in results. Some deep learning models have arrived in recent years with a major focus on classification, but there is still a lack of study in deep clustering models with a focus on missing data, which will be the focus on the methods proposed in the following section.

# 4 Proposed Methodology

**Contents**

The development of this thesis started with the creation of an improvement of a standard VAE through the creation of an AE-based architecture to use as a foundation that could be able to work with a variety of datasets and that was flexible to the maximum amount of changes possible. The focus of this dissertation was on image data, however, this scheme can be extended to other data such as survey or sensor obtained data, or even graph-structured data. This extension possibility was tested out with the implementation of an alternative network where the convolutional layers were replaced by dense layers and received conventional numerical data, this same clustering methodology could be applied for this case even though its results are not presented.

With this in mind, the first step was the implementation of an AE-like architecture built with dynamic blocks, the goal was that this flexible architecture could be transformed and adapted into a variety of AE-based architectures by changing the necessary key features such as its loss function and that it could also be adjusted into receiving different data, with the blocks being adapted into the data size.

For the clustering of the data in the latent space of the AE, with the usage of the UMAP dimensionality reduction approach, a manifold learning technique that is applied to the latent space of the autoencoder before a standard clustering algorithm helped improve the quality of the defined clusters, hence leading to more accurate results. The chosen shallow clustering is executed after this step of learning the representations of the data. In particular, HDBSCAN clustering is used, which provides the advantage of automatic detection of the appropriate number of clusters, and also outlier detection.

These key ideas for the clustering of data, and the dynamic convolutional blocks were then used as a base and applied to three different main architectures: 1) a masked variational autoencoder, 2) an IWAE - imputed weights autoencoder, [55] an alternative to the classical VAE that uses a strictly tighter log-likelihood lower bound derived from importance weighting; 3) MIWAE, which is based on IWAE but particularly developed for missing value imputation. Some experiments using a MoCo were also done, by testing out the embedding obtained when running the model and applying a UMAP + HDBSCAN clustering approach.

In this chapter, these architectures will be analyzed in detail.

## 4.1 Global Architecture

An overall idea of the logic behind how the experimented architectures were used can be seen in figure 4.1. An AE-based model is used to encode the image dataset. With the obtained latent, space a dimensionality reduction is performed and clustering analysis is carried out in these embeddings. The reconstructions obtained by the decoders were also used to analyze the difference in images. In the initial stages the original image datasets were used, this later evolved into the addition of patches with missing values, and the creation of binary masks that could be used on the loss functions of the models.

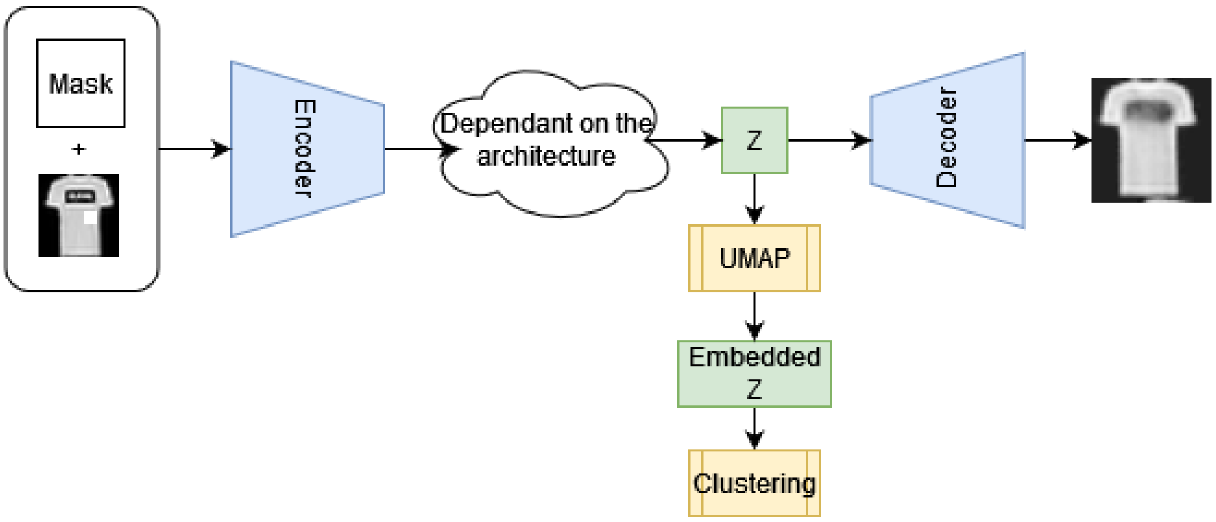


**Figure 4.1:** General architecture

As most clustering algorithms suffer from the curse of dimensionality, a dimensionality reduction of the embedding was seen as a fundamental step. This was done through Uniform Manifold Approximation and Projection (UMAP), a technique for manifold learning used for dimension reduction that can be applied in a similar way to t-SNE for visualization, while being strong at preserving the structure of the data in smaller dimensions in a fast and efficient way. Due to the importance of dimensionality reduction in the data science field, this technique is considered a viable choice for its usage in machine learning. [21] Therefore, for the clustering of the latent space, the first step was to use UMAP for dimensionality reduction. In this step, two variations were tested, one where the parameters were the default ones from the library, and in a alternative, a dynamic number of neighbors was defined. This dynamic number of neighbors influences how locally the data is viewed. The following expression was followed:

$$n\_neigh = max\{int\Big(\frac{dataset\_size}{300}\Big), 100\} \tag{4.1}$$

After dimensionality reduction with UMAP, a shallow clustering algorithm is applied. Different solutions could be applied for this, such as Gaussian Mixture Models (as in [20]), K-Means (as in [18]), or Hierarchical Clustering. However, in this thesis the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [33] is chosen. HDBSCAN is an extension of the DBSCAN algorithm which transforms it into a hierarchical clustering algorithm. This algorithm offers some advantages compared to classic shallow clustering techniques such as K-means and GMM, namely its ability to automatically detect outliers by creating a different cluster destined for this type of data point. Another advantage is the fact that the amount of clusters is detected automatically according to the data, which is a substantial improvement versus most deep learning-based clustering solutions where the number of clusters must be known a-priori.

## 4.2 Masked VAE

One of the main contributions of this dissertation comes from introducing a binary mask when facing missing values on the data. In the proposed solution, a variant of a normal VAE is implemented by changing the loss function of the model. In this variation, a binary mask from the missing points of the data is created. With the mask obtained, it is then used as input along with the images, and in the moment of the loss calculations, multiplied by the reconstructed image.

The goal of this approach was to modify the reconstruction loss function to use a binary mask $m_i \in \{0,1\}^L$ which indicates if a certain value is missing. Thus, the loss function, which corresponds to an MSE, is computed over the observed values in the following way:

$$L_{rec} = \frac{1}{N}\sum_{i=1}^{n}(x_i' - \tilde{x}_i{}')^2, \tag{4.2}$$

where $x' = x \circ m$ represents an element-wise product between the original input data $x$ and the mask $m$, and $\tilde{x}' = \tilde{x} \circ m$ an element-wise product between the reconstructed data $\tilde{x}$ and the mask $m$.

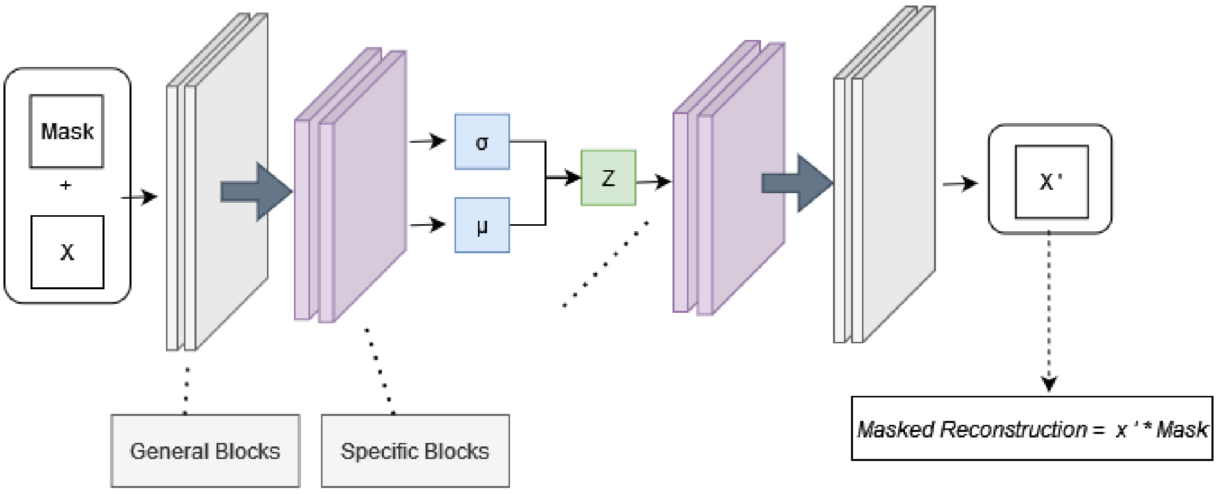


**Figure 4.2:** Masked VAE

This technique aims for the bias introduced by missing data to be ignored by assigning an error value of zero to the areas of the images where missing values are observed.

A visual representation of the VAE model adapted to this idea is represented in figure 4.2.

The idea of a binary mask was in here introduced in 3 different architectures, but it is a solution scalable to multiple types of structures that use a compatible loss function.

## 4.3 IWAE

Importance Weighted Autoencoders [55], is a similar generative model to VAE, offering to learn richer representations with more latent representations. The key difference is that through importance weighting, the generative model is trained with a tighter log-likelihood lower bound. This is done through the generation of multiple ”approximate posterior” samples in the recognition network, with averaged weights being used.

This model does not focus on missing data or clustering, however, it is the main base for MIWAE, a model explored in detail in the next section that adopts this model into solving the overhead created by missing data. To explore the fundamental approach to importance weights, some models were implemented and tested. In all of them, the clustering above described was applied in the latent space. Two main implementations are being considered for this analysis: 1) an implementation of the original IWAE, and *2) An adaptation of the base VAE from this dissertation's work with its latent space transformed and the encoder output changed into returning multiple samples.* Since the results from the second implementation were not satisfactory, the results obtained were dropped and only the ones from the first implementation were considered for the final analysis.

The original architecture was tested using the MNIST and the OMNIGLOT dataset and it was not meant for its usage with clustering, however, for this work, the same clustering methodology was used by picking its latent space and performing clustering. Since OMNIGLOT is considered to be more adequate for one-shot classification tasks, its testing was discarded since it's not as relevant for this work.

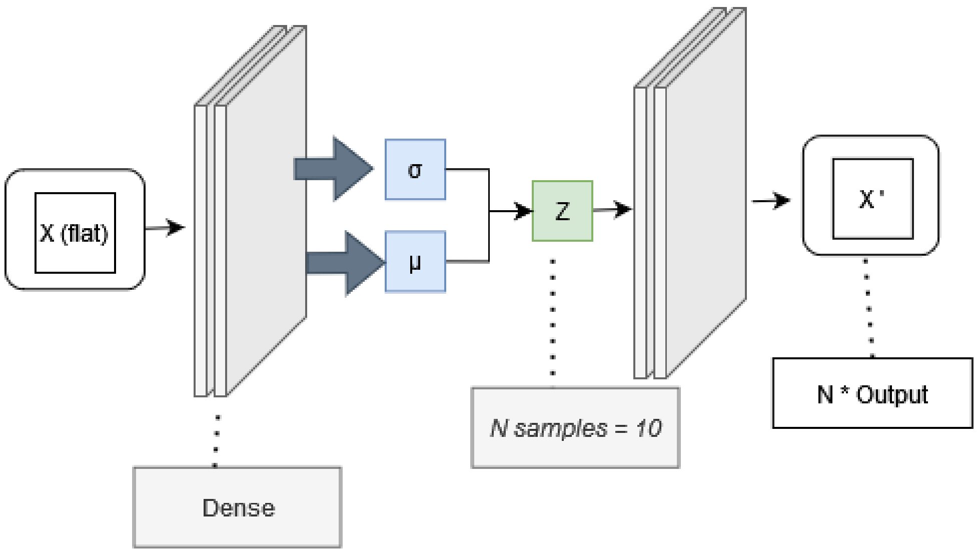


**Figure 4.3:** Original IWAE

The main reasoning for the study of this specific architecture comes from another work in the area, in [45], MIWAE was proposed, used some of the ideas of this solution with a focus on the problem of

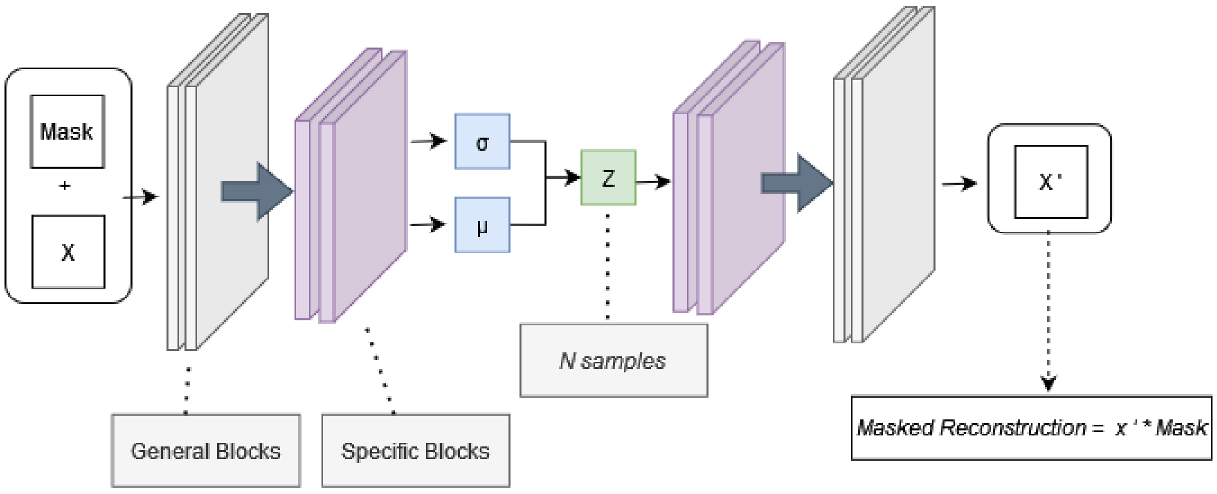


**Figure 4.4:** IWAE applied with the general architecture

missing data, where it is also introduced a binary mask to the solution.

## 4.4 MIWAE

The MIWAE [45] model focuses on the handling of missing at random data through deep learning. It is based on the IWAE architecture above, and similar to the idea offered in the masked VAE, a binary mask is used during the training. In the original implementation, however, the focus is on continuous datasets and the model is built with a few dense layers.

To work with this extension of the IWAE architecture, some alterations were made. This work was focused around using very simple synthetic numeric datasets, with its main focus being the imputation of the data. To adapt to the problem of this work, an AE-like architecture with convolutional blocks adapted from the one here proposed was implemented. This allowed for the MIWAE architecture to work with convolutional layers and support image datasets. Since this architecture also used the idea of a mask for its training, the same logic was kept. 4.5

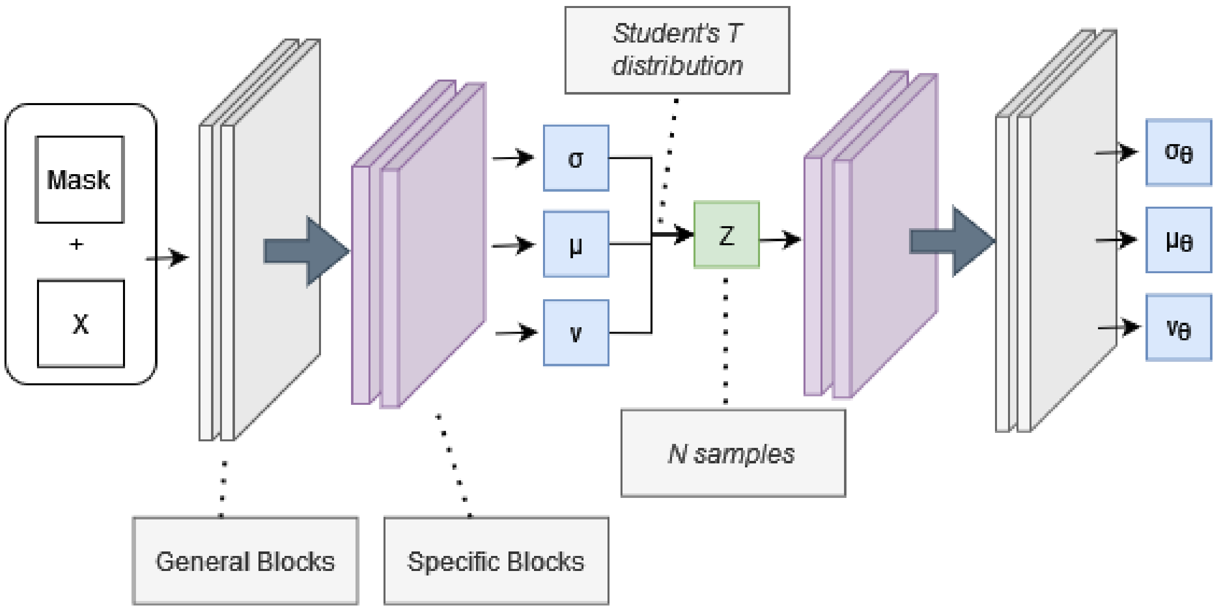


**Figure 4.5:** MIWAE

The formula for the MIWAE bound that is used for the training and for the computing of imputation is shown in 4.3.

$$\mathcal{L}_k(\theta,\gamma) = \sum_{i=1}^{n} E_{z_{i1}}, ..., z_{ik} \sim q_\gamma(z|x_i^o)[log\frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(x_i^o|z_{ik})p(z_{ik})}{q_\gamma(z_{ik}|x_i^o)}] \tag{4.3}$$

## 4.5 Architecture Details

To transform the approach into a more general structure, a flexible and data-independent architecture was developed based on a convolutional block that can be adapted to different types of autoencoder-based models. Based on the advantages of the usage of more dynamic architectures by mixing different kernel sizes, the convolutional blocks were added. The order for these convolutional blocks is represented in figure 4.6, the reasoning behind their structure comes from the advantages of using more dynamic architectures by mixing different kernel sizes. Two distinct blocks are used for this, a general and a specific. In the general block the more general and significant features of the image are reconstructed, while in the specific blocks, more particular features are reconstructed.

Two of the most common choices for filter sizes are 3x3 or 5x5, mostly due to memory and simplicity concerns. In this implementation, the choice was two 3x3 blocks, since even though both have the same receptive field, the chosen one does not require as many mathematical operations, which leads to less training time [56] Another important aspect of the blocks is the usage of Batch Normalization [57] before

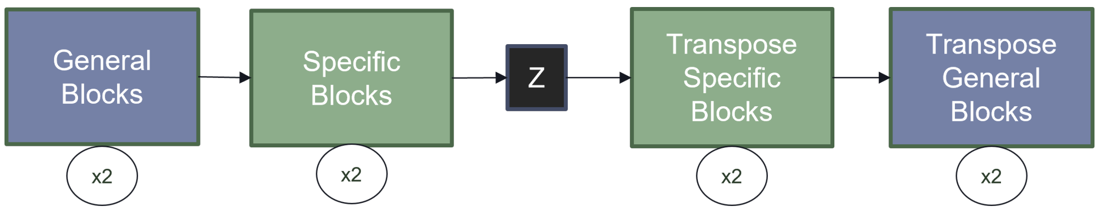


**Figure 4.6:** Diagram Blocks structure that can be aplied to any Autoencoder architecture

the activation function, according to [58]. These blocks can be seen in figure 4.7.

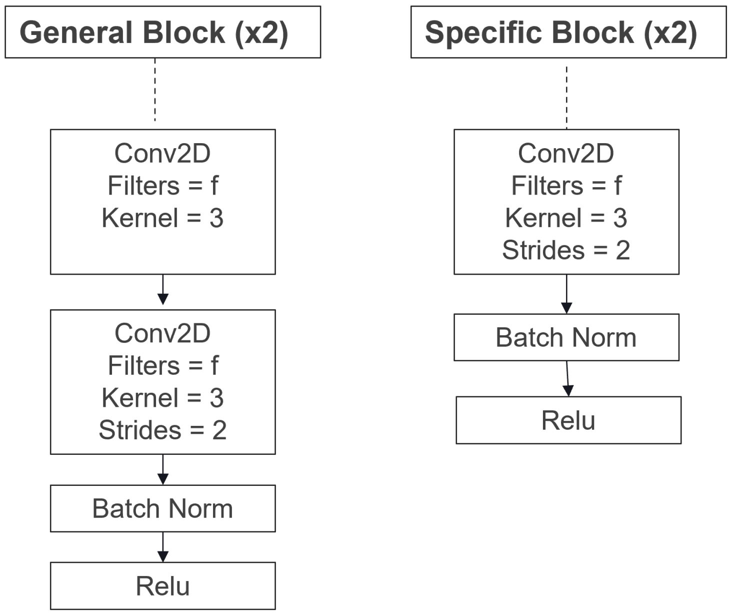


**Figure 4.7:** Diagram of blocks

Another important factor is the depth of the architecture, this depth is dependent on the data since it is not an optimal choice to set the same characteristics defined for different datasets. With that in mind, the depth of the network and the number of filters are dependent on the image size of the dataset being used.

Assuming that an image has dimensions DxD, the number of blocks in the encoder can be calculated by:

$$N_{Blocks} = log_2(D) - 1 \tag{4.4}$$

This amount of blocks in the encoder assures that no matter how big the image is, before the latent space of the encoder the data is 2x2xF in size, where F is the number of filters in the last layer of the encoder. The number of general and specific blocks is given by:

$$N_{general} = \lceil N_{Blocks}/2 \rceil \tag{4.5}$$

$$N_{specific} = N_{Blocks} - N_{general} \tag{4.6}$$

The number of filters is given by multiplying D by two for each block of the encoder. This provides an approximately equal number of general and specific blocks in the architecture. The autoencoder-like architecture can be seen summarized in figure 4.8, when using the example of a dataset with an image size of 32x32.

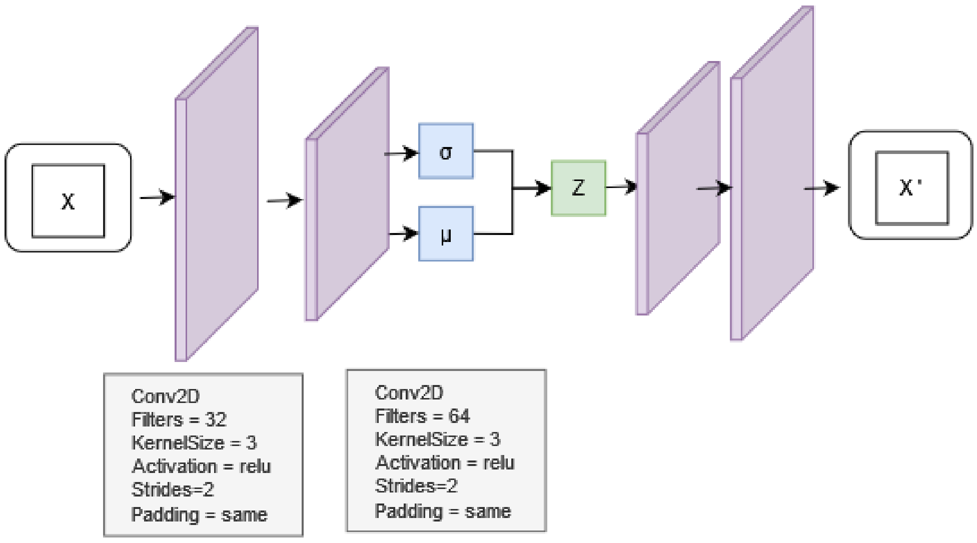


**Figure 4.8:** Vanilla Autoencoder

This AE-like architecture can accommodate any type of autoencoder by changing the latent space. In this dissertation, two architectures from current literature were also an object of study, with a special focus on MIWAE, as it is a state-of-the-art VAE-based imputation method. However, as MIWAE is not initially designed by considering images as inputs, it was specifically adapted to work the remaining VAE structure, as previously described.

## 4.6 Miscellaneous

Throughout this work, some research, implementation, and experiments of other methods in the literature were done with less dept due to unsatisfactory results. This includes an analysis and testing on the MoCo [39] framework, where the implementation by [59] was used partially for its trained instances, where the with the obtained embedding was used with the clustering tested in this dissertations. The clustering accuracy obtained was quite low compared to the results obtained in these frameworks so it was concluded that these architectures are better suited for classification tasks.

## 4.7 Summary

In this chapter an overview of the main architectures was provided, where the key features of the developed VAE are explained, such as the dynamic convolutional blocks that are dependent on the data used and that can be applied to other AE-based architectures, something that was done with the MIWAE. Another key feature is the usage of a binary mask on the loss function, a change that allowed for the bias created by missing values to be reduced and to allow for better clustering and reconstruction of images, something that will be shown in chapter 6. On the path to the final solution, a study of more complex techniques such as MoCo was also approached by taking the calculated embeddings and introducing the UMAP+HDBSCAN clustering as well, these experiments showed however that this type of solution is more appropriate for classification tasks and that it is also focused on more complex datasets such as ImageNet and STL10. Another topic of study was the WAE model, which failed to provide satisfactory results and was therefore dropped from the implementation since it showed not to be as appropriate for a missing data problem approach.

# 5

# Experiments and results

## Contents

This chapter evaluates the proposed methodology, by providing details into a selection of design choices, the influence of multiple factors and provide the results obtain from diverse experiments that give strength to the proposed model. A special attention is given to how a dataset with missing data was created, the creation of a stable model and how it was affected with the existence of missing data, and finally the exploration of multiple ideas to overcome the problem.

## 5.1 Datasets used

- MNIST: A dataset of handwritten digits images with 70000 examples separated into ten classes. Each sample is a 28x28 grayscale image.

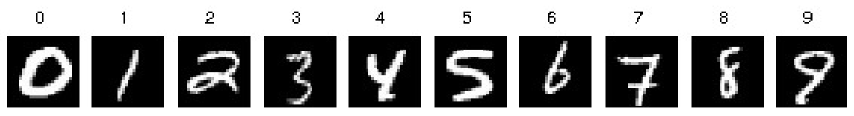


**Figure 5.1:** Example images of the MNIST dataset

- FMNIST: A dataset of fashion items images with 70000 examples separated into ten classes. Each sample is a 28x28 grayscale image.

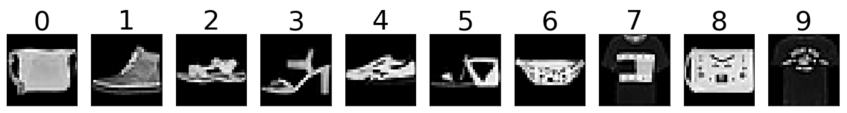


**Figure 5.2:** Example images of the FMNIST dataset

- USPS: A dataset of handwritten digits images with 9298 examples separated into ten classes. Each sample is a 16x16 grayscale image.

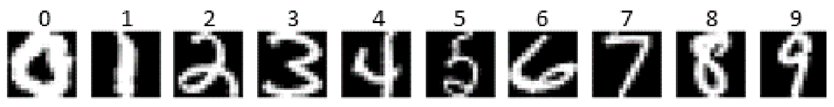


**Figure 5.3:** Example images of the USPS dataset

- Coil: The Columbia Object Image Library (COIL-20) dataset contains images of 20 objects, and for each of them there are 72 images captured every 5 degrees along a viewing circle. Each sample is a 128x128 grayscale image.

**Figure 5.4:** Example images of the COIL20 dataset

To better fit the architecture, image sizes with a size of $2^n$ are better suited due to the methods used to calculate the number of blocks and build the architecture. Therefore, the FMNIST and MNIST samples were padded from 28x28 to 32x32. For the USPS dataset, since the images were already suited no changes were implemented. The COIL20 dataset could hypothetically fit the architecture, however with an original size of 128x128 the resulted architecture would become overly complicated, to simplify it, the images were downsampled from 128x128 to 64x64 using an antialiasing filter.

## 5.2 Creation of Patches

For the generation of an alternative dataset with random missing data, in the initial stages of this work this was achieved simply by adding empty patches of 10x10 pixels in random central areas of the image, but it later evolved into adding a variance in the height and width of the patches. The reasoning for the focus on the central parts of the image is due to the fact that the most important information occurs in the center sections of the images, therefore inserting patches on the edges would not cause a significant impact on the results, since the inserted patches would be considered background. Hence, for the selection of the location of the patch, a margin of 4 pixels in the center of the image is considered and the starting pixel is obtained from a random uniform distribution with a range from *i=4* to *i=width-4* and *j=4* to *j=height-4*. For the width and height of the inserted patches, the goal was to obtain up to 10% of the pixels missing, which is reflected in a range from 4 to 10 pixels in the datasets with the size of 32x32 and a bigger range (10 to 20 pixels) for the COIL20 dataset since its image size is 64x64. For the USPS dataset, since there is more space in the image likely to have meaningful information, the patch size of 4 to 10 pixels was kept. This logic is represented in figure 5.5

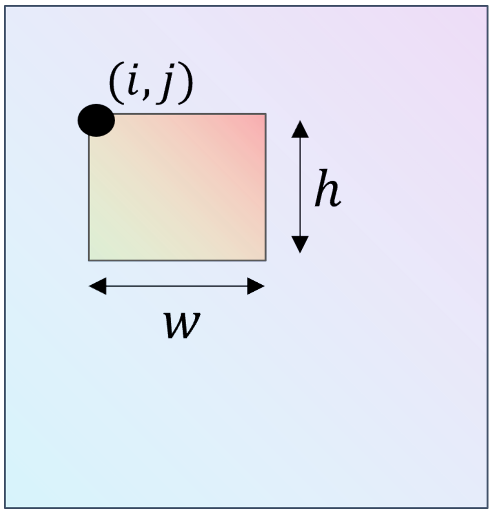


**Figure 5.5:** Creation of patches

Until the end of this document, the new datasets will be referenced as:

- MNIST-Patches: The result of adding random patches to all instances from the MNIST dataset.

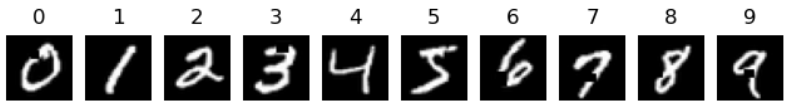


**Figure 5.6:** Example images of the MNIST-patches dataset

- FMNIST-Patches: The result of adding random patches to all instances from the FMNIST dataset.

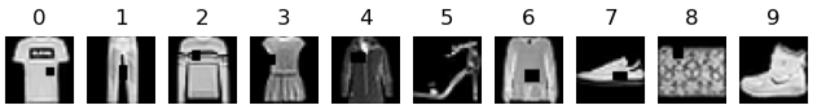


**Figure 5.7:** Example images of the FMNIST-Patches dataset

- USPS-Patches: The result of adding random patches to all instances from the USPS dataset.

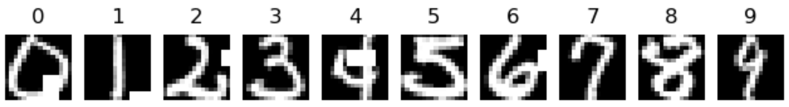


**Figure 5.8:** Example images of the USPS-Patches dataset

- COIL20-Patches: The result of adding random patches to all instances from the COIL20 dataset.

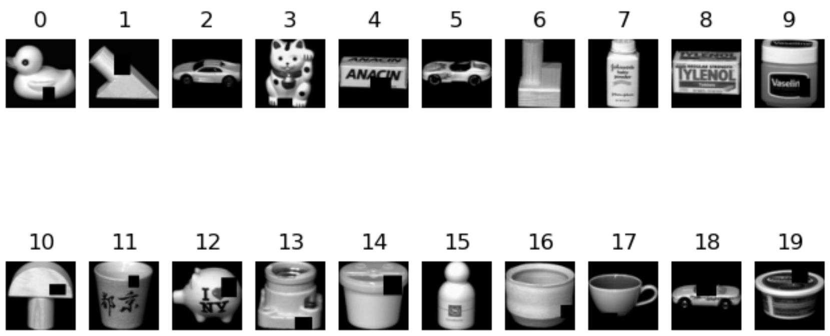


**Figure 5.9:** Example images of the COIL20-Patches dataset

The datasets with missing values also include the changes mentioned in the section above, with a resizing of the MNIST and FMNIST samples from 28x28 to 32x32, and the COIL20 dataset being downsampled from 128x128 to 64x64.

## 5.3 Evaluation

Clustering accuracy was used to evaluate the performance of HDBSCAN and GMM, which measures the proportion of data points for which the obtained clusters can be correctly mapped to the correct classes. This mapping can be obtained using the Hungarian algorithm [60], and the accuracy is given by:

$$ACC(y_{true}, y_{pred}) = max_T(\frac{\sum_{i=1}^{N} 1(y_{true}(i) = T(y_{pred}(i)))}{N}), \tag{5.1}$$

where $y_{true}$ represents the ground truth labels, $y_{pred}$ the predicted labels, N is the total number of samples, and finally, T is the best one-to-one mapping that matches the clustering indexes to the ground truth labels.

For the HDBSCAN clustering, two different accuracies were generated, one involving the full data set, and the other one only taking into account the points that were correctly labeled, ignoring the outliers.

For a richer analysis of the results, a GMM clustering accuracy was also retrieved from the same embeddings obtained in the tested models.

Throughout the results shown in this chapter, UMAP was a constant factor in every clustering calculation, where it was used on the embedding before running the clustering algorithm. It is therefore of big importance to first see its impact when used in different situations. Table 5.1, provides an analysis of the impact of adding dimensionality reduction with UMAP when applied to shallow clustering algorithms.

One of the metrics used for this were ARS - Adjusted Rand index [61], which is a way to measure the similarity of the two label sets, ignoring permutations and AMI, Adjusted mutual information, which is a way to measure the agreement between two sets, in this case, the obtained labels and the original labels. AMI was more recently proposed and works similarly to Normalized Mutual Information (NMI), which is also often used in the literature.

In table 5.1, it is possible to see the comparison of the usage of UMAP in three cases where shallow clustering is being used: 1) Direct application of the K-means algorithm on a flattened input data, 2) K-means on the embedding obtained from a VAE, 3) GMM with the embedding obtained from VAE. The dataset used for this was a complete version of the MNIST dataset, on the built VAE. For all cases UMAP significantly improved the results, which provided confirmation to the advantages of UMAP offered in the literature [20].

| | No UMAP | | | UMAP | | |
|---|---|---|---|---|---|---|
| | ARS | AMI | ACC | ARS | AMI | ACC |
| K-means | 0.37 | 0.50 | 0.53 | **0.79** | **0.86** | **0.82** |
| K-Means + VAE | 0.52 | 0.62 | 0.65 | **0.95** | **0.94** | **0.98** |
| GMM + VAE | 0.83 | 0.86 | 0.92 | **0.96** | **0.95** | **0.98** |

**Table 5.1:** Impact of the usage of UMAP as a dimensionality reduction method when used along with shallow clustering methods

With the improvements from dimensionality reduction obtained, the next step was to observe the difference when performing a density-based clustering algorithm. In the table 5.2 the results from the HDBSCAN algorithm when using full labels were added, and since the accuracies obtained were similar, throughout the experiments GMM clustering was also considered.

| | ARS | AMI | ACC |
|---|---|---|---|
| VAE + UMAP + GMM | 0.96 | 0.95 | 0.98 |
| VAE + UMAP + HDBSCAN (Full Labels) | 0.95 | 0.94 | 0.98 |

**Table 5.2:** Results from HDBSCAN and GMM clustering.

## 5.4 Results of convolutional blocks architecture

The first implementation step was to build a stable architecture that was able to provide good clustering results in a complete dataset. Table 5.3 presents the clustering results on the MNIST dataset of the implementation of the VAE with the dynamic convolutional blocks as described in chapter 4.

The first three lines represent the clustering through the usage of UMAP + HDBSCAN, where the first line shows that the standard VAE is able to assign 67% of the data points to a cluster, with a 43% accuracy, while when considering the whole dataset this accuracy drops to 32%. In the second column, we can see the first indicator of success by the noticeable difference in accuracy, which increases to 98% on both methods of clustering (and with all datapoints being assigned a cluster).

| | Basic VAE | VAE with Dynamic Convolutional Blocks |
|---|---|---|
| Percentage of labeled points | 0.67 | 1 |
| Accuracy of labeled points | 0.43 | 0.98 |
| Accuracy on full dataset | 0.32 | 0.98 |
| Accuracy of gmm | 0.38 | 0.98 |

**Table 5.3:** Influence of the dynamic blocks architecture with MNIST dataset

## 5.5 Influence of Patches in Clustering Results

After obtaining a stable architecture providing good results on complete datasets, the next step was the analysis of the impact of introducing missing values in the data used for model training. A first glance into this impact is represented in table 5.4, where the clustering results when considering the full dataset (i.e. including the data points not labeled in the HDBSCAN algorithm) shows that for all the datasets where missing data was introduced, a decrease of the accuracy occurred.

| | Full image | | With Patches | |
|---|---|---|---|---|
| | Med | $\sigma$ | Med | $\sigma$ |
| MNIST | 0.978 | 0.007 | 0.914 | 0.008 |
| FASHION MNIST | 0.584 | 0.009 | 0.537 | 0.017 |
| USPS | 0.967 | 0.002 | 0.873 | 0.057 |
| COIL20 | 0.803 | 0.016 | 0.759 | 0.021 |

**Table 5.4:** Impact of missing values in the accuracy from UMAP + HDBSCAN considering the full dataset. For this experiment the median and standard deviation of 10 runs is considered.

Also considering the same case but considering the labeled data by the HDBSCAN algorithm, table 5.5 show the influence on the accuracy of the successfully labeled points, as well as the difference in the percentage of data that the algorithm was able to assign labels.

| | | Full image | | With Patches | |
|---|---|---|---|---|---|
| | | Med | $\sigma$ | Med | $\sigma$ |
| MNIST | Percentage of labeled points | 0.980 | 0.004 | 0.90 | 0.009 |
| | Accuracy of labeled points | 0.983 | 0.0005 | 0.974 | 0.001 |
| FASHION MNIST | Percentage of labeled points | 0.73 | 0.062 | 0.84 | 0.017 |
| | Accuracy of labeled points | 0.716 | 0.033 | 0.609 | 0.011 |
| USPS | Percentage of labeled points | 0.99 | 0.009 | 0.94 | 0.005 |
| | Accuracy of labeled points | 0.973 | 0.001 | 0.968 | 0.001 |
| COIL20 | Percentage of labeled points | 0.95 | 0.029 | 0.89 | 0.077 |
| | Accuracy of labeled points | 0.834 | 0.022 | 0.819 | 0.066 |

**Table 5.5:** Impact of missing values in the percentage and accuracy of HDBSCAN labeled data. For this experiment the median and standard deviation of 10 runs is considered.

For the full dataset, there seems to be a correlation between the size of the images and the amount of information contained, for the USPS dataset, where the image sizes were smaller and the images more simple, the biggest difference occurred, with a loss of 10% of accuracy, while the smallest difference can be seen in the COIL20 dataset, the one with the biggest images size.

For HDBSCAN labeled data the accuracies also dropped, as well as the percentage of labeled, which can be seen in 5.5

## 5.6 Influence of Mask in the results

After the analysis of the impact of adding missing data analyzed, the next step was to adapt the model to overcome the problem. This is where the usage of a binary mask in the model inputs was introduced with its results being shown in this section. To ensure a more rigorous quality of results, for each trained instance the results were obtained 10 times, with a median and a standard deviation being calculated.

The results showed that through the implementation of a mask on the dynamic VAE, the sensibility to missing data was softened in most cases.

For the FMNIST-Patches dataset, represented in table 5.6 the difference can be seen in the accuracy results for GMM with a difference of 6% and HDBSCAN on the full dataset with 4%. For the subset of labeled data, there seems to be a similar value, however, the percentage of labeled points also increased, which shows that certain data points were now clustered with the introduction of the mask.

| | FMNIST-Patches + Mask | | FMNIST-Patches + No Mask VAE | |
|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Percentage of labeled points | **0.83** | 0.029 | 0.69 | 0.083 |
| Accuracy of labeled points | 0.675 | 0.009 | 0.688 | 0.025 |
| Accuracy on full dataset | **0.578** | 0.008 | 0.537 | 0.017 |
| Accuracy of GMM | **0.661** | 0.021 | 0.607 | 0.018 |

**Table 5.6:** Influence of mask on FMNIST-Patches dataset

On the MNIST-Patches dataset, represented in table 5.7, the results align with the observations on the FMNIST-Patches dataset, the HDBSCAN algorithm improved  2% on the full dataset, and the percentage of labeled points also increased. In this case, however, the GMM algorithm showed similar results for both cases.

| | MNIST-Patches + Masked VAE | | MNIST-Patches + Standard VAE | |
|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Percentage of labeled points | **0.97** | 0.008 | 0.9 | 0.022 |
| Accuracy of labeled points | 0.972 | 0.001 | 0.974 | 0.002 |
| Accuracy on full dataset | **0.944** | 0.004 | 0.922 | 0.008 |
| Accuracy of GMM | 0.968 | 0.005 | 0.964 | 0.001 |

**Table 5.7:** Influence of mask on MNIST-Patches dataset

On the USPS-Patches dataset, the impact is shown in table 5.8, the biggest difference can be observed, when applying the HDBSCAN algorithm to the full dataset. Also, the amount of labeled points increases with the proposed approach compared to the standard VAE.

| | USPS-Patches + Masked VAE | | USPS-Patches + Standard VAE | |
|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Percentage of labeled points | **0.95** | 0.006 | 0.84 | 0.049 |
| Accuracy of labeled points | **0.973** | 0.001 | 0.928 | 0.018 |
| Accuracy on full dataset | **0.94** | 0.005 | 0.873 | 0.057 |
| Accuracy of GMM | **0.965** | 0.001 | 0.955 | 0.001 |

**Table 5.8:** Influence of mask on USPS-Patches dataset

Finally, for the COIL20-Patches, represented in table 5.9, the results do not show improvement of the proposed approach, with only a slight variation in the results of HDBSCAN.

| | COIL20-Patches + Masked VAE | | COIL20-Patches + Standard VAE | |
|---|---|---|---|---|
| UMAP - dynamic parameters | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Percentage of labeled points | 0.94 | 0.051 | 0.91 | 0.037 |
| Accuracy of labeled points | 0.879 | 0.034 | 0.888 | 0.026 |
| Accuracy on full dataset | 0.847 | 0.008 | 0.841 | 0.017 |
| Accuracy of gmm | 0.5 | 0.0 | 0.5 | 0.0 |

**Table 5.9:** Influence of mask on COIL20-Patches dataset

## 5.7 Alternative misssing values imputation

The VAE needs to have a complete dataset to be able to train, consequently, the missing pixels were substituted by zero after obtaining a binary mask, however, the option to simply replace by other values was still a possibility. Therefore, some experiments were done with this in mind, focusing on the alternatives of using 1)an initialization with random values, 2) the average of the closest pixel imputation and 3) the average of two closest pixels. In this scenario, the dataset used as an example is the FMNIST-Patches.

Table 5.10 shows the results of this experiment, it can be concluded that replacing these values with zero is the best option, since not only were the results better when the techniques looking at neighbors were tested it created a relevant increase in computational complexity.

| | HDBSCAN | | | GMM |
|---|---|---|---|---|
| | Percentage of labeled points | Accuracy of labeled points | Full dataset acurracy | |
| Random | 0.71 | 0.657 | 0.526 | 0.574 |
| Zero | **0.83** | **0.675** | **0.578** | **0.661** |
| KNN = 1 | 0.73 | 0.656 | 0.527 | 0.576 |
| KNN = 2 | 0.7 | 0.658 | 0.520 | 0.565 |

**Table 5.10:** Clustering resuls for FMNIST-Patches dataset, when considering alternative missing values imputation techniques

## 5.8 MIWAE and IWAE architectures

For IWAE and MIWAE, two of the implemented architectures described in chapter 4, similar experiments were attempted, where the MNIST and fashion MNIST datasets were used with and without patches of missing data added.

On the path to obtaining the most optimal MIWAE implementation, an analysis of a few parameters was done, for example, the number of samples $K$ used in the latent space. The final choice was to keep *K=1* since the improvements of using a bigger number of samples did not show great results and it caused a bigger computational overhead.

| | K=5 | | K=1 | |
|---|---|---|---|---|
| | med | stnd dev | med | stnd dev |
| Percentage of labeled points | 0.68 | 0.08 | 0.59 | 0.1 |
| Accuracy of labeled points | 0.6 | 0.03 | 0.6 | 0.05 |
| Accuracy on full dataset | 0.47 | 0.01 | 0.49 | 0.03 |
| Accuracy of GMM | 0.54 | 0.01 | 0.54 | 0 |

The MIWAE implementation did not show to provide a significant improvement in clustering results compared to the VAE architecture, however, as seen in the image reconstruction section, it was successful in the task of imputing missing data.

The clustering results for the best cases with the VAE and MIWAE models can be observed in the following table:

| | VAE - No Mask | | VAE - MASK | | MIWAE | |
|---|---|---|---|---|---|---|
| | med | stnd dev | med | stnd dev | med | stnd dev |
| Percentage of labeled points | 0.69 | 0.083 | 0.83 | 0.029 | 0.59 | 0.1 |
| Accuracy of labeled points | 0.688 | 0.025 | 0.675 | 0.009 | 0.6 | 0.05 |
| Accuracy on full dataset | 0.537 | 0.017 | 0.578 | 0.008 | 0.49 | 0.03 |
| Accuracy of GMM | 0.607 | 0.018 | 0.661 | 0.021 | 0.54 | 0 |

**Table 5.11:** Accuracies of FMNIST-Patches on VAE and MIWAE architectures

For the IWAE architecture, the baseline used was an implementation available online. [1] It showed very satisfactory results in the complete MNIST dataset, even though it did not surpass the VAE model proposed in this work. This model also showed one of the biggest impactful differences in clustering results when used with missing values, with the accuracy on the full dataset when using HDBSCAN dropping from 92% to 76%. The introduction of a binary mask showed a slight improvement as well. Similar conclusions can be drawn for the FMNIST dataset, see table 5.13.

| | MNIST-Patches | | MNIST |
|---|---|---|---|
| | No mask | Mask | |
| Percentage of labeled points | 0,75 | 0,79 | 0,94 |
| Accuracy of labeled points | 0,927 | 0,924 | 0,968 |
| Accuracy on full dataset | 0,764 | 0,799 | 0,924 |
| Accuracy of gmm | 0,85 | 0,85 | 0,95 |

**Table 5.12:** IWAE results on MNIST dataset

Overall the MIWAE and IWAE models did not surpass the proposed VAE architecture, however, some interesting conclusions can be obtained. The overall idea of using multiple samples seems to show that it is not an optimal path to approach the problem, but it does show that the usage of a mechanism to

[1] Implementation available at https://github.com/nbip/IWAE

| | FMNIST-Patches | | FMNIST |
|---|---|---|---|
| | No mask | Mask | |
| Percentage of labeled points | 0,41 | 0,48 | 0,62 |
| Accuracy of labeled points | 0,67 | 0,60 | 0,62 |
| Accuracy on full dataset | 0,35 | 0,36 | 0,4268 |
| Accuracy of gmm | 0,48 | 0,47 | 0,5884 |

**Table 5.13:** IWAE results on FMNIST dataset

| | IWAE | | MIWAE | VAE | |
|---|---|---|---|---|---|
| | No mask | Mask | | No Mask | Mask |
| Percentage of labeled points | 0,41 | 0,48 | 0.59 | 0.69 | **0.83** |
| Accuracy of labeled points | 0,67 | 0,60 | 0.6 | 0.69 | **0.68** |
| Accuracy on full dataset | 0,35 | 0,36 | 0.49 | 0.54 | **0.58** |
| Accuracy of gmm | 0,48 | 0,47 | 0.544 | 0.60 | **0.66** |

**Table 5.14:** Comparison between clustering results on FMNIST-Patches betweeen IWAE, MIWAE and VAE

alter the loss function depending on the existing missing values can significantly improve the results.

## 5.9 Image Reconstruction

For the analysis of the reconstruction of images from the VAE, IWAE, and MIWAE architectures, the visual outputs obtained were observed and some reconstruction metrics were retrieved, namely Mean Squared Error (MSE) and Structural Similarity Index (SSIM) , which is often used to quantify image quality degradation on data compression or data transmission processes [62]. For the IWAE and VAE models, the results were obtained with and without the usage of the binary mask.

Table 5.15 summarizes the MSE between the patched dataset used as input to the AE-based architectures, and the reconstructed output of the model being analyzed. For this metric the VAE architecture showed the best results, however, the effect of the usage of the mask is not revealed in this metric.

| MSE | FMNIST | MNIST | USPS | COIL20 |
|---|---|---|---|---|
| MIWAE | 0.629 | 0.099 | 0.879 | 0 |
| IWAE - No Mask | 0.186 | 0.055 | - | - |
| IWAE - Mask | 0.186 | 0.055 | - | |
| VAE - No Mask | 0.007 | 0.003 | 0.771 | 0.010 |
| VAE - Mask | 0.015 | 0.003 | 0.754 | 0.009 |

**Table 5.15:** Mean Squared Error between patched dataset and the reconstructions obtained in different variants of the model

For the Structural Similarity Index (SSIM) metric, presented in table 5.16 where the goal is to have the highest similarity possible, the VAE also was the one that showed the best values and with a slight improvement when using the binary mask.

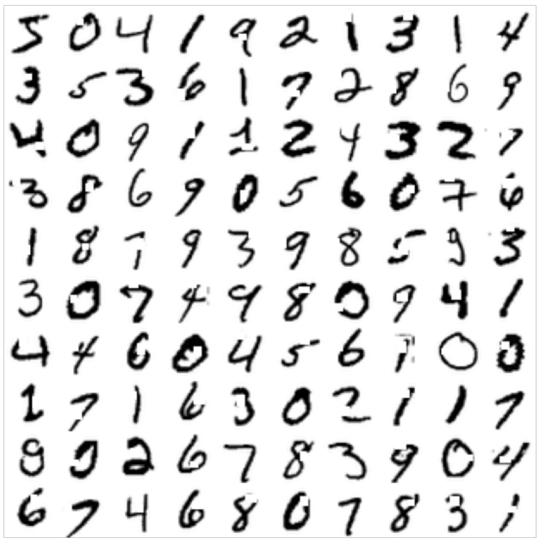
| SSIM | FMNIST | MNIST | USPS | COIL |
|---|---|---|---|---|
| MIWAE | -0.031 | 0.308 | 0.816 | 0 |
| IWAE - No Mask | 0.498 | 0.828 | - | - |
| IWAE - Mask | 0.498 | 0.828 | - | - |
| VAE - No Mask | 0.942 | 0.052 | -0.358 | 0.010 |
| VAE - Mask | 0.758 | 0.941 | -0.331 | 0.867 |

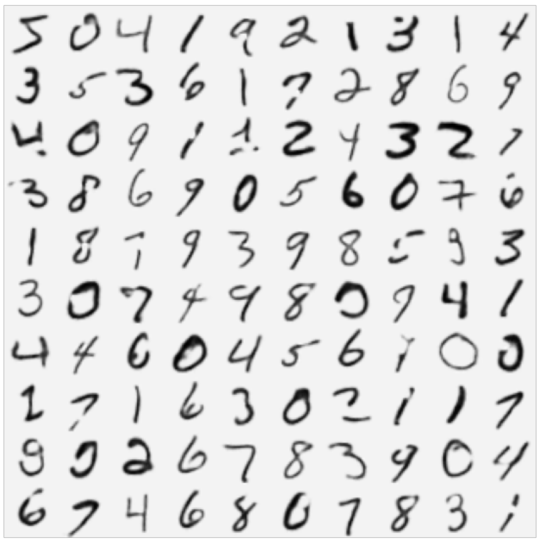
**Table 5.16:** Structural Similarity Index (SSIM) results

The biggest observation from the reconstructed data comes from the actual images obtained.

In the following images, a representation of a) the patched data that the models received, b) the reconstruction of the same data when using no mask during training and c) the same reconstruction when using a binary mask.

The MNIST-Patches, figure 5.10 and FMNIST-Patches, figure 5.11, are the ones where the highest difference can be observed. When using no mask an interesting phenomenon occurs where the color of the overall images also changes, this is a sign of how a slight removal of information can cause a bias in the final results and reconstructions. When observing the patches in more detail, which is easier to observe through the FMNIST-Patches due to the bigger amount of information in each image, it is possible to see that the patches disappeared almost entirely in the reconstructed image, although with some information lost and the color of the whole image is affected.


**(a)** Input


**(b)** No Mask

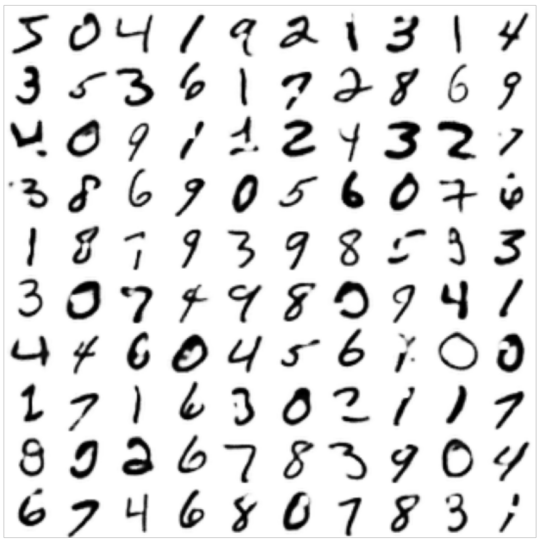
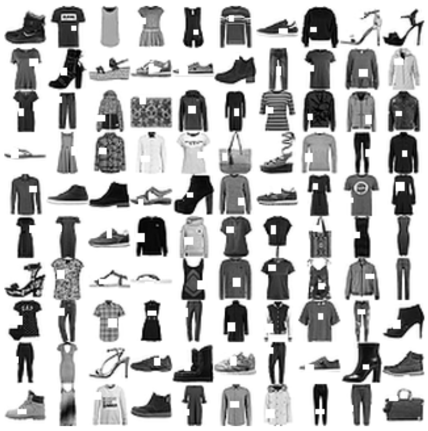
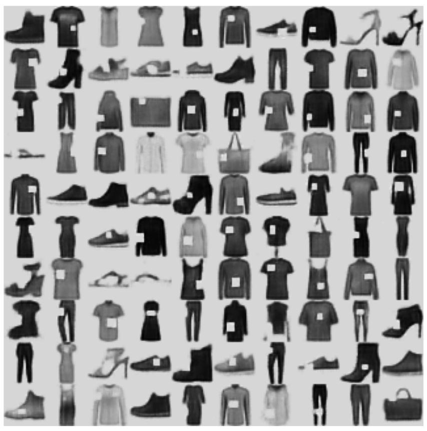

**(c)** Mask

**Figure 5.10:** MNIST reconstructions


**(a)** Input


**(b)** No Mask

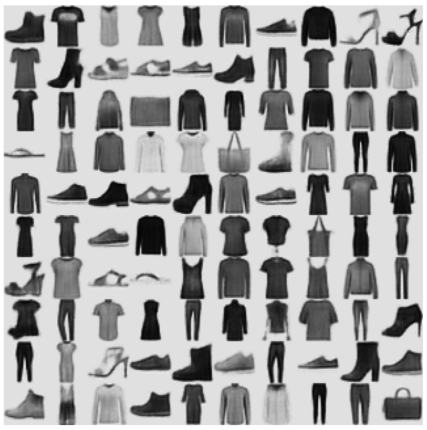
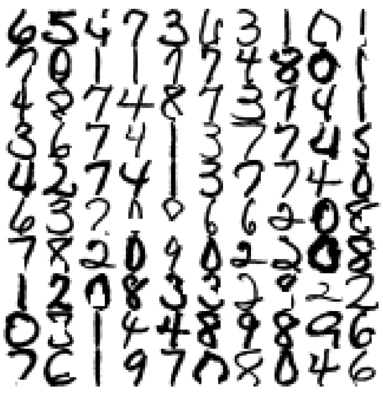
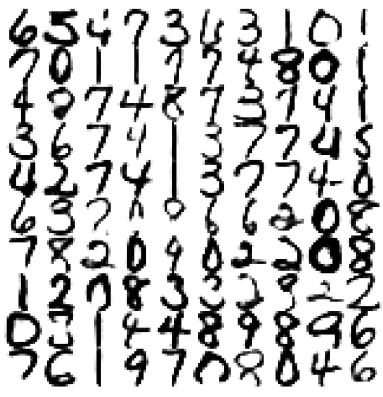

**(c)** Mask

**Figure 5.11:** Fashion MNIST reconstructions


**(a)** Input


**(b)** No Mask

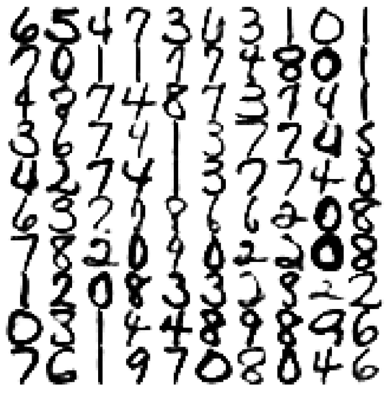
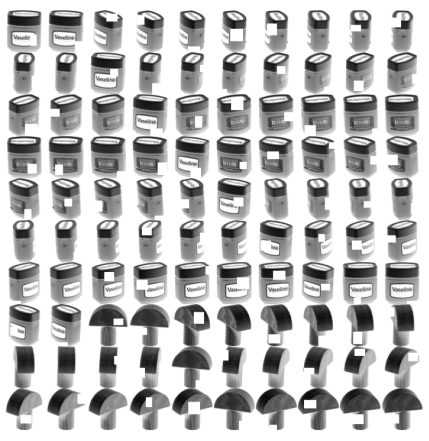
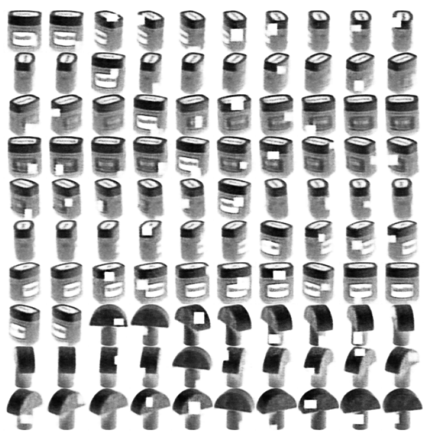

**(c)** Mask

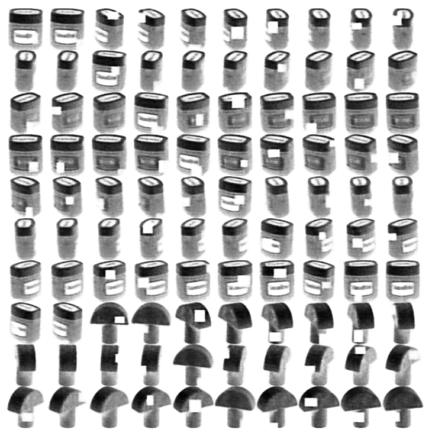
**Figure 5.12:** USPS reconstructions



**(a)** Input



**(b)** No Mask



**(c)** Mask

**Figure 5.13:** COIL20 reconstructions

## 5.10 Cluster Detection Analysis

When analyzing the detected number of clusters by the HDBSCAN algorithm, some interesting remarks were made on the FMNIST dataset regarding the importance of context in machine learning fields. The FMNIST dataset has originally 10 labels, however, when training using this dataset, most of the time, the final results detected one extra cluster on the data. When analyzing this in-depth it is noticeable that while the original cluster for the label ”bag”, for example, included images of bags with and without handles, the model considered them to be completely different instances. The same occurred for images of sandals, where the model divided them into a cluster of flat sandals and a cluster of high-heeled sandals. As humans that know the context, we may know that those belong to the same category, however, in these pictures they are very different cases visually. The case for the split clusters is presented in figures 5.14 and 5.15

At the same time, some of the existing clusters were merged, on the original labels, there are 5 separate clusters for T-shirts/Tops, Pullovers, Dresses, Coats, and shirts. However when training models with this data, the most common result is 4 clusters for these images, with the dresses being commonly mistaken with tops or jackets. This example is presented in figure 5.16

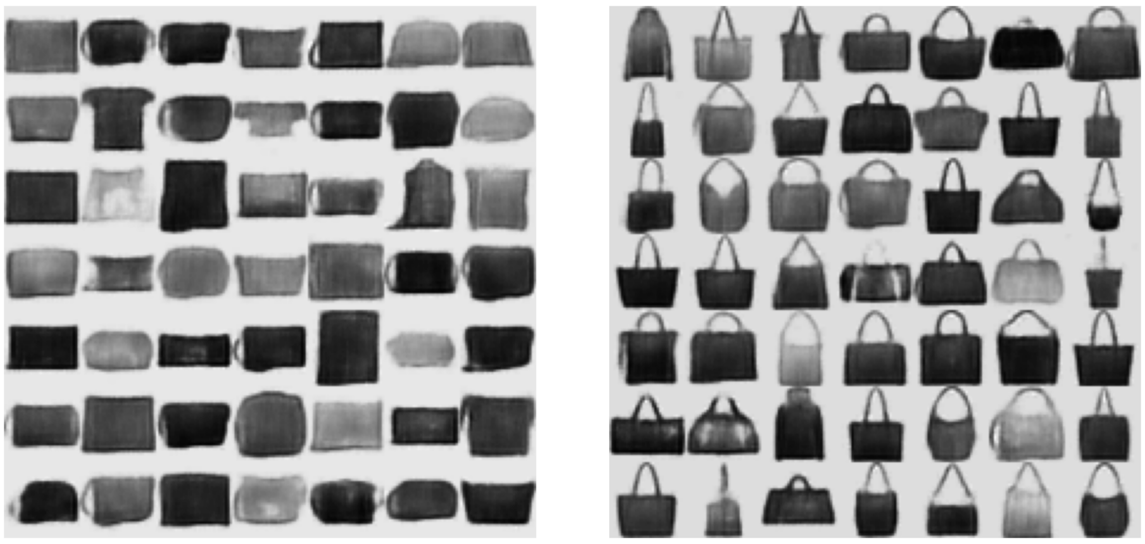
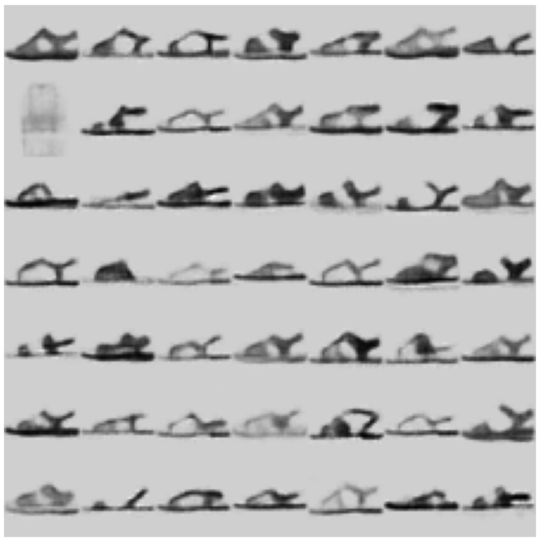

**(a)** Bags with no handle **(b)** Bags with handle

**Figure 5.14:** FMNIST clusters for bags separated by the algorithm into two different ones.


**(a)** Sandals with no heels

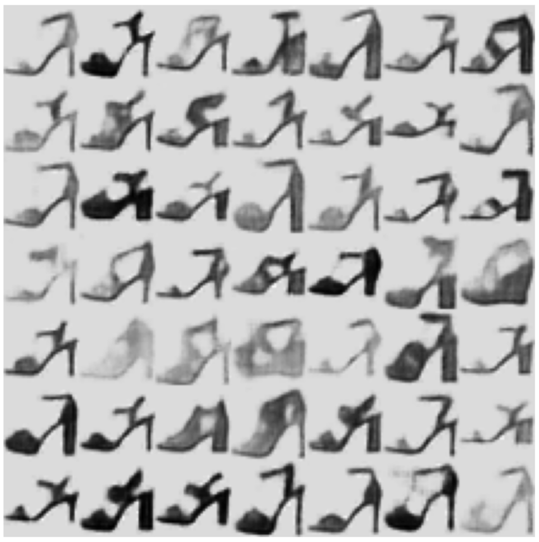

**(b)** Sandals with heels

**Figure 5.15:** FMNIST clusters for sandals separated by the algorithm into two different ones.

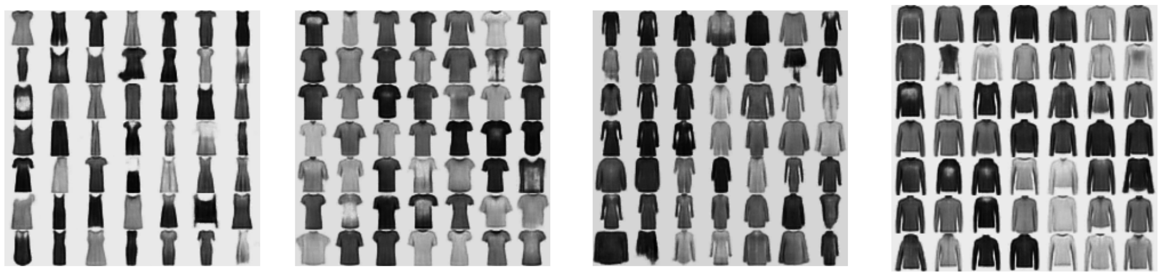

**(a)** **(b)** **(c)** **(d)**

**Figure 5.16:** FMNIST clusters for Jackets, Tshirts, tops, pullovers and dresses.

## 5.11 Final Clustering

The final accuracy results for HDBSCAN on full data, for complete and incomplete datasets, and on the multiple architectures explored in this dissertation are shown in table 5.17. This allows a full observation of the impact of patches and how the VAE with the usage of a binary mask provides the best results.

| Clustering on full data set | FMNIST | FMNIST - Patches | MNIST | MNIST - Patches | USPS - Patches | COIL20 - Patches |
|---|---|---|---|---|---|---|
| MIWAE | 0.42 | 0.47 | | 0.44 | - | - |
| IWAE | 0.42 | 0.35 | 0,925 | 0.764 | - | - |
| VAE Masked | 0.590 | 0.578 | 0.98 | 0.94 | 0.944 | 0.811 |
| VAE | 0.584 | 0.537 | 0.98 | 0.922 | 0.873 | 0.759 |
| MIWAE + VAE | | 0.52 | | | | |

**Table 5.17:** Accuracy on full dataset from the implemented architectures and other architectures in the literature for comparison

For a better comparison, table 5.18 shows the perfomance of some of the most common clustering algorithms from the current literature on complete datasets.

| | FMNIST | MNIST | USPS | COIL20 |
|---|---|---|---|---|
| K-means | 0.474 | 0.532 | 0.668 | - |
| SC-Ncut | 0.508 | 0.656 | 0.649 | - |
| GMM | 0.463 | 0.389 | 0.562 | - |
| AE+K-Means | 0.566 | 0.818 | 0.662 | - |
| DEC [16] | 0.518 | 0.89 | 0.762 | - |
| N2D [20] | 0.672 | 0.979 | 0.958 | - |
| DynAE | 0.591 | 0.987 | 0.981 | - |
| DEPICT [19] | 0.392 | 0.965 | 0.899 | - |
| VaDE [26] | 0.578 | 0.945 | 0.566 | - |

**Table 5.18:** Clustering performance from different algorithms in the datasets analysed in this thesis

Overall the developed VAE architecture in this work, especially after introducing the binary mask achieved the goals and is able to compete with the many different techniques existent in the literature even when missing data was introduced.

## 5.12 Summary

This chapter provided an analysis of the results offered by the proposed methodology. The design choices from the experiments were explained in more detail, and the results from a selection of experiments that gave strength to the solution were analyzed.

# 6

# Conclusion

**Contents**

This chapter provides some closing remarks about the work done in this thesis, the achieved results, and existent possibilities for future work.

## 6.1 Conclusions

With the increasing growth of data-related fields, missing data problems is a recurring problem that is fundamental to approach. In this dissertation, the impact of this obstacle when performing clustering in deep learning models was analyzed and verified. Different ideas were studied, tested, and analyzed, including experiments using imputed weights and multiple sampling, with one of the greatest factors that offered the most improvements in the results was the usage of a binary mask in the loss function.

Starting with a simple variational autoencoder that evolved into a more complex architecture dependent on data and that through the usage of a binary mask in the loss function, was able to overcome some of the impact created when missing data in images was added to the problem.

## 6.2 Limitations and Future Work

Due to the flexibility of the implemented architecture and the usage of a binary mask when facing missing data problems, this work could be potentially used to continue researching and exploring more AE-based architectures. The adaptation of this idea to other data types could also be a possibility and was even lightly tested out with simple data.

# Bibliography

[1] Y. Yang and H. Wang, “Multi-view clustering: A survey,” *Big Data Mining and Analytics*, vol. 1, no. 2, pp. 83–107, 2018.

[2] “Sensors: technologies and global markets,” 2022. [Online]. Available: https://www.bccresearch.com/market-research/instrumentation-and-sensors/sensors-technologies-markets-report.html

[3] “Melanoma lesion detection and segmentation using deep region based convolutional neural network and fuzzy c-means clustering,” *International Journal of Medical Informatics*, vol. 124, pp. 37–48, 4 2019.

[4] M. Phillips, H. Marsden, W. Jaffe, R. N. Matin, G. N. Wali, J. Greenhalgh, E. McGrath, R. James, E. Ladoyanni, A. Bewley, G. Argenziano, and I. Palamaras, “Assessment of accuracy of an artificial intelligence algorithm to detect melanoma in images of skin lesions,” *JAMA Network Open*, vol. 2, pp. e1 913 436–e1 913 436, 10 2019.

[5] L. . Zou, J. . Zheng, C. . Miao, M. J. Mckeown, and Z. J. Wang, “3d cnn based automatic diagnosis of attention deficit hyperactivity disorder using functional and structural mri diagnosis of attention deficit hyperactivity disorder using functional and structural mri. 3d cnn based automatic diagnosis of attention deficit hyperactivity disorder using functional and structural mri,” 2017.

[6] H. S. Choi, J. Y. Choe, H. Kim, J. W. Han, Y. K. Chi, K. Kim, J. Hong, T. Kim, T. H. Kim, S. Yoon, and K. W. Kim, “Deep learning based low-cost high-accuracy diagnostic framework for dementia using comprehensive neuropsychological assessment profiles,” *BMC Geriatrics*, vol. 18, pp. 1–12, 10 2018.

[7] K. H. Thung, P. T. Yap, and D. Shen, “Multi-stage diagnosis of alzheimer’s disease with incomplete multimodal data via multi-task deep learning,” *Deep learning in medical image analysis and multimodal learning for clinical decision support : Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, held in conjunction with MICCAI 2017 Quebec City, QC,...*, vol. 10553, pp. 160–168, 2017.

[8] F. Abid, “A survey of machine learning algorithms based forest fires prediction and detection systems,” *Fire Technology*, vol. 57, pp. 559–590, 3 2021.

[9] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu, and D. Liu, “A deep learning based forest fire detection approach using uav and yolov3,” *1st International Conference on Industrial Artificial Intelligence, IAI 2019*, 7 2019.

[10] T. P. Carvalho, F. A. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance,” *Computers Industrial Engineering*, vol. 137, p. 106024, 11 2019.

[11] J. Tautz-Weinert and S. J. Watson, “Using scada data for wind turbine condition monitoring – a review,” *IET Renewable Power Generation*, vol. 11, pp. 382–394, 3 2017.

[12] F. Wang, L. P. Casalino, and D. Khullar, “Deep learning in medicine—promise, progress, and challenges,” *JAMA Internal Medicine*, vol. 179, pp. 293–294, 3 2019.

[13] C. Y. Cheng, W. L. Tseng, C. F. Chang, C. H. Chang, and S. S. F. Gau, “A deep learning approach for missing data imputation of rating scales assessing attention-deficit hyperactivity disorder,” *Frontiers in Psychiatry*, vol. 11, p. 673, 7 2020.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” 2011.

[16] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[17] F. Li, H. Qiao, B. Zhang, and X. Xi, “Discriminatively boosted image clustering with fully convolutional auto-encoders,” 2017.

[18] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” in *international conference on machine learning*. PMLR, 2017, pp. 3861–3870.

[19] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, “Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization.”

[20] R. McConville, R. Santos-Rodríguez, R. J. Piechocki, and I. Craddock, “N2d: (not too) deep clustering via clustering the local manifold of an autoencoded embedding,” *Proceedings - International Conference on Pattern Recognition*, pp. 5145–5152, 8 2019.

[21] L. Mcinnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” 2020.

[22] D. P. Kingma and M. Welling, “Auto-encoding variational bayes.”

[23] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework,” 2016.

[24] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, and D. London, “Understanding disentangling in $\beta$-vae,” 4 2018.

[25] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, “Wasserstein auto-encoders.”

[26] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, “Variational deep embedding: An unsupervised and generative approach to clustering,” *arXiv preprint arXiv:1611.05148*, 2016.

[27] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, “Deep unsupervised clustering with gaussian mixture variational autoencoders.”

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[29] W. Harchaoui, P.-A. Mattei, and C. Bouveyron, “Deep adversarial gaussian mixture auto-encoder for clustering,” 2017.

[30] S. E. Chazan, S. Gannot, and J. Goldberger, “Deep clustering based on a mixture of autoencoders,” pp. 1–6, 2019.

[31] D. Zhang, A. Arbor, M. Y. S. T. L. Altos, C. B. E. A. S. Jose, and C. L. Balzano, “Deep unsupervised clustering using mixture of autoencoders.”

[32] P. T. [U+FFFD] A. Santos, D. Pedro, T. Tom´, T. Doutora, H. Aidos, D. Tereza, V. Vaz˜, V. Supervisor, . Doutor, P. Tomás, and T. Tomás, “A mixture-of-experts approach to deep image clustering estimating latent sizes and number of clusters electrical and computer engineering examination committee,” 2020.

[33] “hdbscan: Hierarchical density based clustering,” *Journal of Open Source Software*, vol. 2, p. 205, 3 2017.

[34] S. E. Chazan, S. Gannot, and J. Goldberger, “Deep clustering based on a mixture of autoencoders,” *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, 2019.

[35] D. Zhang, A. Arbor, M. Y. S. T. L. Altos, C. B. E. A. S. Jose, and C. L. Balzano, “Deep unsupervised clustering using mixture of autoencoders,” 12 2017.

[36] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *arXiv preprint arXiv:2002.05709*, 2020.

[37] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” *arXiv preprint arXiv:1911.05722*, 2019.

[38] X. Chen*, S. Xie*, and K. He, “An empirical study of training self-supervised vision transformers,” *arXiv preprint arXiv:2104.02057*, 2021.

[39] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” *arXiv preprint arXiv:2003.04297*, 2020.

[40] Q. Cheng, Q. Yuan, M. K. Ng, H. Shen, and L. Zhang, “Missing data reconstruction for remote sensing images with weighted low-rank tensor model.”

[41] “Framework for the treatment and reporting of missing data in observational studies: The tarmos framework,” *European Journal of Endocrinology*, vol. 183, pp. E7–E9, 4 2020.

[42] O. F. Ayilara, L. Zhang, T. T. Sajobi, R. Sawatzky, E. Bohm, and L. M. Lix, “Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry,” *Health and Quality of Life Outcomes*, vol. 17, pp. 1–9, 6 2019.

[43] P. Royston and I. R. White, “Multiple imputation by chained equations (mice): implementation in stata,” *Journal of statistical software*, vol. 45, pp. 1–20, 2011.

[44] R. Lall and T. Robinson, “The midas touch: Accurate and scalable missing-data imputation with deep learning,” *Political Analysis*, vol. 30, no. 2, p. 179–196, 2022.

[45] P. A. Mattei and J. Freiisen, “Miwae: Deep generative modelling and imputation of incomplete data,” *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 7762–7772, 12 2018.

[46] N. B. Ipsen, P.-A. Mattei, and J. Frellsen, “not-miwae: Deep generative modelling with missing not at random data,” *arXiv preprint arXiv:2006.12871*, 2020.

[47] Y. Gong, H. Hajimirsadeghi, J. He, M. Nawhal, T. Durand, and G. Mori, “Variational selective autoencoder,” pp. 1–17, 2019.

[48] S. Eduardo, A. Nazábal, C. K. I. Williams, and C. Sutton, “Robust variational autoencoders for outlier detection and repair of mixed-type data,” 2020.

[49] "Deep learning for clustering of multivariate clinical patient trajectories with missing values," *Giga-Science*, vol. 8, 11 2019.

[50] J. T. Chi, E. C. Chi, and R. G. Baraniuk, "k-pod: A method for k-means clustering of missing data," *http://dx.doi.org/10.1080/00031305.2015.1086685*, vol. 70, pp. 91–99, 1 2016.

[51] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: A review," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, p. 90–105, jun 2004.

[52] D. Pimentel-Alarcon, L. Balzano, R. Mareia, R. Nowak, and R. Willett, "Group-sparse subspace clustering with missing data," *IEEE Workshop on Statistical Signal Processing Proceedings*, vol. 2016-August, 8 2016.

[53] D. Pimentel, R. Nowak, and L. Balzano, "On the sample complexity of subspace clustering with missing data."

[54] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries."

[55] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *arXiv preprint arXiv:1509.00519*, 2015.

[56] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, 2015.

[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[58] C. K. Sø nderby, T. Raiko, L. Maalø e, S. r. K. Sø nderby, and O. Winther, "Ladder variational autoencoders," vol. 29, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/6ae07dcb33ec3b7c814df797cbda0f87-Paper.pdf

[59] "niuchuangnn/spice." [Online]. Available: https://github.com/niuchuangnn/SPICE

[60] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 3 1955.

[61] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, p. 846, 12 1971.

[62] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.