universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Deep Probabilistic Clustering
for Multi-View Data and Missing Data“

verfasst von / submitted by

Donatella Novakovic, BSc (WU)

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 921
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Informatik UG2002
Betreut von / Supervisor:	Ass.-Prof. Dipl.-Ing. Dr. Sebastian Tschiatschek

Acknowledgements

Throughout the writing of this thesis, I have received tremendous support. First, I would like to express my special gratitude to my supervisors Dr. Sebastian Tschiatschek and Dipl.-Ing Lukas Miklautz, who may not be listed officially but supported me at least as much. I appreciate both of your continuous feedback, your patience and your guidance. The countless and constructive discussions with you have made a significant contribution to this work.

In addition, I would like to acknowledge Dr. Claudia Plant for her encouragement and the opportunities I was given during this elaboration.

Also, I am grateful for my loving parents, who have always encouraged me, and my brother, who was the first to inspire me for computer science and I learned a lot from. And finally, thank you to my partner for all his support and love during this long process.

Contents

List of Tables

List of Figures

1. Introduction

Clustering is an important unsupervised learning task for exploratory data analysis and widely used in fields such as machine learning, pattern recognition or data mining. It has gained a lot of attention in the past years and still remains an active research field. The goal of clustering is to group data, which shares similar characteristics, into clusters. Commonly used clustering algorithms are k-means [9], DBSCAN [10] or Spectral Clustering (SC) [11]. The difference to supervised learning is the aim to discover natural groupings in unlabeled data [12]. Clustering facilitates the process of data exploration and data understanding. The clustering result typically depends heavily on the used data representation or, in other words, the choice of features. Thus, by learning a good representation of the data, we can better extract useful, and relevant information [13].

However, as the dimension of data increases, traditional clustering algorithms experience computational expense and poor performance. Therefore, dimensionality reduction has become a crucial technique to overcome this difficulty. It enables the projection of high-dimensional data to a lower-dimensional space by trying to preserve relevant information. Existing dimensionality reduction techniques, such as Principal Component Analysis (PCA) [14] or Linear Discriminant Analysis (LDA) [15] are often applied together with clustering, as in [16] or [17]. Moreover, Deep Neural Networks (DNNs) have become an attractive alternative for projecting high-dimensional data to a lower-dimensional space since they yield good representations, e.g., in [18, 19]. These previous works applied dimensionality reduction and clustering independently and sequentially. However, in recent years a lot of research has been conducted showing an improvement of performance by using these two methods in a joint manner as in [20, 21, 22].

Besides high-dimensional data, a second challenge we face is the heterogeneity of data, e.g., a single instance might come with features in the form of images, text, audio, or video [12]. Many deep clustering algorithms can be applied to one type of data but are not well suited for handling multiple data types. In this thesis, we use the term multi-view data to express that single instances can be represented by different data types. Since many real-world data sets consist of a mix of various features, multi-view clustering has

become a relevant topic in research as in [4, 23, 24]. One major advantage of multi-view data is that complementary and supplementary information from multiple views can be leveraged. Consequently, multi-view representations can improve the clustering performance [4, 25, 26]. Unfortunately, combining different views can be challenging when facing multiple data types because it demands the creation of a shared space between independent information and learning the importance of this information for the clustering task [4, 27].

The term *complex data* is frequently used in connection with unlabeled, high dimensional or multi-view data. However, another vital challenge is missing data. Especially when dealing with large amounts of data or multiple data views, we often face incomplete information. For example, companies lack information about their customers, e.g., age, customer numbers or complete addresses, because data was not collected [28]. While in recent years, numerous methods have been proposed for missing data imputation or high-dimensional and multi-view data clustering, there is hardly any research addressing those challenges jointly.

In this thesis, we aim to close this gap. We focus on developing Partial Multi-View Variational Deep Embedding (Partial MV-VaDE), a deep probabilistic clustering model for multi-view and missing data. We propose an approach combining different elements of state-of-the-art methods to tackle this challenge. The basis of our approach is formed by the existing deep probabilistic clustering model Variational Deep Embedding (VaDE) [3] initially designed for clustering single-view data. By extending VaDE's architecture with an additional Variational Autoencoder (VAE) and integrating the Partial VAE from [8], which uses a so-called *Pointnet Plus* (*PNP*) structure, we enable multi-view and missing data clustering.

The remainder of this thesis is structured as follows: Chapter 2 discusses the problem setting and defines the goals and contributions of this thesis. In Chapter 3, we provide the necessary background information on deep clustering, multi-view learning and missing data imputation, and explain essential preliminaries. Chapter 4 offers an extensive review of existing deep clustering approaches, including single-view and multi-view clustering methods, as well as generative and discriminative models. Also, we discuss several methods proposed for missing data imputation. The methodology of our approach is introduced in Chapter 5. Subsequently, we present the results of the conducted experiments in Chapter 6. Finally, Chapter 7 draws a conclusion and discusses possible future work.

2. Problem Setting

In this thesis, we focus on the following deep probabilistic clustering problem, which considers multi-view and missing data. We use a common notation convention throughout this thesis, where we denote vectors of values as lower case bold symbols, e.g., $\boldsymbol{v}$. To refer to the i^{th} position in $\boldsymbol{v}$, we use the notation v_i. Similarly to the vectors, we denote matrices as upper case bold symbols, e.g., $\boldsymbol{M}$.

We aim to develop and investigate models that can effectively cluster based on observed features from different modalities. For instance, we might be interested in clustering patients into healthy and sick, based on partially filled out health questionnaires and x-ray images. As pointed out in [8], assessing a patient's health status often requires several tests that can be not only time-consuming but costly. This includes the special case of all features of a specific modality being missing, for example cases where x-ray images are not available. Thus, as part of this thesis, we want to also consider the problem where an entire modality can be missing.

As it might not be possible to be certain about cluster assignments given only partially available data, we take a probabilistic approach for modeling the data. Generally, probabilistic clustering methods softly assign data points to clusters given a probability, which indicates the likelihood of a data point belonging to a cluster. However, given partial observations, the cluster assignments of unobserved data can be inferred given a subset of observed data as in [8]. As the set of observed data can have different sizes, consequently the inference requires to be able to handle arbitrary sets of observations [8].

Let $\boldsymbol{x}^{(1)} = [x_1^{(1)}, ..., x_{|\mathrm{N}|}^{(1)}]$ and $\boldsymbol{x}^{(2)} = [x_1^{(2)}, ..., x_{|\mathrm{M}|}^{(2)}]$ be two vectors with |N| and |M| random variables describing the same instance $\boldsymbol{x}$, where $\boldsymbol{x}^{(1)}$ denotes the first data view and $\boldsymbol{x}^{(2)}$ the second data view and the value of variables of each view can be queried: $x_j^{(1)}$ for $j \in N$ and $x_j^{(2)}$ for $j \in M$. Furthermore, analogous to [8], let each view have a subset of variables $\boldsymbol{x}_O^{(1)}$ and $\boldsymbol{x}_Q^{(2)}$ that are observed, where $O \subset N$, $Q \subset M$ and $\boldsymbol{x}_U^{(1)}$, $\boldsymbol{x}_V^{(2)}$ that are unobserved, where $U = N \setminus O$ and $V = M \setminus Q$, respectively. So, the sets of observed values can differ for $\boldsymbol{x}_O^{(1)}$ and $\boldsymbol{x}_Q^{(2)}$.

Similar to [3] and [7], we assume that data samples are generated from an unobserved

latent variable $\boldsymbol{z}$, which is distributed according to a prior probabilistic distribution, i.e., a Gaussian mixture, and belongs to a cluster c. We aim to maximize the likelihood of data samples $\boldsymbol{x}_O^{(1)}$ and $\boldsymbol{x}_Q^{(2)}$ and learn an estimation of the distributions' parameters, i.e., the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$, to group data samples generated from the same distribution to the same cluster c. Therefore, analogous to [8], we optimize a partial clustering objective depending solely on observed data samples $\boldsymbol{x}_O^{(1)}$ and $\boldsymbol{x}_Q^{(2)}$ from both views.

$$\begin{aligned}\mathcal{L}_{partial} = \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x}_O^{(1)},\boldsymbol{x}_Q^{(2)})}\big[\log p(\boldsymbol{x}_O^{(1)},\boldsymbol{x}_Q^{(2)}|\boldsymbol{z})\big] \\ - D_{KL}\big(q(\boldsymbol{z},c|\boldsymbol{x}_O^{(1)},\boldsymbol{x}_Q^{(2)}) \parallel p(\boldsymbol{z},c)\big)\end{aligned} \tag{2.1}$$

We aim to develop a deep clustering model that can effectively model partially observed data from different modalities. Thus, we study existing deep clustering methods, and investigate different approaches for multi-view clustering and missing data imputation. We focus on the existing deep probabilistic clustering model Variational Deep Embedding (VaDE) [3], and expect to learn if the model's architecture can be extended such that multiple data views can be handled. We examine an additive and multiplicative fusion technique to create a shared subspace between two data modalities and want to determine if these are suitable baseline methods for extending VaDE for multi-view data clustering. Moreover, we are interested if VaDE's clustering result can benefit from an additional view. To effectively deal with partially observed data in multiple data modalities, we investigate a state-of-the-art method for missing data imputation, the Partial Variational Autoencoder (VAE) with the *Pointnet Plus* (*PNP*) structure from [8]. We aim to learn if the integration of the Partial VAE into VaDE enables to effectively deal with missing data samples. As for readability and simplicity, we combine the methods' names and propose the Partial MV-VaDE to cluster multi-view and missing data.

The main contributions of our work can be summarized as follows:

- We study state-of-the-art deep clustering methods for single-view and multi-view data differentiating between generative and discriminative approaches and present an up-to-date taxonomy that can be used for further research.
- We propose a deep probabilistic clustering model, Partial MV-VaDE that can model partially observed data from two data modalities. To develop Partial MV-VaDE, we take the following steps:
  - We re-assess the performance of VaDE, which forms the basis of our approach and compare the clustering results using different pre-training models.

- We extend VaDE's architecture by another VAE to enable multi-view data clustering. To create a common subspace between two data modalities, we explore an additive and multiplicative fusion technique. The conducted experiments show that the additive approach can fit the purpose of MV-VaDE. We evaluate MV-VaDE on four multi-view data sets and compare the performance to several baselines and the clustering scores of three state-of-the-art deep multi-view clustering models.
- We integrate the Partial VAE with the *PNP* structure for each data view to effectively handle only partially observed data in both views. We evaluate the Partial MV-VaDE on several multi-view data sets for which we generate missingness and compare the clustering performance to one baseline and two state-of-the-art deep multi-view clustering models.
- Finally, we evaluate Partial (MV-) VaDE under different experimental conditions, where we generate various and higher amounts of missingness for multi-view data sets. To determine the model's capability of handling missing data, we investigate its changes in cluster probabilities given an increasing amount of unobserved data samples.

3. Background

Deep clustering has become an attractive solution to discover patterns and understand the underlying structure of unlabeled and high-dimensional data. However, we are interested in whether the architecture of Deep Neural Networks (DNNs) allows the integration of techniques for processing multi-view data and handling missingness.

In this chapter, we give a brief overview of relevant background material for understanding this thesis. First, some fundamental preliminaries regarding deep clustering are explained. We review two methods in more detail, namely the Variational Autoencoder (VAE) [7] and the Variational Deep Embedding (VaDE) [3], which form the basis of our approach. Next, we focus on multi-view deep clustering and discuss several fusion techniques, which enable the projection of multiple data views onto a common subspace representation. Finally, the attention is drawn to incomplete data, where we discuss the second building block of our approach, specifically the Partial VAE introduced by [8].

3.1. Deep Clustering

As outlined in [29], given large amounts of complex data, where only a few or no labels exist, applying supervised methods becomes more difficult and hardly possible. For this reason, unsupervised classification has become an essential method to process unlabeled data into similar groupings [29].

Numerous methods have been proposed, e.g., [3, 29, 30, 31], yielding good clustering performances by using DNNs and combining the tasks of dimensionality reduction and clustering. As explained in [29], the non-linear transformation can adjust the latent space based on the clustering result. Thus, clustering and representation learning can benefit from each other when performed simultaneously.

The following sections give a brief overview of necessary preliminaries regarding deep clustering models. Moreover, we review existing deep clustering approaches. One common way to distinguish these methods is to group them into discriminative and generative approaches [1]. While many well-performing discriminative algorithms have been proposed, e.g., [21, 30, 32], these methods are solely applicable to clustering tasks as their goal is to

learn a suitable decision boundary given observed data samples and the assumptions of the clustering model. Contrary, generative models such as [7, 33] aim to learn a probabilistic joint distribution of the underlying structure of the data, which enables the generation of new and similar data samples. For example, the Variational Autoencoder (VAE) [34] assumes that a latent variable is sampled from a Gaussian distribution described by mean and variance, allowing similar data sampling. The VAE and the generative deep clustering algorithm, named Variational Deep Embedding (VaDE) [3] are our main focus of interest.

3.1.1. Preliminaries

Generally, deep clustering algorithms comprise three essential components: (1) a deep neural network (DNN) to project high-dimensional data to a lower-dimensional latent representation, (2) a network loss to measure the similarity of input data and their reconstructions, and (3) a clustering loss to assign data samples to clusters. We will discuss these three elements in more detail and give some basic understanding of the deep clustering algorithm structure.

In surveys [1] and [2], the authors established a constructive taxonomy for deep clustering algorithms. We adopt their taxonomy and extend it with additional information, which we gained throughout our research.

Deep Neural Networks (DNNs) are the main building block of deep clustering algorithms. A DNN enables the projection of data to a latent representation, which subsequently can be used for clustering. Since clustering high-dimensional data becomes inefficient with the increasing number of features, dimensionality reduction is often required. Using a DNN, we can project high-dimensional data to a lower-dimensional space and try to capture important information.

During the research, we identified various models, e.g., the Autoencoder [6] or Variational Autoencoder [7], and differences in network architecture such as fully-connected or convolutional networks. In the following, we want to summarize the different architectures and briefly explain their purpose.

- The *Autoencoder* (AE) [6] is an unsupervised DNN, which aims to learn the reconstruction of input data. An AE consists of an encoder network and a decoder network. The encoder learns to effectively reduce data dimension by trying to capture important information. It transforms the input data into a latent representation by performing data compression. The decoder learns to reconstruct data from the latent space to be as close to the original input as possible. In this case, the latent

dimension is defined smaller than the input's dimensionality to prevent the model from learning an identity mapping.

- The *Stacked Autoencoder* (SAE) [35] is a DNN that consists of multiple layers of AEs. The model is trained layer-wise, and the output of each layer is connected to the input of the successive layer.
- The *Variational Autoencoder* (VAE) [7] is considered as a deep generative model. Similar to the AE, the VAE is composed of an encoder network and a decoder network. The difference is that the VAE encodes the input data as a distribution over the latent space from which we can sample a point in latent space.
- The *Generative Adversarial Network* (GAN) [33] consists of two competing networks, particularly a generator and a discriminator. The generator aims to learn a distribution to produce data samples, whereas the discriminator's goal is to distinguish between true and generated data samples.

As outlined in [1] and [2], these models can be based on different network architectures:

- The *Feed Forward Neural Network* (FFNN) or the *Multi-Layer Perceptron* (MLP) builds on multiple connected layers of neurons.
- The *Convolutional Neural Net* (CNN) inspired by the human visual cortex comprises several convolutional layers and subsampling layers, followed by one or more fully-connected layers.
- The *Deep Belief Network* (DBN) is a generative graphical model, which is composed of multiple Restricted Boltzmann Machines (RBMs).

The Network Loss can differ according to the used architecture. In the following, we will explain the network loss of the AE, VAE and GAN.

An AE uses the reconstruction loss to measure the decoder's performance in reconstructing the encoded data. Thus, a reconstruction $\hat{x}_i$ is compared to the original input x_i. An illustration of the AE and its objective is given in Figure 3.1. There exist several techniques to compute the reconstruction loss. Commonly used methods are:

- Cross-Entropy (CE) Loss, which is used for binary-valued data and can be expressed as:

$$\mathcal{L}_{CE} = -(x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)) \tag{3.1}$$

  where $x_i \in 0, 1$.

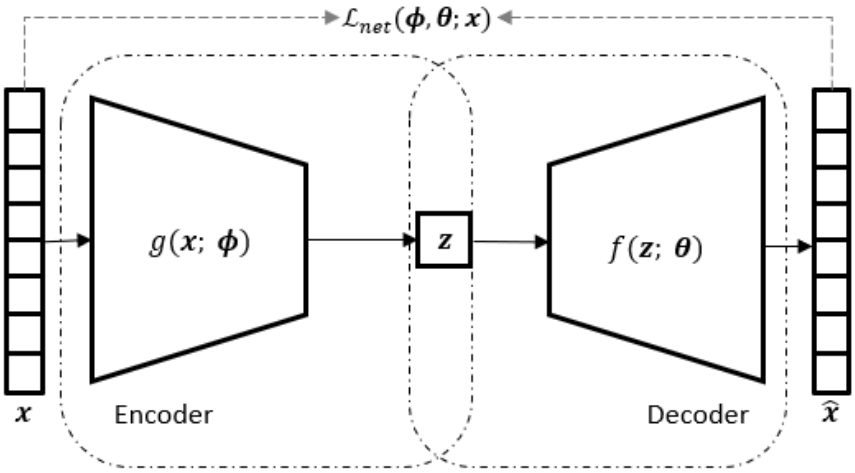


Figure 3.1.: Illustration of an AE, reproduced from [6].

- Sum-of-Squared Error (SSE) Loss, which can be used for real-valued input and can be defined as:

$$\mathcal{L}_{SSE} = \sum_{i=1}^{|\mathrm{K}|} (x_i - \hat{x_i})^2 \tag{3.2}$$

  where |K| is the number of samples.

- Mean-of-Squared Error (MSE) Loss, which is similar to the SSE and used for real-valued data.

$$\mathcal{L}_{MSE} = \frac{\sum_{i=1}^{|\mathrm{K}|} (x_i - \hat{x_i})^2}{|\mathrm{K}|} \tag{3.3}$$

Similar to the AE, a VAE benefits from the reconstruction term and combines it with the KL-Divergence to a variational loss. A more detailed description of the variational loss is given in Section 3.1.2.

Unlike the reconstruction and variational loss, a GAN actually uses two network losses based on a single distance measure between probability distributions. The so-called min-max loss [33] is derived from Cross-Entropy, where the Generator G aims at minimizing and the Discriminator D tries to maximize the loss function:

$$\min_G \max_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{x_i}[\log D(x_i)] + \mathbb{E}_{z_i}[\log(1 - D(G(z_i)))] \tag{3.4}$$

where $D(x_i)$ is the discriminator's probability estimate of x_i originating from the true data distribution and $\mathbb{E}_{x_i}$ is the expected value over all true data distributions. While

$G(z_i)$ is a data sample generated by G, which adds noise to the latent variables $\boldsymbol{z}$ and $\mathbb{E}_{z_i}$ is the expected value given all generated samples, $D(G(z_i))$ is the discriminator's probability estimate of $G(z_i)$ originating from the true data distribution.

The Clustering Loss supports the process of finding clusters for data. In [1], the authors distinguish between two types of frameworks based on the chosen cluster loss:

(1) Frameworks, where cluster assignments are computed directly during the network training, such as:

- The k-means loss is used in, e.g., [21] to minimize the intra-cluster variance.

$$\mathcal{L}_{kmeans} = \sum_{j=1}^{|\mathrm{C}|} \sum_{i=1}^{|\mathrm{K}|} \parallel x_i^{(j)} - r_j \parallel^2 \tag{3.5}$$

  Here r_j is the centroid of the j^{th} cluster and |C| is the number of total clusters.

- The cluster assignment hardening loss, e.g., the Kullback-Leibler (KL) Divergence measures the distance between an approximated distribution $q(x_i)$ and true distribution $p(x_i)$.

$$\mathcal{D}_{KL}(p \parallel q) = \sum_{i=1}^{|\mathrm{K}|} p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right) \tag{3.6}$$

  In [30], the KL-divergence between soft cluster assignments and an auxiliary distribution is minimized to simultaneously learn feature representation and cluster assignments.

- The agglomerative clustering loss applied in [32], aims to merge two clusters with maximum affinity in a step-wise manner until some defined stopping criterion is reached.

(2) Frameworks, using cluster regularization, which enforces the network to enhance a cluster structure. To obtain cluster assignments, additional clustering is required and performed on the learned data representation. Examples for cluster regularization losses are:

- The locality-preserving loss, e.g., in [36, 37], which enforces a suitable clustering representation by preserving the local neighborhood of latent variables $\boldsymbol{z}$. A simplified formulation of a locality-preserving loss can be expressed as follows:

$$\mathcal{L}_{loc} = \sum_{i,j \in \boldsymbol{k}_i} \mathcal{A}_{i,j} \parallel z_i - z_j \parallel^2 \tag{3.7}$$

where $\boldsymbol{k}_i$ is the set of k-nearest neighbors of a data sample x_i, and $\mathcal{A}$ represents a similarity measure.

- The group sparsity loss used in [37], where the latent space is divided into a number of groups that corresponds to the assumed number of clusters |C|. The idea is to determine relevant groups because these keep the intrinsic properties of data. Therefore, the groups are weighted according to their sizes, where larger groups are more relevant.

The Total Loss of deep clustering algorithms is often a combination of network loss and clustering loss such as in [20, 21, 29]. The optimizing objective can be expressed as follows [1]:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_{net} + (1 - \lambda)\mathcal{L}_{cluster} \tag{3.8}$$

where $\mathcal{L}_{net}$ and $\mathcal{L}_{cluster}$ indicate any network loss and any clustering loss, and λ can define a trade-off between structure preservation and cluster compression. The structure preservation indicates the degree of data reconstruction, whereas the cluster compression specifies the cluster separation. Unfortunately, this hyperparameter can be challenging to determine in an unsupervised setting. As outlined by [22], the task of learning data reconstruction can be conflicting with the task of clustering. However, the authors also explain, that by ignoring the reconstruction loss, the regularization of the embedded space is missing, which can lead to arbitrary clusterings. For this reason, many deep clustering approaches use a pre-training considering the reconstruction term only [3].

3.1.2. Variational Autoencoder (VAE)

The Variational Autoencoder (VAE), first defined in 2014 by [7] and [38], is a method, which jointly learns a generative model and the variational approximation. This approach has become attractive for various applications, e.g., unsupervised learning, representation learning, or generative modeling. This section aims to give some preliminary knowledge and understanding of VAEs. Definitions and explanations are adopted from the original papers [7, 38] and [34], which discusses the approach in more detail.

The VAE is a probabilistic model, which comprises a generative and an inference network. It assumes that data samples $\boldsymbol{x}$ are generated from an unobserved latent variable $\boldsymbol{z}$, which is sampled from a prior distribution. The joint probability distribution of the

VAE over the data samples $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$ can be defined as follows:

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}), \tag{3.9}$$

where the latent variables $\boldsymbol{z}$ are sampled from a prior $p(\boldsymbol{z})$, and the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ of $\boldsymbol{x}$ depends on the latent variables $\boldsymbol{z}$.

By this means, the generative process includes two significant steps, where (1) a latent variable $\boldsymbol{z}$ is sampled from a prior distribution $p(\boldsymbol{z})$, and (2) a value $\boldsymbol{x}$ is generated from the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$, which can be parameterized using a DNN. The key idea is to infer samples of the latent variables $\boldsymbol{z}$, which describe the observed data samples $\boldsymbol{x}$. However, the calculation of the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ is an intractable problem because of the evidence $p(\boldsymbol{x})$, which requires expensive computation. Consequently, a second posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$ is introduced, approximating the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$.

So, the VAE consists of two independently parameterized models: (1) the inference, or recognition model $q(\boldsymbol{z}|\boldsymbol{x})$, defined by the probabilistic encoder, and (2) the generative model $p(\boldsymbol{x}|\boldsymbol{z})$ represented by the probabilistic decoder. The probabilistic encoder $q(\boldsymbol{z}|\boldsymbol{x})$ aims to approximate the intractable true posterior $p(\boldsymbol{z}|\boldsymbol{x})$ by learning, given the data samples $\boldsymbol{x}$, a Gaussian distribution over possible values of $\boldsymbol{z}$, which describe a generated sample $\boldsymbol{x}$. Similarly, the probabilistic decoder $p(\boldsymbol{x}|\boldsymbol{z})$ creates a Gaussian distribution over possible values of $\boldsymbol{x}$ given the latent variables $\boldsymbol{z}$. Figure 3.2 visualizes the architecture of the VAE, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ define the means and variances of the prior distribution from which a single latent variable $\boldsymbol{z}$ can be sampled by using the *reparameterization trick* [7]: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma^2} * \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}$ is an auxiliary noise variable that follows a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$.

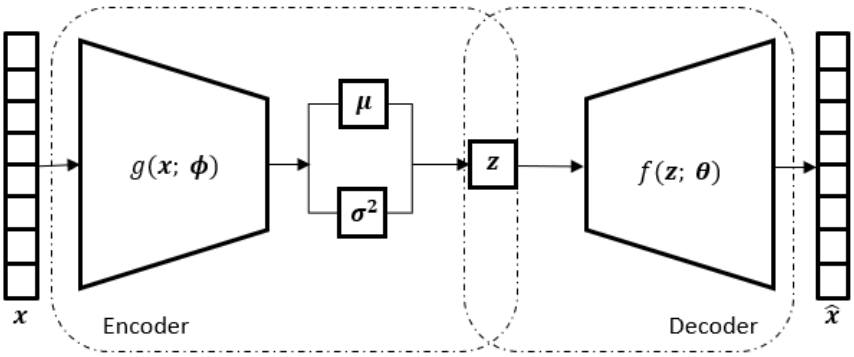


Figure 3.2.: Illustration of the VAE, reproduced from [7].

The models' objective is to maximize the evidence lower bound (ELBO), also called the variational lower bound:

$$\mathcal{L}_{ELBO}(\boldsymbol{x}) = E_{q(\boldsymbol{z}|\boldsymbol{x})}\big[\log p(\boldsymbol{x}|\boldsymbol{z})\big] - D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\big) \tag{3.10}$$

While maximizing the ELBO approximately maximizes the marginal likelihood $p(\boldsymbol{x})$, it also minimizes the KL-divergence between the approximated posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$ and the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. As the goal is to maximize the ELBO with respect to both, the variational and generative parameters, and a standard *Monte Carlo* gradient estimator results in high variances, the authors introduce the Stochastic Gradient Variational Bayes (SGVB) estimator [7]. The SGVB estimator is yield by using the *reparameterization trick* on the variational lower bound and can be optimized straightforwardly with a stochastic optimization method, e.g. Stochastic Gradient Descent (SGD).

3.1.3. Variational Deep Embedding (VaDE)

The deep clustering framework Variational Deep Embedding (VaDE) [3] forms the main building block in this thesis. Following the descriptions and definitions of [3], this section explains the methodology of the model.

VaDE is an unsupervised generative clustering model, which combines the VAE and the Gaussian Mixture Model (GMM). Hence, the model can capture the statistical structure of data and generate realistic data samples. Analogous to the VAE, VaDE assumes that data samples $\boldsymbol{x}$ are generated from an unobserved latent variable $\boldsymbol{z}$. Here, $\boldsymbol{z}$ is sampled from a cluster c, which is picked from a prior distribution. The joint probability of VaDE over the data samples $\boldsymbol{x}$, latent variables $\boldsymbol{z}$ and a cluster c is defined as follows:

$$p(\boldsymbol{x}, \boldsymbol{z}, c) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}|c)p(c) \tag{3.11}$$

where the cluster c is chosen from a prior $p(c)$. The likelihood $p(\boldsymbol{z}|c)$ of the latent variables $\boldsymbol{z}$ depend on c, and the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ of the data samples $\boldsymbol{x}$ depend on $\boldsymbol{z}$. The process of generating data samples $\boldsymbol{x}$ distinguishes for binary data implicating a multivariate Bernoulli distribution and real-valued data, where $\boldsymbol{x}$ are sampled from a Gaussian distribution. The steps of the generative process can be summarized as follows:

1. Cluster selection in $p(c)$: $c \sim Cat(\boldsymbol{\pi})$
2. Latent variable sampling in $p(\boldsymbol{z}|c)$: $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \boldsymbol{I})$
3. Decoding to observation in $p(\boldsymbol{x}|\boldsymbol{z})$: $\boldsymbol{x} \sim Ber(\boldsymbol{\mu}_{\boldsymbol{x}})$ or $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_{\boldsymbol{x}}^2 \boldsymbol{I})$

In the first step, a cluster c is chosen given its prior probability $\boldsymbol{\pi}$, where $Cat(\boldsymbol{\pi})$ indicates its categorical distribution. $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ denote the mean and variance of the Gaussian distribution corresponding to the cluster c, and $\boldsymbol{I}$ is an identity matrix. Accordingly, a latent variable $\boldsymbol{z}$ can be selected from the cluster's Gaussian distribution in the second step. A DNN $f(\boldsymbol{z};\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ is used on the latent variables $\boldsymbol{z}$ and computes the expectation for the mean and variances of the observed data samples $\boldsymbol{x}$ denoted as $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x^2$, respectively. Finally, $Ber(\boldsymbol{\mu}_{\boldsymbol{x}})$ or $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_x^2\boldsymbol{I})$ can represent the multivariate Bernoulli or the Gaussian distribution from which binary or real-valued data samples $\boldsymbol{x}$ can be selected.

VaDE aims at optimizing the likelihood of the data samples $\boldsymbol{x}$ by inferring representative samples of the latent variables $\boldsymbol{z}$. As already discussed in Section 3.1.2 and explained by [7], the calculation of the true posterior $p(\boldsymbol{z}, c|\boldsymbol{x})$ is expensive. For this reason, a second posterior distribution (or variational posterior) $q(\boldsymbol{z}, c|\boldsymbol{x})$ is introduced to approximate the true posterior $p(\boldsymbol{z}, c|\boldsymbol{x})$. The approximation is realized by employing another DNN $g(\boldsymbol{x};\boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$. By this means, the evidence lower bound (ELBO) can be maximized using the Stochastic Gradient Variational Bayes (SGVB) [7] estimator and the *reparametrization trick* [7]. As derived in [3], the $\mathcal{L}_{ELBO}(\boldsymbol{x})$ can be expressed as follows:

$$\begin{aligned}\mathcal{L}_{ELBO}(\boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z},c)}{q(\boldsymbol{z},c|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x})}\big[\log p(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}|c) + \log p(c) - \log q(\boldsymbol{z}|\boldsymbol{x}) - \log q(c|\boldsymbol{x})\big]\end{aligned} \tag{3.12}$$

Further, the ELBO can be rewritten as:

$$\mathcal{L}_{ELBO}(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x})}\big[\log p(\boldsymbol{x}|\boldsymbol{z})\big] - D_{KL}\big(q(\boldsymbol{z},c|\boldsymbol{x}) \parallel p(\boldsymbol{z},c)\big) \tag{3.13}$$

where the first expression is considered the reconstruction term quantifying how well data is represented. The second term is the KL-divergence between the prior $p(\boldsymbol{z}, c)$ and the variational posterior $q(\boldsymbol{z}, c|\boldsymbol{x})$. This term enforces the network to enhance a cluster structure.

Figure 3.3, which is reproduced from [3], shows a simplified illustration of VaDE's data generative process. The probabilistic decoder denoted by $f(\boldsymbol{z};\boldsymbol{\theta})$ yields a distribution over possible data samples $\boldsymbol{x}$ given the latent variables $\boldsymbol{z}$. The latent variables $\boldsymbol{z}$ are generated based on c, which is picked from a GMM. Contrary, the probabilistic encoder $g(\boldsymbol{x};\boldsymbol{\phi})$ produces a Gaussian distribution over possible values of latent variables $\boldsymbol{z}$ from a

cluster c, given the data samples $\boldsymbol{x}$.

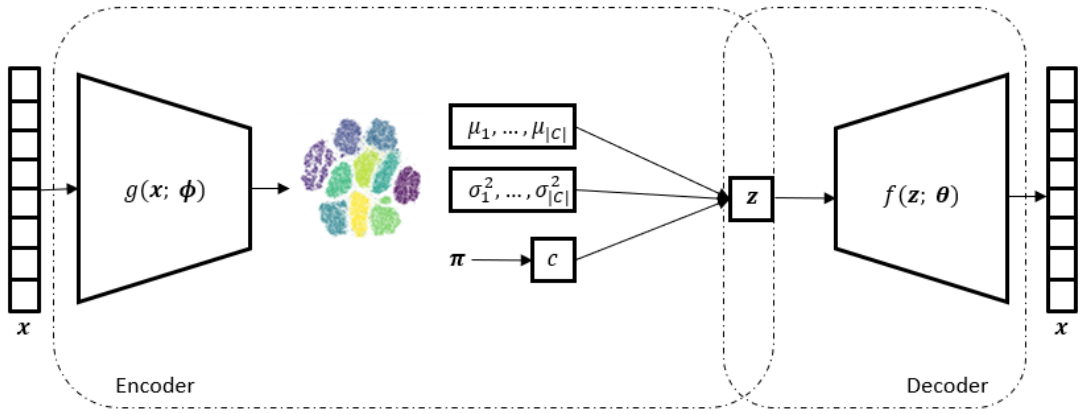


Figure 3.3.: Illustration of the VaDE architecture, reproduced from the original paper [3].

3.2. Multi-View Deep Clustering

In many real-world applications, including healthcare, finance or marketing, we face diverse data, including images, text, video or audio. In this thesis, we use the term multi-view data to summarize different data types describing a single instance. For example, we can describe an orange in multiple ways. We can use an image or describe the fruit by its features such as taste, color, calories, or even shape. All of these are different views of the same object. In the literature, there exist other terms like mixed data, multi-modal data or heterogeneous data, which we summarize under the umbrella term multi-view data.

As discussed in [5, 20], one substantial advantage of multi-view data is that observing data from different viewpoints offers complementary and supplementary information. This gain in information can support finding intrinsic patterns in data and improve the performance of clustering tasks.

Few efforts have been made studying deep clustering algorithms for multi-view data, e.g., in [23, 39, 40]. Clustering multi-view data is a complex task, which requires creating a shared subspace between multiple data views and at the same time identifying the importance of different views on the clustering result. As for deep learning, a DNN is often used to extract modality-specific features for example in [4, 24]. Multi-modal representation learning is used to integrate these features into a common subspace.

There exist various approaches for multi-modal representation learning, which are

extensively discussed in [20]. Generally, simple baseline methods such as fusion by addition, concatenation or multiplication can be used to create a joint representation of multiple data views.

In the context of deep clustering some more suitable methods were proposed. For example, [23, 39] use Canonical Correlation Analysis (CCA) to learn a nonlinear transformation of two data views whose correlation is maximized. The multi-modal representation in [4] is created by the sum of affinity matrices which are weighted based on each view's estimated confidence. Contrary, in [24] the authors introduce a self-expressive layer forcing the modalities to share the same affinity matrix.

3.3. Missing Data Imputation

Another frequent problem in real-world applications is missing data. For example, an online shop can lack information about users, e.g., age, gender, or orders, because data were not collected correctly or due to system errors [28]. Also, medical records may contain incomplete information about the patient's health status [8, 28]. Unfortunately, acquiring more data is not only time-consuming but costly. However, to make more suitable recommendations for customers or to give a clinical diagnosis, additional knowledge is often required for better understanding and decision-making [8].

Several approaches addressing missing data imputations, e.g., [28, 41], have taken advantage of the VAE and amortized inference. We focus on a more recent approach, the Partial VAE presented by [8] to predict missing data samples, and discuss this method in the following section.

3.3.1. Partial VAE

The Partial VAE [8] is the second building block used in this thesis. Definitions and explanations are used from the original paper [8].

The Partial VAE is an extension of the VAE, which can be used for missing data imputation. It addresses the challenge, where a set of random variables $\boldsymbol{x} = [x_1, ..., x_{|\mathrm{N}|}]$ can be divided into a subset of observed $\boldsymbol{x_O}, O \subset N$ and unobserved $\boldsymbol{x_U}, U = N \setminus O$ variables. Also, the set can contain arbitrary partitions of $\{U, O\}$, hence the set size of the observed variables can differ. As the variational lower bound and the amortized inference cannot be applied for VAE training given differently sized sets of observations, the authors extend the amortized inference to handle partially observed data in the Partial VAE.

The data generation $p(\boldsymbol{x}|\boldsymbol{z})$ in a VAE is factorized:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_j p_i(\boldsymbol{x_j}|\boldsymbol{z}) \tag{3.14}$$

For this reason, observed data samples $\boldsymbol{x_O}$ and unobserved data samples $\boldsymbol{x_U}$ can be considered conditionally independent when given the latent variables $\boldsymbol{z}$, and consequently the inferences about $\boldsymbol{x_U}$ can be reduced to the inference about $\boldsymbol{z}$ [8]:

$$p(\boldsymbol{x_U}|\boldsymbol{x_O}, \boldsymbol{z}) = p(\boldsymbol{x_U}|\boldsymbol{z}) \tag{3.15}$$

As a result, $\boldsymbol{x_U}$ can be estimated from $\boldsymbol{z}$, given the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x_O})$. The authors introduce a partial inference network $q(\boldsymbol{z}|\boldsymbol{x_O})$ to approximate the true posterior $p(\boldsymbol{z}|\boldsymbol{x_O})$ and express the partial variational lower bound, which solely depends on $\boldsymbol{x_O}$, as follows:

$$\begin{aligned} \log p(\boldsymbol{x_O}) &\geq \log p(\boldsymbol{x_O}) - D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x_O}) \parallel p(\boldsymbol{z}|\boldsymbol{x_O})\big) \\ &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x_O})}\big[\log p(\boldsymbol{x_O}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{x_O})\big] \\ &\equiv \mathcal{L}_{partial} \end{aligned} \tag{3.16}$$

To handle parameter-sharing across arbitrary sized sets of observed variables, the authors propose to extend the amortized inference by introducing a set function, which is invariant to permutations, and define $q(\boldsymbol{z}|\boldsymbol{x_O})$ as an encoding:

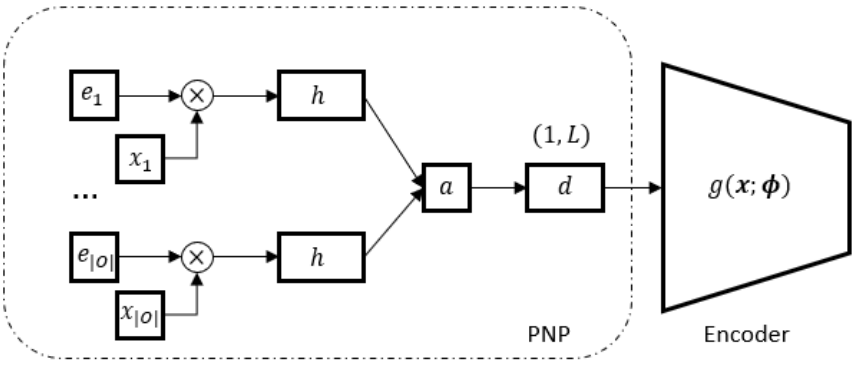


Figure 3.4.: Illustration of the Partial VAE encoder reproduced from the original paper [8].

$$\boldsymbol{d}(\boldsymbol{x_O}) \colon = a(h(s_1), h(s_2), ..., h(s_{|O|})), \tag{3.17}$$

where s_j contains the information of the input variable x_j and the corresponding identity variable e_j, which is an unknown embedding optimized during training and $|\mathrm{O}|$ is the number of observed variables. The variable s_j is constructed by element-wise multiplication $s_j = e_j * x_j$, referred to as the *Pointnet Plus* (*PNP*) [8] setting. The neural network $h(\cdot)$ maps s_j to the latent space with dimension L, and $a(\cdot)$ performs a permutation invariant summation operation. The resulting encoding $\boldsymbol{d}$ with latent dimension L serves as input to the probabilistic encoder network of a VAE. Figure 3.4 reproduced from [8], illustrates the Partial VAE encoder for a better understanding.

4. Related Work

Various deep clustering techniques have been proposed targeting high-dimensional and unlabeled data sets. Few efforts have been made to study deep clustering for multi-view data. In the following sections, we discuss several state-of-the-art methods for deep clustering. We study discriminative models and generative models for both single-view and multi-view data. Finally, we review different methods to predict missing data.

4.1. Discriminative Deep Clustering Models for Single-View Data

One of the first well-known deep clustering methods is Deep Embedded Clustering (DEC) [30]. DEC is used for performance comparison by numerous other works. The method works in a two-stage process, where first, an AE is used to learn a meaningful data representation. Secondly, the clustering objective minimizes the KL-divergence between soft assignments and auxiliary target distribution. Both soft assignments and centroids are updated in an alternating process.

Similar to DEC, [5] present Discriminatively Boosted Clustering (DBC), a unified image clustering framework. DBC adopts the characteristics of DEC. However, the difference between these two models is that DBC uses a CNN. The authors show that DBC outperforms DEC on image data sets.

Almost simultaneously with DEC, [32] introduce the novel method Joint Unsupervised Learning (JULE). JULE uses a CNN combined with an agglomerative clustering approach, where clusters are merged based on an affinity matrix. The model is optimized by alternating between two steps: (1) Cluster assignments are updated in the forward pass, while subsequently, (2) parameters are updated in the backward pass. Contrary to DEC and DBC, JULE does not require any pre-training process.

In [42], an unsupervised discrete representation learning method, called Information Maximizing Self-Augmented Training (IMSAT) is proposed. IMSAT uses data augmentation to model the invariance of learned data representations. An MLP is used to map data into a discrete representation. During this process, predictions are regularized to be close

to the original data. The authors define this process of regularization as *Self-Augmented Training (SAT)*. SAT is combined with the Regularized Information Maximization (RIM) for clustering, where information-theoretic dependencies between inputs and predictions are maximized.

Deep Adaptive Clustering (DAC) [43] is an image clustering framework based on a CNN. The authors transform the clustering problem into a so-called binary pairwise classification model, where DAC assumes that each pair of images belongs to the same cluster or a different cluster. Images are represented by label features, whose similarity is measured using cosine distance. By introducing a clustering constraint, the learned label features are transformed to one-hot vectors and subsequently can be used for clustering. The authors integrate an adaptive learning algorithm since the ground truth similarities are unknown.

Clustering Convolutional Neural Network (CCNN) [44] solves clustering and feature learning iteratively, where first k-means is applied on extracted features to obtain cluster assignment. Secondly, the CNN updates its parameters and the cluster centroids. To reduce computation and memory costs, the authors integrate the mini-batch k-means algorithm. Their experiments show high scalability compared to other clustering methods such as DEC.

The common ground of the discussed models is that the network is constraint by a clustering loss only. The methods DEC and DBC use an AE loss to learn a meaningful data representation, but this step is conducted independently from the clustering task. We refer to this approach as two-staged because a pre-training and fine-tuning step is performed, each optimizes a different objective function.

However, [32] presents the first method with a combined loss. The Deep Clustering Network (DCN) uses joint dimensionality reduction and clustering. DCN is based on an AE and combined with the k-means loss. The authors propose an optimization criterion comprising three elements: (1) dimensionality reduction, (2) data reconstruction, and (3) cluster assignment. DCN is optimized using alternating Stochastic Gradient Descent (SGD), where the network parameters, assignments, and centroids are updated.

Introducing a combined loss can lead to a conflict between the reconstruction and the clustering objective. Several deep clustering models attempt to balance this trade-off with a hyperparameter, which can be hard to tune. However, [22] propose the framework: Autoencoder Centroid based Deep-Clustering methods (ACe/DeC) without the need of a hyperparameter. ACe/DeC learns a latent representation of two separate spaces, a clustering and a shared space. This is accomplished by splitting the clustering loss

into an objective that preserves the cluster structure and another that contains shared information. The authors show that the integration of ACe/DeC into DCN can achieve better clustering results. Also, using a centroid-based objective makes it applicable to numerous centroid-based methods.

[5] presents the Deep Subspace Clustering Networks (DSC-Nets), which integrates a self-expressive layer between encoder and decoder. The self-expressive layer enables to learn an affinity matrix which is further used for subspace clustering. The loss function during training combines the reconstruction loss and a regularization loss for the non-linear mapping of the data.

In [45], it is claimed that models such as DSC-Nets suffer from a weak latent representation and, therefore, limit real-world data applicability. The authors propose Distribution Preserving Subspace Clustering (DPSC), which minimizes the KL-divergence between the original data distribution and the latent distribution. Thus, in addition to the reconstruction and self-expressive term used in DSC-Nets, DPSC introduces the distribution consistency loss. The authors show that this additional cluster objective improves the model's performance.

The Deep Manifold Clustering (DMC) model is proposed in [36] and aims to solve unsupervised Multi-Manifold Clustering (MMC) problems. DMC optimizes a joint loss function, which comprises two objectives: (1) A locality-preserving objective is minimized, ensuring that structure-preserving representations are learned. (2) A clustering-oriented constraint is introduced aiming at extracting cluster-specific representations. Given the transformed representation, the k-means algorithm is performed for further clustering.

In [46] a new deep embedded regularized clustering (DEPICT) model is proposed to create an efficient data-subspace mapping and accurate cluster assignments. DEPICT brings two key aspects: First, a soft-max function is integrated to predict probabilistic assignments optimized with KL-divergence. Secondly, a regularization term ensures the balance of assignments. The clustering and auxiliary reconstruction loss functions are combined to a unified objective and optimized using an alternating approach.

The Improved Deep Embedded Clustering (IDEC) [20] model is based on the ideas of DEC. The difference between these two methods lies in formulating the objective function. While DEC works in a two-staged phase and optimizes the reconstruction term and clustering loss separately, IDEC incorporates a joint objective because the authors claim that a joint optimization is essential for the clustering result. Therefore, DEC keeps the decoder part of the AE, such that the model can perform clustering and learn a meaningful representation at the same time. Their conducted experiments show that IDEC outperforms DEC.

Also, the Spectral Clustering with Deep Embedding (SCDE+) [47] model uses DEC as a basis and combines it with SC. Essentially, the architecture of DEC is used to learn a data representation, where SC is subsequently performed for clustering. As the SC algorithm requires prior knowledge of the number of clusters, the authors integrate a soft-max layer for estimation.

Whereas the previously discussed methods consider one single valid clustering, [29] argue that real-world data can be partitioned in various valid ways. Recently, the authors proposed a novel model called Deep Embedded Non-Redundant Clustering (ENRC), which is the first to combine deep learning with non-redundant clustering. The ENCR model consists of a non-redundant clustering layer between the encoder and decoder network of an AE. While a linear transformation matrix aligns the structures within each clustering, feature weights serve as a soft separation mechanism of the space. By this means, the model can find multiple valid non-redundant clusterings in the embedded space.

4.2. Generative Deep Clustering Models for Single-View Data

Since discriminative models primarily serve the purpose of clustering tasks, generative models have become a compelling and efficient alternative. One of the commonly used approaches in this research area are the VAE and GAN, which serve as a basis for the clustering task.

Deep Clustering with VAEs

The Variational Deep Embedding (VaDE) [3] was one of the first unsupervised generative clustering approaches to combine the VAE with a GMM. Details on VaDE were discussed in Section 3.1.3.

Similarly to VaDE, [48] also combines the VAE with a GMM and presents the Gaussian Mixture VAE (GMVAE). The difference between those two methods lies in the formulation of the generative process. Contrary to VaDE, GMVAE introduces an additional variable sampled from a Gaussian distribution. Subsequently, the cluster probability can be chosen from the introduced distribution.

More recently, the Deep Clustering via a Gaussian mixture VAE with Graph Embedding (DGG) [49] model was proposed. DGG uses graph embeddings to preserve the local structure of the data. While VaDE learns the distribution of latent features independently, DGG enforces the connected distribution to become closer and defines a similarity graph.

Model	Network	Architecture	$\mathcal{L}_{net}$	$\mathcal{L}_{cluster}$	Objective training	Methods
Discriminative	DNN	MLP	-	Assignment loss	$\mathcal{L}_{cluster}$	IMSAT
		CNN	-	Assignment loss	$\mathcal{L}_{cluster}$	JULES DAC CCNN
	AE	MLP	AE-loss	Assignment loss	2-staged	DEC
					Joint	DCN ACe/DeC IDEC ENCR
				Regularization loss	Joint	DMC
					2-staged	SCDE+
		CNN	AE-loss	Assignment loss	2-staged	DBC
					Joint	DEPICT
				Regularization loss	Joint	DSC-Nets DPSC
Generative	VAE	MLP	VAE-loss	Assignments loss	Joint	VaDE GMVAE DGG
	Adversarial AE	MLP	AE-loss	Assignments loss	Joint	DAC
	GAN	MLP	GAN-loss	Assignments loss	Joint	DASC CatGAN
		CNN	GAN-loss	Assignments loss	Joint	ClusterGAN
				Regularization loss	Joint	InfoGAN

Table 4.1.: Summary of existing deep clustering methods for single-view data, which can be considered an extension of [1, 2] including more recently proposed approaches.

The authors claim that the combination of graph information and GMM lead to more accurate clustering results. The objective function minimizes the Jenson-Shannon (JS) divergence [50] between data samples represented as nodes and their posterior distribution.

Deep Clustering with GANs

The Deep Adversarial Clustering (DAC) [51] model is based on the idea of adversarial AEs and combined with a GMM to enable a suitable clustering space. The AE consists of a GM-based random generator and an DNN-based adversarial discriminator, which enforces the latent representation to follow the GM prior distribution. DAC aims at optimizing three objectives using SGD, a reconstruction loss, the GMM likelihood and an adversarial loss.

In [52], the Categorical Generative Adversarial Networks (CatGAN) model is proposed. Contrary to standard GANs which learn a binary discriminative function, the GAN model used in CatGAN is extended to multi-class tasks. As a result, the discriminator does not predict the probability of a single sample belonging to the true distribution but classifies all data samples into a defined number of classes. At the same time, the generator creates samples for all classes. The objective function consists of (1) a discriminator loss, where mutual information between observed examples and their class distribution is optimized and (2) a generator loss increasing the classifier's robustness.

Similarly to CatGAN, [53] proposes a deep clustering approach based on a GAN, called Information Maximizing Generative Adversarial Networks (InfoGAN). The difference to CatGAN is that InfoGAN combines a variational regularizer, which ensures high mutual information between latent variables and the generator's distribution with the min-max loss.

A different generative clustering approach is proposed by [31], named the ClusterGAN. The authors replace the discriminator with an encoder network and integrate a clustering specific loss which enables clustering in the latent space.

Deep Adversarial Subspace Clustering (DASC) [54] combines adversarial learning with subspace clustering. DASC's generator consists of three components: (1) an AE learning the data representation, (2) a self-expressive layer producing an affinity matrix for clustering and (3) a sampling layer generating real and fake data samples for evaluating the subspace quality.

Table 4.1 summarizes the existing deep clustering methods for single-view data and can be considered an extension of [1, 2], including more recent approaches. Following [1, 2], the methods are grouped by their components, which were explained in Section 3, such as (1) the model type distinguishing between discriminative and generative approaches, (2) the network grouping methods that are based on, e.g., an AE, VAE or GAN, respectively, or (3) the architecture, MLP versus CNN. Moreover, we include some information about the methods' objectives and optimization procedure. We group these methods by their network loss and cluster loss. As for the cluster loss, we broadly differentiate between a cluster assignment loss, where cluster assignments are computed directly during the network training and a cluster regularization loss, which are objectives that enforce the network to enhance a cluster structure. The optimization procedure can be distinguished by using (1) a cluster loss solely, (2) a two-staged model, where first a reconstruction term is optimized and subsequently a clustering objective, or (3) a joint training approach, where network loss and cluster loss are optimized as one total loss.

4.3. Deep Clustering Models for Multi-View Data

One of the earliest approaches of deep clustering targeting multi-view data is Deep Canonical Correlation Analysis (DCCA) [39]. DCCA consists of two MLPs and uses CCA to learn nonlinear mappings of two data views. The parameters of both transformations are learned jointly to maximize the total correlation.

The DCCA model is extended to Deep Canonically Correlated Autoencoders (DCCAE) in [23]. Instead of a simple MLP, the authors propose using two AEs and adding a reconstruction loss to improve the model's performance.

Both models, the DCCA and the DCCAE use a sequential learning approach, where further clustering is required to obtain cluster assignments. For example, k-means can be performed on the shared data representation obtained by the models. Moreover, both approaches are restricted to two data views.

However, in [55], the Multi-view Clustering via Deep Matrix Factorization (MvC-DMF) is proposed. MvC-DMF uses semi-non-negative matrix factorization for learning hierarchical semantics of more than two data views. By this means, distances between samples of the same cluster are minimized layer-wise. A graph regularizer is integrated, preserving geometric structures in each view. The model is optimized in an alternating approach such that SC can be applied to obtain the cluster assignments.

A different approach is introduced by [4], called the MultiSpectralNet (MvSN) [4]. MvSN combines Siamese nets [56] with SC for multi-view clustering, where first, each view is trained independently. Subsequently, a multi-modal representation is created by the sum of affinity matrices weighted based on each view's estimated confidence. Further, to obtain cluster assignments, k-means is applied to the shared representation.

Also, in [57], the authors present a method combining DNNs with SC, the Multi-view Spectral Clustering Network (MvSCN). MvSCN aims at optimizing two objectives, a so-called within-view and between-view similarity. Similarly to MvSN, Siamese nets [56] are used to learn an affinity matrix for each view. Each view is mapped to the embedded space given the obtained affinity matrices. Contrary to MvSN, an extra layer is added, performing an orthogonal transformation on the input data and projects view-specific features into a shared space. Again, k-means is applied to compute the cluster assignments.

[24] introduce one of the first methods using a GAN for multi-view clustering, named the Deep Adversarial Multi-view Clustering (DAMC) model. DAMC consists of four elements: (1) a multi-view denoising encoder projecting each view to a lower-dimensional space, (2) a multi-view denoising generator creating data samples, (3) one discriminator for each view, distinguishing generated from true data distributions, and finally (4) a deep

Model	Network	Architecture	$\mathcal{L}_{net}$	$\mathcal{L}_{cluster}$	Objective training	Methods
Discriminative	DNN	MLP	-	Regularization loss	$\mathcal{L}_{cluster}$	DCCA MvC-DMF
	SiameseNet	MLP	-	Regularization loss	$\mathcal{L}_{cluster}$	MvSN MvSC
	AE	MLP	AE-loss	Regularization loss	Joint	DCCAE
Generative	GAN	MLP	GAN-loss	Assignments loss	Joint	DAMC

Table 4.2.: Summary of existing deep clustering methods for multi-view data.

clustering layer, minimizing the KL-divergence between true and generated distributions. Overall, the model is optimized using a joint loss function, including a reconstruction term, the GAN loss and a clustering loss.

Table 4.2 gives an overview of discussed deep clustering methods targeting multi-view data. Compared to the number and variety of existing deep clustering approaches for single-view data, there still exists research potential for deep clustering multi-view data as existing methods mainly integrate objectives for cluster regularization. While cluster regularization losses often require further clustering to obtain cluster assignments, DAMC [40] is the only method that proposes a cluster assignment loss. Generally, objectives, where network and cluster losses are optimized jointly could be studied further.

4.4. Methods for Missing Data Imputation

[41] and [28] are one of the first to use the advantages of the VAE and amortized inference to manage missingness in data sets. As stated in [8], particularly for partially observed data, it has become an attractive technique because of the speed requirement in various real-life use-cases.

In [41], amortized inference is applied during training with complete data. The pre-trained model is then used to infer missing data entries in the partially observed data sets.

Since this method does not apply to many real-world data sets, [28] proposes Zero Imputation (ZI), a technique where missing data is imputed with zeros and fed to the inference network. ZI is used for both training and test set with missing data points. However, a disadvantage of ZI is that it struggles with biases introduced from data sets, where values are not missing entirely at random.

A more recent approach is presented by [8], called Efficient Dynamic Discovery of high-value information (EDDI). The framework comprises the discussed Partial VAE,

which predicts missing data points, and an acquisition function, which maximizes the expected information gain. A detailed description of the methodology of the Partial VAE can be found in the previous Section 3.3.1. The advantage is that the Partial VAE can handle arbitrary sets of observed data and shares parameters across different sized sets.

Whereas EDDI serves as an efficient solution given enough training data, a novel inference method is proposed in [58], the Bayesian Deep Latent Gaussian Model (BELGAM). The authors address the ice-start problem, which targets the deployment of machine learning tasks given the minimum available amount of data. BELGAM can quantify uncertainty and uses it to acquire unfamiliar but informative data to reduce the cost of acquisition.

5. Methodology

This section proposes an extension of the VaDE [3] model to cluster multi-view data sets containing missing data features. Therefore, we expand the model's architecture to exploit two different data views. Furthermore, we integrate the Partial VAE model in the *PNP* setting [8] to enable the handling of missing features. We refer to the proposed model as Partial MV-VaDE.

The suggested approach motivates the study of deep generative clustering methods targeting real-world data, i.e., multi-view and missing data. Moreover, Partial MV-VaDE gives an idea of how existing single-view clustering methods and techniques for missing data imputation can be combined to solve a more complex challenge.

5.1. Network Architecture

Given two data views $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ describing the same instance $\boldsymbol{x}$, where each view can have a different dimension, we extend VaDE's network with an additional VAE to a multi-view inference model (probabilistic MV-encoder) and a multi-view generative model (probabilistic MV-decoder). The probabilistic MV-encoder and probabilistic MV-decoder comprise fully connected networks for each data view $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. The probabilistic MV-encoders aims to learn a latent embedding separately for $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ based on important modality-specific features. Using this setting we can handle views with different data dimensions.

As this work concentrates on solving the clustering task for exclusively two data views, the model is restricted to this number of modalities. However, the further extension of the network architecture to more than two data views is straightforward and can be considered future work.

To handle missing features $\boldsymbol{x}_U^{(1)}$, $\boldsymbol{x}_V^{(2)}$, we integrate the Partial VAE architecture in the *PNP* setting from [8] separately for each view. As discussed in Section 3.3.1 and extensively studied by [8], observed variables $\boldsymbol{x}_O^{(1)}$, $\boldsymbol{x}_Q^{(2)}$ in a VAE can be considered conditionally independent from unobserved variables $\boldsymbol{x}_U^{(1)}$, $\boldsymbol{x}_V^{(2)}$ given the latent variables $\boldsymbol{z}$. Consequently, the inference about $\boldsymbol{x}_U^{(1)}$, $\boldsymbol{x}_V^{(2)}$ can be reduced to the inference about $\boldsymbol{z}$.

However, the inference under only partially observed data requires the model to share parameters across different sized sets of observed data. For this reason, we use the proposed set function *PNP* [8] to encode observations $\boldsymbol{x}_O^{(1)}$, $\boldsymbol{x}_Q^{(2)}$ of each view. The *PNP* uses an aggregation operation that is invariant to the permutations of elements of $\boldsymbol{x}_O^{(1)}$, $\boldsymbol{x}_Q^{(2)}$, hence allows arbitrary sized observation sets [8]. The obtained encodings can then be used as input for the probabilistic MV-encoder network.

5.2. Generative Process

Similar to [3], we assume that data samples from both data views $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are generated from an unobserved latent variable $\boldsymbol{z}$, which is distributed according to a prior probabilistic distribution, i.e., a Gaussian mixture, and belongs to a cluster c. We aim to maximize the likelihood of data samples from $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ and learn an estimation of the distributions' parameters, i.e., the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$, to group data samples generated from the same distribution to the same cluster c. Analogous to [3], the generative process given two data views can be described as follows:

1. **Cluster selection** In the first step, a cluster c is chosen given its prior probability $\boldsymbol{\pi}$: $c \sim Cat(\boldsymbol{\pi})$, where $Cat(\boldsymbol{\pi})$ indicates its categorical distribution. The Gaussian distribution corresponding to c is described by its mean $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\sigma}_c^2$.

2. **Latent variable sampling** Secondly, latent variables $\boldsymbol{z}$ can be sampled from this distribution: $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \boldsymbol{I})$.

3. **Decoding to observation** The MV-decoder $f(\boldsymbol{z}; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ is used on $\boldsymbol{z}$ and computes the expectation for mean ($\boldsymbol{\mu}_{\boldsymbol{x}^{(1)}}$, $\boldsymbol{\mu}_{\boldsymbol{x}^{(2)}}$) and variances ($\boldsymbol{\sigma}^2_{\boldsymbol{x}^{(1)}}$, $\boldsymbol{\sigma}^2_{\boldsymbol{x}^{(2)}}$) of the data samples from each view $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$.

   Finally, given the Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_{x^{(1)}}, \boldsymbol{\sigma}^2_{x^{(1)}} \boldsymbol{I})$ and $\mathcal{N}(\boldsymbol{\mu}_{x^{(2)}}, \boldsymbol{\sigma}^2_{x^{(2)}} \boldsymbol{I})$ a data sample from $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ can be chosen. In the case of binary-valued data, data points are sampled from a Bernoulli distribution $Ber(\boldsymbol{\mu}_{x^{(1)}})$ and $Ber(\boldsymbol{\mu}_{x^{(2)}})$.

As the goal is to optimize the likelihood of each views' data samples, we build on recent advances in variational inference in order to efficiently train the described latent variable model and introduce similarly to [3] a probabilistic MV-encoder $g(\boldsymbol{x}; \boldsymbol{\phi})$. The MV-encoder aims to approximate the true posterior distribution, hence learns an estimation of the distributions' parameters such that data samples generated from the same distribution can be grouped to the same cluster. As a result, the process of the inference model involves one single step:

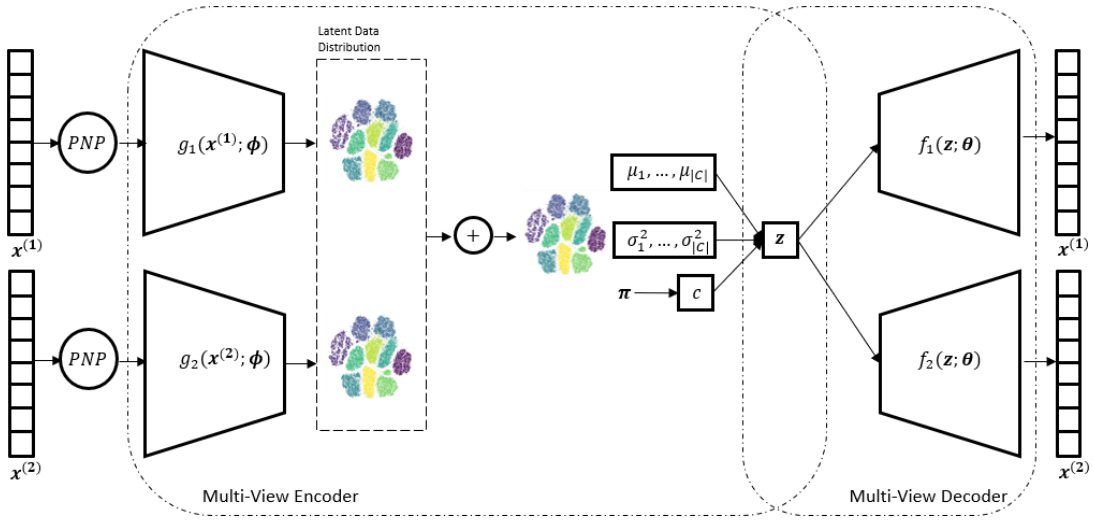


Figure 5.1.: Illustration of the Multi-View VaDE architecture with a PNP setting.

Learning latent embeddings Given the data samples of the first data view $\boldsymbol{x}^{(1)}$ and a second data view $\boldsymbol{x}^{(2)}$, the MV-encoder $g(\boldsymbol{x};\boldsymbol{\phi})$ learns a separate latent embedding for each view. The learned embeddings describe the Gaussian distribution of a cluster c by its mean $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\sigma}_c^2$. As we aim to create a common embedding space shared by both views, we examine an additive and multiplicative technique to fuse the learned embeddings. The process of modality fusion is described in more detail in the following Section 5.4.

Figure 5.1 gives a simplified illustration of the Partial MV-VaDE, where observations of each view are encoded using the *PNP*. The resulting encodings are then used as input for the MV-encoder network, which learns latent embeddings on modality-specific features for each data view. A fusion technique is used to create a latent embedding that is shared by both views. The MV-decoder is used on the latent variables, which are sampled from the clusters, and maximizes the likelihood of generated data samples from both views.

5.3. Loss Function

As mentioned before, Partial MV-VaDE aims at optimizing the likelihood of the data samples from each view $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ and grouping data samples generated from the same distribution to the same cluster c. For this reason, we maximize the ELBO by using the SGVB and the *reparametrization trick* from [7]. Further, to consider both observed $\boldsymbol{x}_O^{(1)}$, $\boldsymbol{x}_Q^{(2)}$ and unobserved $\boldsymbol{x}_U^{(1)}$, $\boldsymbol{x}_V^{(2)}$ data samples, we adapt VaDE's ELBO objective

and define a partial lower bound, which depends exclusively on $\boldsymbol{x}_O^{(1)}$, $\boldsymbol{x}_Q^{(2)}$ as proposed by [8] for single-view data. The final loss function $\mathcal{L}_{partial}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$ can be extended for multi-view and missing data as follows:

$$\begin{aligned}\mathcal{L}_{partial} = \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x}_O^{(1)},\boldsymbol{x}_Q^{(2)})}\big[&\log p(\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}|\boldsymbol{z}) \\ &+ \log p(\boldsymbol{z}|c) + \log p(c) - \log q(\boldsymbol{z}|\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}) - \log q(c|\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)})\big]\end{aligned} \tag{5.1}$$

The objective function can be rewritten as:

$$\begin{aligned}\mathcal{L}_{partial} = \mathbb{E}_{q(\boldsymbol{z},c|\boldsymbol{x}_O^{(1)},\boldsymbol{x}_Q^{(2)})}\big[&\log p(\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}|\boldsymbol{z})\big] \\ &- D_{KL}\big(q(\boldsymbol{z}, c|\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}) \parallel p(\boldsymbol{z}, c)\big)\end{aligned} \tag{5.2}$$

where the first expression is considered the reconstruction term, while the second term defines the KL-divergence between prior and variational posterior. In fact, given our multi-view setting, the reconstruction term comprises a reconstruction loss for data samples from the first view $\boldsymbol{x}^{(1)}$ and data samples from a second view $\boldsymbol{x}^{(2)}$. The notation $\log p(\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}|\boldsymbol{z})$ serves as simplification, which in fact can be composed in:

$$\log p(\boldsymbol{x}_O^{(1)}, \boldsymbol{x}_Q^{(2)}|\boldsymbol{z}) \equiv \log p(\boldsymbol{x}_O^{(1)}|\boldsymbol{z}) + \log p(\boldsymbol{x}_Q^{(2)}|\boldsymbol{z}) \tag{5.3}$$

5.4. Modality Fusion

The MV-encoder learns a latent embedding for each data view $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. The learned embeddings describe the Gaussian distribution of a cluster c by its mean $\boldsymbol{\mu}_{c^{(1)}}$, $\boldsymbol{\mu}_{c^{(2)}}$ and variance $\boldsymbol{\sigma}^2_{c^{(1)}}$, $\boldsymbol{\sigma}^2_{c^{(2)}}$. As we seek to create a shared embedding of both views, we examine two simple fusion functions, i.e., fusion by addition and fusion by multiplication. Both methods are somewhat naive approaches for mapping cluster distributions of two learned embeddings. Surely, more suitable approaches exist to create a meaningful shared embedding space. However, these two methods fit our purpose of determining if the idea of extending VaDE to target multi-view data sets is generally conceivable. Furthermore, the model's architecture facilitates a straightforward integration of both fusion methods. Important to note is that this additive and multiplicative method require the latent dimensions of all considered views to be of the same size.

So, by mapping the mean $\boldsymbol{\mu}_{c^{(1)}}$, $\boldsymbol{\mu}_{c^{(2)}}$ and variance $\boldsymbol{\sigma}^2_{c^{(1)}}$, $\boldsymbol{\sigma}^2_{c^{(2)}}$ of each view's Gaussian

distribution corresponding to a cluster c, we aim to create a distribution of c, that is shared by both data views. The mapping process is accomplished by either

- an additive approach:

$$\boldsymbol{\mu}_c = \boldsymbol{\mu}_{c^{(1)}} + \boldsymbol{\mu}_{c^{(2)}}, \boldsymbol{\sigma}_c^2 = \boldsymbol{\sigma}_{c^{(1)}}^2 + \boldsymbol{\sigma}_{c^{(2)}}^2 \tag{5.4}$$

- or multiplicative approach:

$$\boldsymbol{\mu}_c = \boldsymbol{\mu}_{c^{(1)}} \times \boldsymbol{\mu}_{c^{(2)}}, \boldsymbol{\sigma}_c^2 = \boldsymbol{\sigma}_{c^{(1)}}^2 \times \boldsymbol{\sigma}_{c^{(2)}}^2 \tag{5.5}$$

Both fusion techniques are examined and compared in Section 6.

5.5. Model Implementation

The Partial MV-VaDE was implemented using Python with the deep learning library PyTorch [59]. An implementation of the model and all experiments are available at: `https://github.com/NovakoDo/Deep-Probabilistic-Clustering`. The code used for re-implementing VaDE [1] and integrating the Partial VAE [2] is adapted from their source code, where the deep learning libraries Keras [60] and Tensorflow [61] were used.

We tried to ensure flexibility throughout the implemented network architectures so that single components could be exchanged. By this means, the models can be modified and improved for future work. The files `train.py` and `eval.py` are the main files to train and evaluate the models. All models, including the AE, VAE, VaDE and their variations in architecture (single-view, multi-view, partial-view), are implemented in `models.py`. To support the code flexibility, we use `.yaml`-configuration files, where the model, the data set and corresponding parameters, for example, the learning rate, batch size or model layers, can be defined. We use four different scripts defining training and evaluation functions for each setting: single-view data, multi-view data, single-view partially observed data, and multi-view partially observed data. The files are named according to their setting, e.g., `partial_mv_train_and_eval_functions.py`.

[1] Source code of VaDE [3] available at: `https://github.com/slim1017/VaDE`
[2] Source coder of Partial VAE [8] available at: `https://github.com/microsoft/EDDI`

6. Experiments

We conducted a large number of experiments to assess the performance of our models. In this chapter, we describe the experimental setup, the choice of model parameters, and the obtained results. We will justify our chosen (pre-)training approaches and discuss differences between the models.

The experiments are organized in three parts where, first, we re-assess the performance of VaDE. In the second step, we shift the focus towards multi-view data and extend VaDE's architecture by an additional VAE in order to handle multi-view data, which we refer to as MV-VaDE. We evaluate MV-VaDE on multi-view data sets and compare the performance results to other methods. Finally, we evaluate VaDE with the integrated *PNP* structure, called Partial VaDE, on data sets for which we generate missing data points.

The goals of this chapter are defined as follows:

1. Assessing the performance of VaDE under different experimental conditions:
    - Explaining differences in (pre-)training and reporting procedure compared to [3].
    - Examining multiple pre-training techniques, i.e., the AE, VAE and SAE.
2. Assessing the performance of MV-VaDE:
    - Evaluating if an additional view of data can increase VaDE's clustering performance.
    - Exploring two different fusion techniques, i.e., fusion by addition, and fusion by multiplication.
    - Comparing MV-VaDE to baseline and existing models.
3. Assessing the performance of Partial (MV-)VaDE.
    - Comparing Partial MV-VaDE to existing and similar methods.
    - Showing how effectively Partial VaDE handles higher amounts of missingness.

- Examining the change of cluster probabilities given higher amounts of missingness.

6.1. Evaluation Metrics

The performance of all models is measured using the unsupervised clustering accuracy (ACC) [30], which is defined as follows:

$$\mathrm{ACC} = \max_{m} \frac{\sum_{i=1}^{|\mathrm{K}|} 1\{y_i = m(c_i)\}}{|\mathrm{K}|} \tag{6.1}$$

where $|\mathrm{K}|$ is the total number of samples, y_i is the ground-truth label, c_i is the cluster assignment generated by the model, and m ranges over all possible mappings between clusters and labels. To find the best mapping, we use the Hungarian algorithm [62].

As one evaluation metric might not be enough to provide a full picture of the clustering models' performances, we use the normalized mutual information (NMI) [63] score as an additional performance metric:

$$\mathrm{NMI}(\boldsymbol{y}, \boldsymbol{c}) = \frac{\mathrm{I}(\boldsymbol{y}, \boldsymbol{c})}{0.5(\mathrm{H}(\boldsymbol{y}) + \mathrm{H}(\boldsymbol{c}))} \tag{6.2}$$

where I defines the mutual information between $\boldsymbol{y}$, the ground-truth labels, and $\boldsymbol{c}$, the cluster labels. Mutual information determines the reduction in entropy H of cluster labels and can be expressed as follows:

$$\mathrm{I}(\boldsymbol{y}, \boldsymbol{c}) = \mathrm{H}(\boldsymbol{y}) - \mathrm{H}(\boldsymbol{y}|\boldsymbol{c}) \tag{6.3}$$

Entropy H quantifies uncertainty by:

$$\mathrm{H}(\boldsymbol{y}) = -\sum_{i=1}^{|\mathrm{K}|} p(y_i) \log p(y_i) \tag{6.4}$$

where $p(y_i)$ is the probability of a class label. Both metrics, ACC and NMI, lie in the range between zero and one, where one represents the perfect clustering result.

6.2. Data set Description

For replicating the performance results reported in [3], we evaluate the re-implemented VaDE model on three single-view data sets, which were used in the original paper: MNIST

[64], Reuters-10K [65], and HHAR [66]. The performance of (Partial) MV-VaDE is demonstrated on four multi-view data sets, namely WIKI, HW, BDGP, and MV-MNIST. A comprehensive description of the data sets is given in the following. Also, a brief summary of relevant statistics of the single-view and multi-view data sets can be found in Table 6.1 and 6.3, respectively.

Single-view Data sets

- **MNIST** The MNIST [64] data set consists of 70,000 gray-scaled handwritten digits of size 28 by 28 pixels, showing numbers from zero to nine.
- **Reuters-10K** The Reuters-10k data set forms a subset of 10,000 documents sampled from Reuters [65]. To prevent any possible differences which could impact the model's performance, we use the pre-processed data set [1] generated by [3]. The authors use four categories: (1) corporate/industrial, (2) government/social, (3) markets, and (4) economics, and discard all documents with multiple labels. Based on the remaining documents, tf-id features are computed on the 2,000 most frequent words.
- **HHAR** The Heterogeneity Human Activity Recognition (HHAR) [66] data set comprises 10,299 sensor records generated by smartphones and smartwatches. The data set is classified into six categories of human activities, i.e., biking, sitting, standing, walking, stair up, and stair down, where each sample has 561 describing features.

	MNIST	Reuters10-K	HHAR
# Samples	70,000	10,000	10,299
# Cluster	10	4	6
Input Dim.	784-D	2000-D	561-D

Table 6.1.: Overview of relevant statistics of the used single-view data sets.

Multi-view Data sets

- **WIKI** The Wikipedia (WIKI) data set [67] includes text and image features assembled from the "Wikipedia featured articles". The text features are defined by 10-D vectors, and the images are represented as 128-D vectors containing SIFT descriptors. WIKI comprises 2,866 instances of data, including the ten most frequent categories.

[1] Pre-processed data set is available at: `https://github.com/slim1017/VaDE`

- **HW** The Handwritten numerals (HW) [68, 69] data set consists of 2,000 images of handwritten numerals from zero to nine. In contrast to the MNIST data set, where the model training is performed on the images, the HW data set comprises six different features extracted from the collection of images. We use the 76-D Fourier coefficients of the character shapes and the 216-D profile correlations as two different data views.

- **BDGP** The Berkeley Drosophila Genome Project (BDGP) [70] data set contains 2,500 embryo images of drosophila. The images are classified into five categories, where each corresponds to a stage of gene growth. Each of these images is represented by a 1,750-D visual vector and a 79-D textual feature vector.

- **MV-MNIST** We extend the MNIST data set [64] by a second data view, including categorical features, which describe the handwritten digits. Therefore, we define ten categorical features, which serve as a substitute to the already existing images and share additional information. We combine information on the numeric attributes and the form or shape of the handwritten digits. The data is represented by binary values, where the value 1 is assigned if the feature applies and 0 otherwise. We generate the following features to describe the numeric attributes of the MNIST digits:

  1. Parity: We label even numbers as 1 and odd numbers as 0.
  2. Division by three: Numbers divisible by three without a remainder are encoded as 1, otherwise as 0.
  3. Prime Number: Digits belonging to the group of prime numbers are assigned to 1.
  4. Fibonacci Number: Fibonacci Numbers are encoded as 1.
  5. Sum of two squares: Numbers that can be expressed as a sum of two squares are labeled as 1. This includes the digit zero, which is considered the sum of two squares of zeros.

  Also, we define features describing the shape and form of the handwritten digits using:

  6. Symmetry: Digits that are both horizontally and vertically symmetric are labeled as 1, including the digits zero and eight.
  7. Minimum one curve: Digits, where the shape shows at least one curve, are marked as 1. All the remaining digits, i.e., one, four, and seven, are assigned

to 0.

8. Only curves: If the shape reveals only curves, such as in zero, three, or eight, we assign the data point to 1.
9. Bottom curve: Digits with a bottom curve are marked as 1, i.e., three, five, six, and eight.
10. Horizontal line: Numbers having a horizontal line, including two, five, and seven, are encoded as 1.

Table 6.2 gives an overview of the generated features and their encodings. We use One-Hot Encoding (OHE) as a pre-processing step, which transforms these features into a 20-D vector, where (1,0) is True and (0,1) is False. While using OHE on binary features is somewhat uncommon, it enables to encode missing data as (0,0).

Features/ Digits	0	1	2	3	4	5	6	7	8	9
Parity	1	0	1	0	1	0	1	0	1	0
Division by three	1	0	0	1	0	0	1	0	0	1
Prime number	0	0	1	1	0	1	0	1	0	0
Fibonacci number	1	1	1	1	0	1	0	0	1	0
Sum of two squares	1	0	1	0	1	1	0	0	1	1
Symmetry	1	0	0	0	0	0	0	0	1	0
Minimum one curve	1	0	1	1	0	1	1	0	1	1
Only curves	1	0	0	1	0	0	0	0	1	0
Bottom curve	1	0	0	1	0	1	1	0	1	0
Horizontal line	0	0	1	0	0	1	0	1	0	0

Table 6.2.: Overview of the categorical MNIST features, which serve as a second data view. The features are binary encoded, where 1 represents "True" and 0 is "False".

	MV-MNIST	WIKI	HW	BDGP
# Samples	70,000	2,866	2,000	2,500
# Cluster	10	10	10	5
Input Dim. (View 1)	784-D	10-D	76-D	1,750-D
Input Dim. (View 2)	20-D	128-D	216-D	79-D

Table 6.3.: Overview of relevant statistics of the used multi-view data sets.

As for the data preparation, there exist two relevant scripts, `prepare_datasets.py` to pre-process and split data such that each model can be evaluated on the same sets, and `generate_masks.py` to create masks for each data set indicating missingness. Since we

want to assess the performances of the Partial MV-VaDE in handling missing data points, we need to compare the performance evaluated on the complete view of data. Therefore, we create missingness for commonly used data sets, e.g., MNIST [64] or WIKI [67]. A detailed description of how to run the code can be found in the repository.

6.3. Performance Assessment of VaDE

Since VaDE [3] forms the main building block in this thesis, the first objective is the reconstruction of experiments and the replication of performance results presented in the original paper. Therefore, we re-implemented VaDE using PyTorch [59]. For the sake of comparison, we evaluate the re-implemented VaDE on MNIST, Reuters-10K, and HHAR.

The VaDE model can be organized into three stages:

1. The pre-training procedure
2. The GMM, whose resulting weights are used to initialize VaDE
3. The training procedure of VaDE

The authors distinguish the parameters of the pre-training and training stages. The notations and their corresponding descriptions for the parameter settings used in VaDE are summarized in Table 6.4. These notations are consistent across all experiments. To

Notation	Description
D	Input data dimension
L	Latent dimension of set function encoding
$\mathcal{L}_r$	Loss function for the reconstruction term
T_p	Number of pre-training epochs
α_p	Learning rate during pre-training
T_l	Number of training epochs
α_l	Learning rate during training
Decay	Decay for the learning rate scheduler
Step size	Step size for the learning rate scheduler
λ_l	Regularization parameter for the cluster loss during training

Table 6.4.: List of notations and their corresponding descriptions used in VaDE.

replicate the performance of VaDE, we adopt the parameter settings from [3].

Pre-training Procedure [3] explain that VaDE is pre-trained using an SAE because the model suffers from a weak reconstruction. While the authors claim that a few epochs of SAE training provide a good initialization of VaDE, we experienced several difficulties

because of the high volatility in performance given multiple runs of pre-training. For this reason, we think that the pre-training procedure is crucial for the resulting performance of VaDE.

Therefore, we examine different pre-training models, including the SAE, VAE, and AE. For each model and data set, we perform 50 runs, where we set the number of epochs for the AE and VAE to 50. As for the SAE model, each layer is pre-trained for 25,000 iterations and fine-tuned for another 100,000 iterations. The learning rate is set to 0.001, and Adam [71] is used to optimize the loss function. These settings are constant across all three data sets.

GMM After the pre-training procedure, a GMM is applied to the encoded input data. The resulting weights, means, and variances are used to initialize VaDE's parameters, i.e., the cluster probability $\boldsymbol{\pi}$, the cluster mean $\boldsymbol{\mu}_c$, and the cluster variance $\boldsymbol{\sigma}_c^2$. The Reuters-10K data set is considered an exception because in the original implementation, the k-means algorithm is used to initialize $\boldsymbol{\pi}$ for this data set. Although this choice is not reasoned in [3], we use this same approach for our experiment.

Also, opposed to the original implementation where the number of restarts for the GMM and k-means is set to one, we use 100 initializations for each applied GMM, where the best output in terms of inertia is taken [72]. With this setting we aim to minimize the variations in GMM performance.

So, we apply a GMM using 100 restarts on top of each of the 50 pre-trained models for each data set and then pick the ten best performing pre-trainings according to the obtained ACC score for further training with VaDE. With this setting, we aim to decrease the high volatility between VaDE runs and leave this issue for future work.

Training procedure of VaDE Finally, VaDE's network and parameters are initialized using the pre-trained weights and the GMM results, and the actual training stage starts. Following [3], the decoder network architecture is defined as 10-2000-500-500-D, where D specifies the input dimension. The architecture of the encoder network has a mirrored structure. All layers are fully connected and activated with the ReLU [73] activation function. The loss function is optimized using Adam [71]. The number of classes in each data set represents the number of clusters used for the model training. As the authors distinguish the training parameters of the three data sets, Table 6.5 serves as an overview of the chosen settings.

Evaluation We compare the reported performance results of the original VaDE with our re-implemented version. In particular, we examine five different models, which differ mainly in their pre-training procedure. For simplicity and easier reading, we name the

Parameters	MNIST	Reuters-10K	HHAR
Network	D - 500 - 500 - 2000 - 10		
$\mathcal{L}_r$	BCE	MSE	MSE
T_p	50	50	50
α_p	0.001	0.001	0.001
T_l	300	15	120
α_l	0.002	0.002	0.002
Decay	0.9	0.5	0.9
Step size	10	5	10
Batch size	100	100	100

Table 6.5.: Parameter settings of VaDE for MNIST, Reuters-10K and HHAR adopted from [3].

different VaDE models with the chosen pre-training model as a prefix: *pre-trained model* + VaDE. We use a basic Autoencoder as pre-training step for the first model, (1) AE + VaDE. The second model, (2) VAE + VaDE, is pre-trained with a Variational Autoencoder. Because VaDE is based on a VAE and both models optimize the KL-divergence loss, we want to determine if using a VAE leads to a more suitable initialization for VaDE. To efficiently replicate results, we also explore the SAE as a pre-training method in (3) SAE + VaDE.

Contrary to [3], where the maximum ACC score is reported, we average the performance of ten VaDE runs. With this setting, we show that VaDE is not sufficiently robust when comparing several runs.

The experimental setup of these three methods can be summarized as follows:

1. We perform 50 runs of pre-training for each data set using three different models, i.e., the SAE, VAE and AE.

2. For each pre-trained model, a GMM using 100 restarts is applied on the encoded input data.

3. Next, we pick the ten best out of 50 pre-trainings with the highest ACC score for each model (SAE, VAE and AE).

4. These pre-trainings and their corresponding GMM results are used to initialize VaDE's network and parameters, respectively, for further training.

As for the last model, (4) SAE (pretr.) + VaDE, we use the pre-trained weights [2] from the original implementation. Since only the weights of one single run are published, we

[2]Pre-trained weights available at: `https://github.com/slim1017/VaDE`

use the same pre-trained weights for ten runs of VaDE. By this means, we can determine if the volatility of VaDE's clustering scores is high given the same weights initializations. Last but not least, (5) Original VaDE reports the published scores from [3], where the authors state the maximum ACC score.

Results Table 6.6 compares the performance results of the five methods and reports the average ACC and NMI scores across ten runs. We see that VaDE's performance varies depending on the chosen pre-training model. However, none of the pre-training models seems to be significantly better than the other. Furthermore, we observe substantial variances in the resulting ACC and NMI scores. The standard deviations of each method AE + VaDE, VAE + VaDE and SAE + VaDE are high for all three data sets. As for (4) SAE (pretr.) + VaDE, we have lower variations in results because each VaDE run uses the same pre-trained weights. Because of these observations, we found that VaDE's performance depends on the pre-training step and the initialization of parameters.

	MNIST		REUTERS-10K		HHAR	
Methods	ACC	NMI	ACC	NMI	ACC	NMI
AE + VaDE	77.36(±2.84)	78.30(±1.54)	64.15(±3.19)	28.87(±3.79)	**69.23**(±7.27)	**68.05**(±5.10)
VAE + VaDE	**81.27**(±3.40)	**79.33**(±2.27)	56.55(±8.70)	27.39(±6.75)	56.01(±4.54)	64.86(±2.82)
SAE + VaDE	80.44(±4.05)	77.30(±2.32)	65.56(±5.08)	**39.19**(±4.91)	53.49(±2.97)	45.55(±2.50)
SAE (pretr.) + VaDE	79.04(±0.13)	78.69(±0.19)	**69.52**(±1.04)	38.67(±1.23)	61.31(±2.08)	65.24(±0.96)
Original VaDE[1]	**94.46**		**79.83**		**84.46**	

[1] Maximum performance

Table 6.6.: Evaluation of VaDE on MNIST, Reuters-10K and HHAR. Comparison of average ACC (%) and NMI (%) scores, where we performed 50 runs of pre-training and chose the best ten for further training with VaDE.

Since the difference in reporting style makes a fair comparison more difficult, Table 6.7, 6.8, and 6.9 serve as a more detailed summary of the performance statistics. We include the minimum, maximum, and mean ACC scores and the obtained standard deviation across ten runs for each data set. The performance statistics show that the range between the minimum and the maximum scores across ten runs is extensive. This inconsistency could be challenging when dealing with an unsupervised learning task. We can replicate similar results for the MNIST and HHAR data set. However, the maximum ACC scores for Reuters-10K are relatively poor compared to [3].

Methods	Min	Max	Mean	Std
AE + VaDE	72.81	82.77	77.36	2.84
VAE + VaDE	77.24	89.34	**81.27**	3.40
SAE + VaDE	74.97	**91.11**	80.44	4.05
SAE (pretr.) + VaDE	**78.87**	79.25	79.04	**0.13**
Original Vade	-	**94.46**	-	-

Table 6.7.: Evaluation of VaDE on MNIST. Overview of ACC (%) performance statistics, including max, min, mean and std across ten runs of VaDE.

Methods	Min	Max	Mean	Std
AE + VaDE	59.00	68.20	64.15	3.19
VAE + VaDE	39.40	69.30	56.55	8.70
SAE + VaDE	50.70	69.20	65.56	5.08
SAE (pretr.) + VaDE	**67.60**	**70.90**	**69.52**	**1.04**
Original Vade	-	**79.83**	-	-

Table 6.8.: Evaluation of VaDE on Reuters-10K. Overview of ACC (%) performance statistics, including max, min, mean and std across ten runs of VaDE.

Methods	Min	Max	Mean	Std
AE + VaDE	**60.97**	**83.40**	**69.23**	7.27
VAE + VaDE	49.93	65.05	56.01	4.54
SAE + VaDE	47.36	58.54	53.49	2.97
SAE (pretr.) + VaDE	58.16	65.44	61.31	**2.08**
Original Vade	-	**84.46**	-	-

Table 6.9.: Evaluation of VaDE on HHAR. Overview of ACC (%) performance statistics, including max, min, mean and std across ten runs of VaDE.

6.4. Evaluation of MV-VaDE

We want to take advantage of additional information and extend VaDE's architecture by a second network to process multi-view data. We aim to investigate how MV-VaDE handles an additional data view and determine if multi-view data can improve the generative model and clustering result. Further, we examine two basic fusion techniques to create a common embedded space of both data views, i.e., fusion by addition and fusion by multiplication. A more detailed explanation on the MV-VaDE architecture and the fusion techniques was given in Section 5. We evaluate MV-VaDE on four multi-view data sets, including MV-MNIST, WIKI, HW, and BDGP and compare it to two baseline models, AE + GMM and VAE + GMM, as well as four multi-view clustering models, including MvSN [4], DCCA [39], DMSC [40], and DAMC [24].

6.4.1. Fusion Technique Examination

In the first part of this section, we want to examine the two discussed fusion techniques, i.e., fusion by addition and fusion by multiplication. As mentioned in Section 5.2, these two methods are somewhat naive approaches for dealing with two data views in probabilistic clustering. While surely more suitable methods exist for creating a unified distribution, we still aim to determine if one of these two basic approaches fits the purpose of MV-VaDE. For this reason, we compare two MV-VaDE models, MV-VaDE (Add), where we create a shared space for two data views using the additive approach and MV-VaDE (Mul) integrating the multiplicative technique. We evaluate both models on four multi-view data sets, namely MV-MNIST, WIKI, HW, and BDGP.

Training Procedure Since we could not determine a clear preference in pre-training models for VaDE in Section 6.3, we performed a few runs to decide on the most suitable option for each data set. We use a simple AE in the pre-training stage for MV-MNIST, HW, and BDGP. Pre-training a VAE combined with the MSE Loss for the reconstruction term proved to be the better alternative for the WIKI data set.

The fusion techniques require a unified dimension for both input modalities. Hence, both views have the same embedding size, which we chose according to the number of classes. Table 6.10 gives a more detailed summary of the parameters used for each data set.

	MV-MNIST	WIKI	HW	BDGP
Network (1)	D-500-500-2000-10	D-500-500-200-10	D-500-500-2000-10	D-500-500-2000-5
Network (2)	D-15-10	D-500-500-200-10	D-500-500-2000-10	D-500-500-2000-5
$\mathcal{L}_r$	BCE	MSE	BCE	BCE
T_p	50	50	50	50
α_p	0.001	0.001	0.001	0.001
λ_p	-	0.0001	-	-
T_l	300	100	100	100
α_l	0.001	0.0001	0.0001	0.0001
λ_l	-	0.0001	0.001	0.001[1]
Batch size	256	128	128	128

[1] Used only for Partial MV-VaDE.

Table 6.10.: Parameter settings of MV-VaDE for MV-MNIST, WIKI, HW, and BDGP.

As it was done in Section 6.3, we perform 50 runs of pre-training for each model, MV-VaDE (Add) and MV-VaDE (Mul), on each data set. Subsequently, we apply a

GMM and pick the ten pre-trained models with the highest ACC score for further training with MV-VaDE. Again, with this setting, we aim to decrease the high volatility between MV-VaDE runs.

		MV-VaDE (Add)	MV-VaDE (Mul)
MV-MNIST	ACC	**99.11(**±0.10**)**	98.81(±3.04)
	NMI	97.62(±0.27)	**99.11(**±0.99**)**
WIKI	ACC	**58.27(**±0.62**)**	56.58(±1.58)
	NMI	**52.02(**±0.77**)**	51.89(±0.78)
HW	ACC	**85.25(**±3.40**)**	73.85(±4.82)
	NMI	**80.50(**±3.04**)**	71.53(±5.51)
BDGP	ACC	**81.30(**±3.29**)**	76.96(±9.06)
	NMI	**71.02(**±5.79**)**	66.64(±8.51)

Table 6.11.: Comparison of two fusion techniques for MV-VaDE, evaluated on MV-MNIST, WIKI, UCI-mf, and BDGP. We report the average ACC (%) and NMI (%) scores, where we performed 50 runs of pre-training, and chose the ten best for further training with MV-VaDE.

Evaluation and Results We evaluate the ten runs of MV-VaDE (Add) and MV-VaDE (Mul) on the MV-MNIST, WIKI, HW, and BDGP and report the average ACC and NMI scores in Table 6.11.

Considering the results for MV-MNIST, we can see an increase in performance compared to VaDE trained on the single-view MNIST images. By combining the images with the categorical features, we can increase the average ACC score from 77.36% to 99.11% when comparing the results of AE+VaDE, which uses the same pre-training model as MV-VaDE.

Also, the performance scores show that MV-VaDE (Add) outperforms MV-VaDE (Mul). Regarding the standard deviation across the performed runs, creating a shared distribution using the additive approach seems the more robust method, while fusion by multiplication reveals more inconsistencies in the results. For this reason, we decide to use the additive approach for all following experiments with multiple modalities. From now on, we refer to this method simply as MV-VaDE.

6.4.2. Comparison to Baseline Methods

In this second part, we compare the clustering performance of MV-VaDE to two baseline methods, AE+GMM and VAE+GMM, and four state-of-the-art deep clustering algorithms for multi-view data. The considered comparison methods are MvSN [4], DCCA [39], DMSC [40], and DAMC [24]. A more detailed description of these methods can be found in Section 4.3. Also, we want to investigate the Partial MV-VaDE, here using a complete view of data.

Training Procedure For reporting the MV-VaDE results, we re-use the already trained models from MV-VaDE (Add). The Partial MV-VaDE is trained according to the same procedure as MV-VaDE. The only difference is that we increase the weight of the reconstruction term of the first HW data view (76-D Fourier coefficients) by ten. Also, we use a parameter to regularize the KL-divergence for this setting. These two measures are taken to improve the performance results of Partial MV-VaDE given the more complex architecture. However, we disregard these parameters for MV-VaDE, because we observed a decrease in performance.

As for the comparison methods, we do not perform an independent training, but report the published scores. More specifically, we take the ACC and NMI scores of MvSN from [4], where the mean results of 30 conducted runs are reported. The authors evaluate MvSN on the WIKI, and HW data set. For the remaining methods, DCCA, DMSC, and DAMC, we use the performance results from [24]. All three algorithms were evaluated on HW and BDGP. Since DMSC was initially intended for clustering image data, [24] implements zero-padding to use it on HW and BDGP. Also, the authors explain that this approach lowers the algorithm's performance. As we did not find nearer information on the experimental setting, we assume the authors reported the scores of their best runs.

While we re-use the state-of-the-art algorithms' published results, the baseline models are implemented with the same experimental setting as MV-VaDE and Partial MV-VaDE. More specifically, we pre-train 50 AEs and VAEs for each data set and use them for all following methods. For the AE+GMM method, we apply a GMM on each pre-trained AE. Then, we pick the best ten models with the highest ACC score and fine-tune the AE for another 100 epochs. At the end of the training procedure, a second GMM is applied, which returns the final clustering result. The same approach is employed for VAE+GMM using a VAE model. The methods AE+GMM* and VAE+GMM* show the GMM clustering results, which are obtained after the pre-training stage. Also, we use an additional GMM after the MV-VaDE training and refer to this method as MV-VaDE+GMM.

Evaluation and Results We evaluate each method on the same test sets and report

the ACC and NMI scores of the best run in Table 6.12. The results show that the fine-tuning for AE+GMM and VAE+GMM did not necessarily increase the performance compared to AE+GMM* and VAE+GMM*, respectively. Particularly for BDGP, the model's performance profoundly decreases. A possible reason for this adverse effect could be that the model gets stuck in a local minimum. The scores for MV-VaDE lie in a similar range as the baseline models. The GMM on top of MV-VaDE slightly improves the clustering result for the HW data set. While the clustering with Partial MV-VaDE on the HW data set is suboptimal, the obtained performance on WIKI is at least comparable to MV-VaDE and MV-VaDE+GMM. The regularization term λ_l in Partial MV-VaDE shows a positive effect on the model's performance on the BDGP data set.

	WIKI		HW		BDGP	
	ACC	NMI	ACC	NMI	ACC	NMI
MvSN	**92.21**	**87.32**	95.44	**96.65**	-	-
DCCA	-	-	81.40	78.10	57.80	40.90
DMSC	-	-	91.60	85.50	68.10	50.60
DAMC	-	-	**96.50**	93.20	**98.20**	**94.60**
VAE + GMM*	59.02	53.01	92.50	89.03	78.80	65.85
VAE + GMM	59.60	53.31	89.50	84.41	54.40	35.35
AE + GMM*	59.02	**54.40**	**93.00**	**89.47**	79.20	68.14
AE + GMM	**59.74**	54.06	90.00	84.39	72.80	62.86
MV-VaDE	59.45	53.26	90.50	85.60	87.60	81.84
MV-VaDE + GMM	59.31	53.06	92.50	88.68	86.00	78.23
Partial MV-VADE	57.86	50.41	71.00	69.14	**94.80**	**88.23**

Table 6.12.: Comparison of MV-VaDE with several (baseline) models on WIKI, HW and BDGP. We report the best run's ACC (%) and NMI (%) score for MV-VaDE and the baseline models. The results of the state-of-the-art models are taken from [4] and [5].

6.5. Evaluation of Partial (MV-)VaDE

Finally, in this section, we assess the Partial VAE component of the Partial MV-VaDE and aim to learn how well missing data can be handled. First, we evaluate the Partial MV-VaDE on three multi-view data sets for which we generate missing data samples. We compare the Partial MV-VaDE to one baseline model as well as DCCA and DAMC. It is essential to note that both models were not initially designed to handle missingness in

data. However, we want to determine if Partial VaDE can benefit from the Partial VAE + *PNP* architecture and effectively deal with missing data samples. Next, we disregard the multi-view setting and focus solely on the ability to handle missingness. We evaluate Partial VaDE using various stages of missingness and observe the model's changes in cluster probabilities with an increasing number of observed data samples.

6.5.1. Comparison to Baseline Methods

This first experiment evaluates Partial MV-VaDE on WIKI, HW, and BDGP, for which we forge missingness. We compare Partial MV-VaDE to a baseline model, DCCA and DAMC.

Training Procedure Therefore, we integrate a dropout rate ranging from 0% to 50% during the model (pre-)training. This dropout rate defines the number of missing values in data patches. We create so-called missingness masks for each epoch and data batch, which indicate missing data samples with value 0 and observed data samples with value 1. Hence, multiplying the generated mask with the input data results into new data samples implicating missingness. For example, given a normalized MNIST image, where white pixels describe the digit and black pixels the background with values 1 and 0, we indicate missing values by masking white pixels with 0. A more detailed illustration of the generated missingness masks is given in Section 6.5.3.

For each data set, we perform 50 runs of pre-training and apply a GMM subsequently. We then pick ten pre-trained models with the highest ACC score for further training with Partial MV-VaDE. As done for MV-VaDE, we pre-train the BDGP and HW data set using an AE and WIKI on a VAE. Also, we use the same parameters as defined for MV-VaDE, which were described in Table 6.10. Additionally, to these chosen parameters, the *PNP* [8] in Partial MV-VaDE requires to define the latent dimension L of the set function encoding (see Section 3.3.1), which we set to 20 for all data sets.

For the training procedure of DCCA [3] and DAMC [4], we use existing implementations and perform ten runs of each model on each data set. We set the same parameters used for Partial MV-VaDE, including the network architecture, learning rate, batch size and epochs, except for DAMC, where we adopt the number of epochs from the original paper and use 30 epochs for each training step. As DCCA requires further clustering, we use k-means on top of the trained model. To generate missingness, we use the same approach as for Partial MV-VaDE. The difference here is that the architecture of DCCA and

[3]Implementation used from: `https://github.com/Michaelvll/DeepCCA`
[4]Source code available at: `https://github.com/IMKBLE/DAMC`

DAMC was not initially designed for dealing with missingness, so it does not integrate a set function encoding [8] as in Partial VaDE. For this reason, we use a zero imputation method, where input samples and corresponding masks are multiplicated and fed to the network as one input.

Evaluation To assess the performance of Partial MV-VaDE and determine the volatility of results when dealing with randomly missing data samples, we pick the best model according to the ACC score and evaluate its performance on ten sets of missingness masks for the test set. More specifically, we generate ten random masks with a dropout rate of precisely 50%. The choice of which values are missing or observed is made randomly. By generating ten masks, we intend to ensure enough randomness among missing data values. So, we evaluate the best run of Partial MV-VaDE on these ten test set + missingness masks combinations generated for each data set. Each comparison method's best run is evaluated on the same sets.

Besides DCCA and DAMC, we compare the results of Partial MV-VaDE to Partial (V)AE + GMM, which can be considered the pre-training procedure of the Partial MV-VaDE model and Partial MV-VaDE + GMM, where we apply an additional GMM on top of the trained model. We refer to this comparison model as Partial (V)AE + GMM, because for pre-training on the WIKI data set, we used the VAE, which slightly improved the clustering results. However, the pre-training on the HW and BDGP data set is yield by using an AE. By this means, we intend to learn if the model training stopped at an inconvenient point where the cluster representation of Partial MV-VaDE is weak.

Results Table 6.13 summarizes the average ACC and NMI scores for each data set, where each model's best run was evaluated on ten test sets. We see that the methods with the Partial VAE + *PNP* architecture handle missing data samples more effectively than deep clustering algorithms with a more traditional architecture that was not designed for missing data. The results achieved for the WIKI data results are generally poor, including the Partial MV-VaDE which shows somewhat difficulties with this data set. Noticeable is that DAMC achieves a better clustering performance on WIKI data. Still, we see that in comparison the Partial MV-VaDE model achieves higher results for HW. The Partial MV-VaDE + GMM can slightly improve the ACC score of Partial MV-VaDE. However, an additional GMM on top of Partial MV-VaDE does not significantly improve the clustering results. Interestingly enough, the NMI score obtained on BDGP with AE + GMM is considerably higher than after the Partial MV-VaDE training.

	WIKI		HW		BDGP	
	ACC	NMI	ACC	NMI	ACC	NMI
DCCA	23.25(±1.31)	10.85(±1.68)	28.25(±3.04)	23.91(±3.52)	49.96(±8.55)	34.98(±7.42)
DAMC	**33.12**(±2.11)	**26.97**(±2.13)	23.35(±0.90)	23.90(±1.92)	33.20(±1.74)	9.49(±2.66)
Partial (V)AE + GMM[1]	25.56(±1.11)	15.33(±1.40)	42.80(±2.24)	44.07(±2.37)	53.04(±2.39)	**47.33**(±2.98)
Partial MV-VADE	25.95(±1.64)	15.09(±0.80)	**53.33**(±2.68)	**53.13**(±2.52)	56.36(±1.85)	38.39(±2.33)
Partial MV-VaDE+GMM	25.87(±1.28)	15.89(±0.68)	50.31(±1.45)	49.25(±1.62)	**57.24**(±1.90)	38.22(±3.30)

[1] VAE is solely used for WIKI.

Table 6.13.: Comparison of Partial MV-VaDE with DCCA, DAMC and one baseline model on WIKI, HW and BDGP. We report the average ACC (%) and NMI (%) scores for each method, where we picked the best model out of ten and evaluated this run using the same ten test sets.

6.5.2. Evaluation of Partial VaDE on Higher Missingness

We examine Partial VaDE's ability in handling missing data points in more detail. In particular, we are interested in the model's clustering performance given different amounts of missingness.

Training Procedure We integrate a random dropout rate ranging from 0% to 100% for the (pre-)training process. More specifically, we create a mask for each epoch and data batch, which indicates missing data points with the value 0 and observed data points with the value 1. Each mask denotes a randomly chosen number of random data samples as missing (or unobserved), where the number can range between 0%, which implies a fully observed set of samples, and 100% indicating an entirely unobserved set of data samples.

We train Partial VaDE using this setting on the MNIST images and the MNIST features, considered as two separate data sets for this experiment. Since here, we want to focus on assessing the effectiveness of the Partial VAE [8] component of Partial MV-VaDE, we disregard the multi-view setting and train two single-view Partial VaDE models independently on the MNIST images and features. We refer to these models as Partial VaDE (M), where (M) indicates the (pre-)training with missing data samples.

We compare Partial VaDE (M) with another Partial VaDE model, only that in this setting, the model sees a complete view of data during the (pre-)training process. Hence, to this model, we refer to as Partial VaDE (F), where (F) indicates the (pre-)training on fully observed data samples. Analogous to Partial VaDE (M), Partial VaDE (F) (pre-)training is performed separately on the MNIST images and features.

The parameter settings for both data sets are summarized in Table 6.14 and are the same for both models Partial VaDE (M) and Partial VaDE (F). Equivalent to the previous

	MNIST Images	MNIST Features
Batch size	256	256
L	20	20
Network	L-500-500-2000-10	L-500-500-200-10
$\mathcal{L}_r$	BCE	BCE
T_p	50	10
α_p	0.001	0.001
λ_p	-	-
T_l	300	50
α_l	0.0001	0.0001
λ_l	-	-

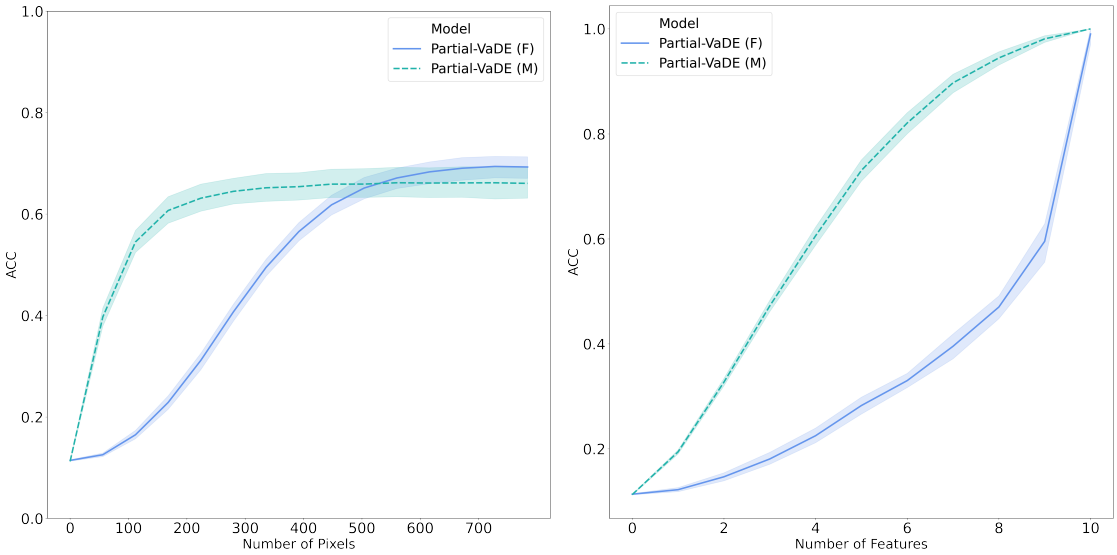
Table 6.14.: Parameter settings of Partial VaDE (M) and Partial VaDE (F) for MNIST images and features.

experiments, we performed 50 pre-trainings for each model and on each data set. For each pre-training, we subsequently applied a GMM and picked ten pre-trained models with the highest ACC score for further training with Partial VaDE. So, we have ten models of each method for both data sets. Since we encountered some difficulties in pre-training MNIST features on the AE, we chose the VAE for this scenario. However, the pre-training of the MNIST images is based on the AE as it was done before.

Evaluation As we are interested in the models clustering performance based on the amount of missingness in the data sets, we evaluate both models on several missingness-masks masking the same test set, where for each mask, we define a different degree of missing data samples. As for the MNIST images, we generate 14 masks, where for each mask, we increase the number of observed data points (pixels) in a step-wise manner. The choice of which data points are observed or unobserved is made entirely random. So, the first mask indicates an entirely unobserved test set, meaning zero pixels are observed. We increase the number of observations (pixels) by a step-size of 56 for each other generated mask so that the final mask shows a fully observed test set.

To ensure enough randomness in missing data samples, we create ten different masks for each step size, e.g., ten masks, where 280 randomly chosen pixels are denoted as missing. By this setting, we want to determine the volatility in results obtained by Partial VaDE given different missing data samples.

The same mask generation process is used for the MNIST features, only that here, given ten dimensions, we use a step size of 1. This results in ten differently masked test sets for ten missingness stages, wherein one more randomly chosen feature is observed in each stage.



(a) MNIST images

(b) Categorical MNIST features

Figure 6.1.: Visualization of the ACC score of the Partial VaDE models given an increasing number of observed values, demonstrated on the MNIST image data set and the categorical MNIST features data set. Partial VaDE (F) was trained with the complete data view, while Partial VaDE (M) was trained on the same data set but using a dropout range between 0-100%.

Results We evaluate the ten trained models of each method on the same generated test + masks sets for each missingness stage and visualize the obtained ACC scores in Figure 6.1. While Figure 6.1a shows both models' results for the MNIST images, Figure 6.1b reveals both models' scores for MNIST features. The curves indicate the averaged ACC score and volatility across ten runs, each evaluated on ten masks of the test set for each missingness stage (number of pixels/features).

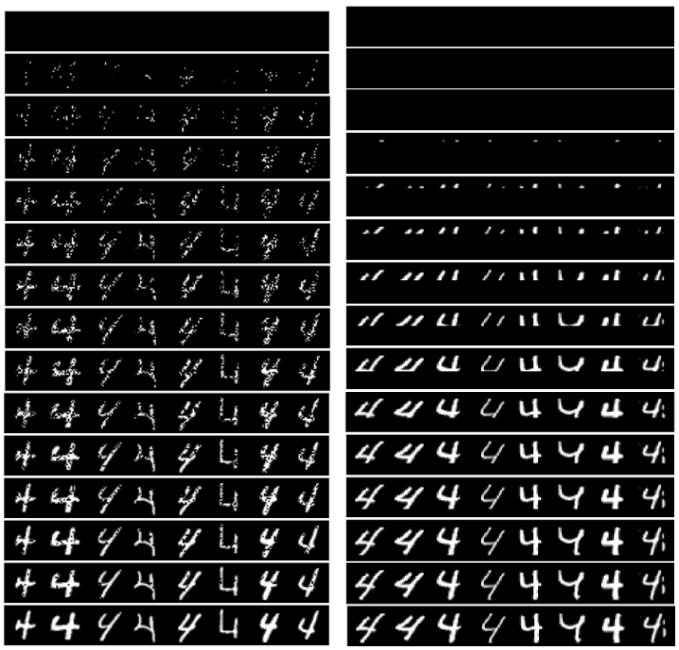
We see that Partial VaDE (M) can handle missing data samples more effectively while Partial VaDE (F) encounters difficulties given only a few observed pixels. Especially regarding the performance on MNIST images, we notice that Partial VaDE (M) achieves a similar ACC score as Partial VaDE (F) with less than half of the observed pixels. Also, the evaluation of the MNIST features shows that Partial VaDE (F) requires more observations to obtain a higher clustering ACC.

6.5.3. Changes in Cluster Probability Given Missing Data

The previous experiment showed that Partial VaDE (M) trained on randomly missing data samples performs similarly well to Partial VaDE (F) given fewer observations, thus

can handle missingness more effectively. However, we are not only interested in the clustering scores of the models but want to investigate their changes of probability for a cluster, i.e., describing the digit 4, in more detail using a similar setting. The cluster probability defines the probability of a data sample belonging to the same distribution. In the following experiment, we examine how the cluster probability of Partial VaDE (M) changes given the same missingness stages discussed in the previous section. By this means, we compare Partial VaDE (M) to Partial VaDE (F) using the same models trained on MNIST images and features from earlier (see previous Section 6.5.2).

Evaluation In this experiment, we differentiate between random and systematic missingness. We aim to learn differences in the models' changes of cluster probabilities when data are missing at random or in the form of systematic patches. Therefore, we evaluate the models in two settings: (1) In the first setting, we use the earlier generated sets of masks containing randomly missing data samples. (2) Additionally, we create masks for the same missingness stages but data patches are missing in a sequential order. Figure 6.2 illustrates the two types of missingness for each stage on the example of MNIST images.



(a) Random missingness (b) Sequential missingness

Figure 6.2.: Illustration of the two missingness settings.

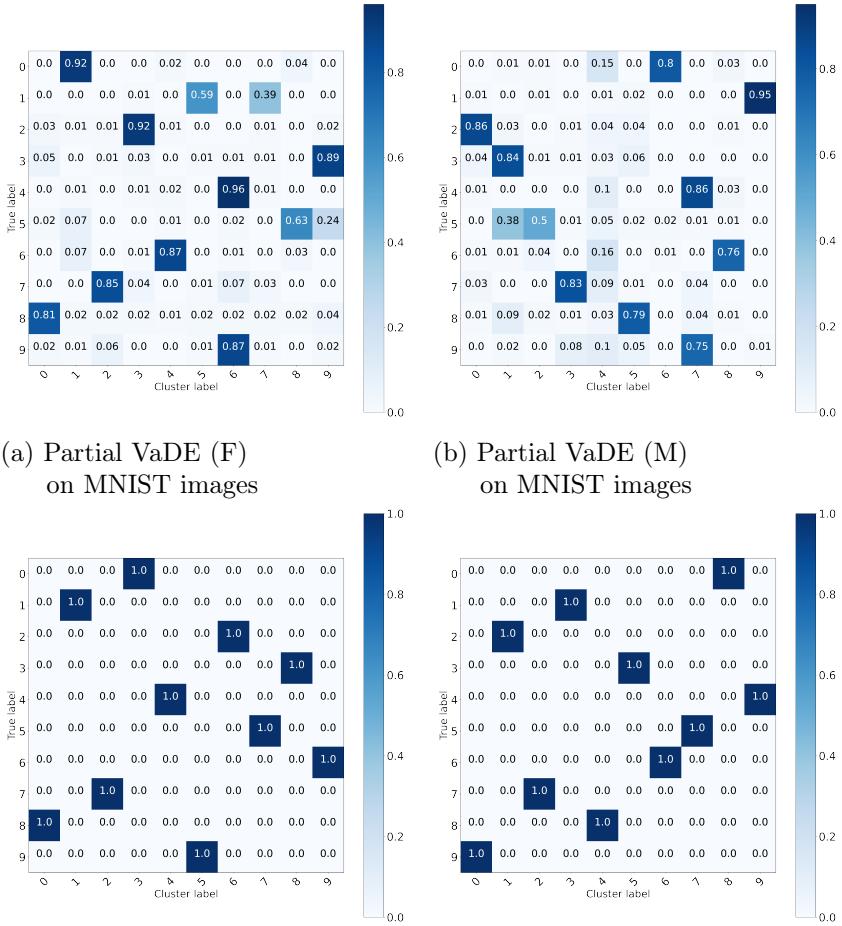
Since we are dealing with an unsupervised model, the resulting cluster labels can differ

from the ground-truth labeling. For this reason, we use a confusion matrix and match the true labels to our obtained cluster labels. Thus, we pick the best performing run of Partial VaDE (M) and Partial VaDE (F) measured by the ACC score and test it on the fully observed data set of MNIST images and features such that we can create a confusion matrix matching the obtained cluster labels with the MNIST ground-truth labels. Figure 6.3 visualizes the resulting mapping for each model and data set. Regarding the MNIST features, we obtain a distinct mapping between cluster and true labels by both models. However, the labels obtained by the models trained on the MNIST images are not distinctly matched. Thus, we pick the top two represented ground-truth labels found in the same cluster. We use these mappings later to visualize each model's changes in cluster probabilities.

We observe the change in the models' cluster probabilities given only images or features as input with the ground-truth label 4. Therefore, we randomly sampled 500 data points with the ground-truth label 4 from the MNIST image and feature test set. We test each model, the Partial VaDE (M) and the Partial VaDE (F), on each data set, MNIST images and features along with the ten generated masks per missingness stage and compute the models' cluster probabilities for each step. As previously explained, we use two settings, where first we test the models using random missingness and secondly sequential missingness. As for the second setting, we do not expect as much variations since the order of sequential missingness always stays the same. Still, also here, we perform ten runs with the sequential missingness mask.

Results Figure 6.4 and 6.5 visualize the cluster probability, according to the increasing number of observed pixels/features for each model Partial VaDE (F) and Partial VaDE (M), respectively. The cluster probability is computed based on the input data samples with the ground-truth label 4. For both figures, the two left plots show the probabilities obtained by the first setting using random missingness, whereas the plots on the right correspond to the second setting of sequential missingness.

In Figure 6.4c, we can see that the cluster label 6 obtained by Partial VaDE (F) is primarily represented by the ground-truth labels 4 and 9. More precisely, 96% of data points belonging to the ground-truth label 4 find themselves in cluster label 6. As expected, the probability of this cluster increases with the number of observed pixels. Comparing this result to Figure 6.4b, we see that more pixel observations are needed for the sequential missingness setting to obtain a higher probability certainty for cluster label 6. However, this could be explained by Figure 6.2b, which has shown that adding pixels sequentially to an entirely unobserved MNIST image leads to no information gain



(a) Partial VaDE (F) on MNIST images

(b) Partial VaDE (M) on MNIST images

(c) Partial VaDE (F) on MNIST features

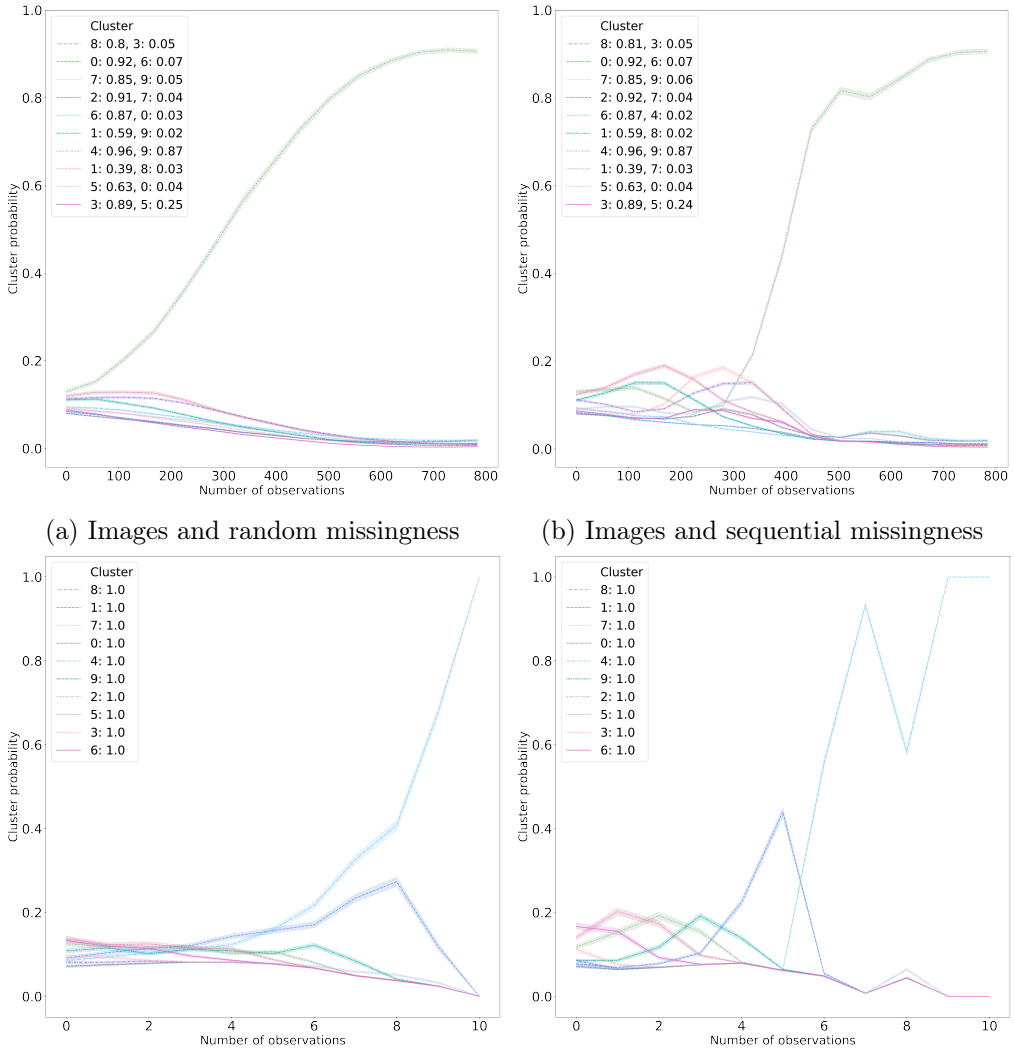
(d) Partial VaDE (M) on MNIST features

Figure 6.3.: Mapping of true labels and cluster labels for Partial VaDE (F) and Partial VaDE (M) demonstrated on the MNIST images data set and the MNIST features data set.

for the second and third missingness stage because the first 112 pixels denote the black background only.

On the other hand, Figures 6.4c and 6.4d summarize the cluster probabilities obtained on the MNIST features. We can see a distinct mapping between cluster label 4 and ground-truth label 4, and that the probability for this label increases with the number of observed features. However, a comparison of the random versus sequential setting is complex for these features because of their ordering, which was shown in Table 6.2. Here, the feature *parity* is the first to be observed. Yet, several other digits share the same

characteristic. In contrast, the *min one curve* feature is not true for digit 4 and only two others. Seeing this information first, the model may have a lower probability certainty for all other ground-truth labels but 1, 4, and 7 from the beginning. While these are only assumptions, cluster probabilities in Figure 6.4d may still be affected by the ordering of features. Furthermore, it is essential to note that each MNIST image of digit 4 can look



(a) Images and random missingness

(b) Images and sequential missingness

(c) Features and random missingness

(d) Features and sequential missingness

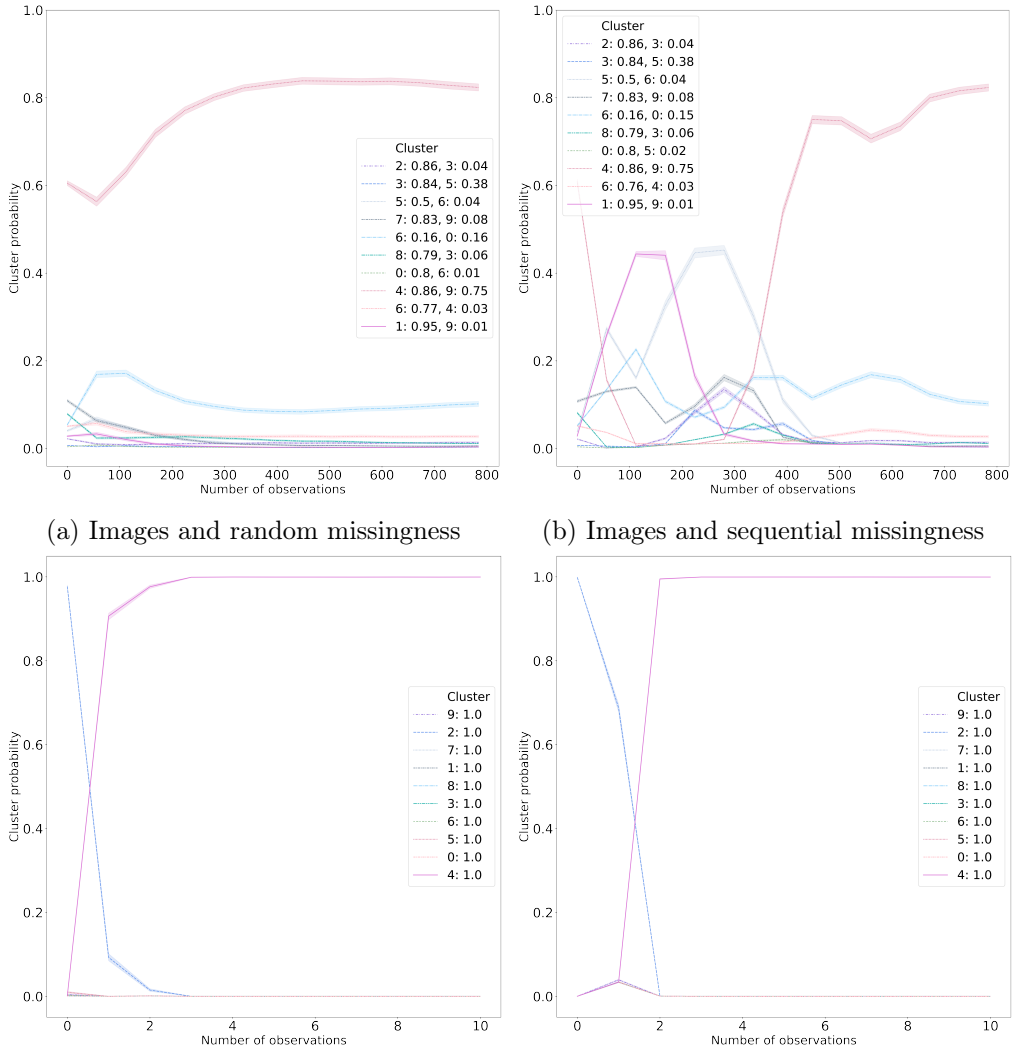
Figure 6.4.: The change of cluster probability of Partial VaDE (F) given an increasing number of observed values, demonstrated on the MNIST images and the categorical MNIST features. The models were trained with a random dropout rate 0-100% during training and evaluated in two settings: (1) random missingness, and (2) systematic missingness.

different, while the generated features describe the target always in the same way.

Moreover, the cluster probabilities obtained by Partial VaDE (F) are evenly distributed given unobserved data samples only. In contrast, the cluster probabilities obtained by Partial VaDE (M) show utterly different behaviours. As shown in Figure 6.5, high cluster probabilities are assigned while zero information is fed to the model. For example, in Figure 6.5a, we see that the probability of cluster 7 represented by the true labels 4 and 9 starts with high certainty, whereas all remaining cluster probabilities are similarly low.

A similar result can be observed for the MNIST feature evaluation in Figures 6.5c and 6.5d, where the random and sequential settings show matching results. In both cases, the model assigns a probability of nearly 100% to the ground-truth label 2, given only unobserved features as input. By increasing the number of observations (features) by one, we see a rapid increase in the probability of ground-truth label 4.

Overall, the results show that Partial VaDE (M) can reach high certainty for the correct cluster given only little information and therefore seems to handle missing data samples effectively.



(a) Images and random missingness

(b) Images and sequential missingness

(c) Features and random missingness

(d) Features and sequential missingness

Figure 6.5.: The change of cluster probabilities of Partial VaDE (M) given an increasing number of observed data points, demonstrated on the MNIST images and the categorical MNIST features. The models were trained with a random dropout rate 0-100% during training and evaluated in two settings: (1) random missingness, and (2) systematic missingness.

7. Conclusion

In this thesis, we studied the task of deep probabilistic clustering for multi-view and missing data. We gave a systematic overview of existing deep clustering methods for single-view and multi-view data, including discriminative and generative approaches. By this means, we reviewed and compared various models and discussed their strengths and weaknesses. Furthermore, we described how current deep clustering methods are built and how they differ in their network architecture, network layer, objective functions or learning setting. Also, we examined several existing methods to predict missing data points.

We raised the awareness for new deep clustering methods targeting multi-view and missing data as this research problem is relatively undiscovered and offers potential for new use cases. Further, we have motivated this challenge and proposed the Partial MV-VaDE, a method targeting this challenge by combining the existing deep probabilistic clustering model VaDE [3], which was initially designed for single-view data sets, with a novel method to predict missing data points, the Partial VAE with the PNP setting from [8]. We implemented the model in Python using the deep learning library PyTorch [59], conducted extensive experiments to assess the model's performance, and compared the results to several baseline methods.

The evaluation of Partial MV-VaDE shows that the model can handle two views of data and cope with missing data points only to a certain extend and does not necessarily outperform simpler baseline methods. Also, the model shows weak performance results and high variances across multiple runs. This issue of inconsistency in clustering scores can also be observed for the traditional VaDE model. We think that the clustering result depends on a good network and parameter initialization.

Whereas Partial MV-VaDE comprises some discrepancies that need to be resolved, it also offers potential research questions worth investigating. Therefore, a possibility for future work is to examine methods, which could achieve a more robust model training and consistent clustering results.

Another possibility for improving the clustering result of two data views is to investigate more suitable fusion techniques designed explicitly for merging data distributions. Gener-

ally, the network architecture can be further extended to enable clustering with more than two data views. By this means, another remaining challenge of deep multi-view clustering is that numerous existing methods use an individual network to extract modality-specific features. This could impact the computational costs when the number of views increases.

Moreover, we tried to ensure a flexible architecture so that single components can be exchanged for future work. For example, different pre-training models, e.g., a denoising model, can be incorporated, or an additional cluster regularizer can be introduced. Also, comparing Partial MV-VaDE's ability to impute missingness to other baseline approaches such as ZI [28] would be worth studying.

Finally, more robust approaches can be similarly extended to serve the purpose of multi-view and missing data clustering. Instead of using VaDE as the main building block, generative clustering methods based on a GAN, such as ClusterGAN [31], could serve as another promising basis.

Bibliography

[1] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[2] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *CoRR*, abs/1801.07648, 2018.

[3] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, pages 1965–1972. ijcai.org, 2017.

[4] Shuning Huang, Kaoru Ota, Mianxiong Dong, and Fanzhang Li. Multispectralnet: Spectral clustering using deep neural network for multi-view data. *IEEE Trans. Comput. Soc. Syst.*, 6(4):749–760, 2019.

[5] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognit.*, 83:161–173, 2018.

[6] Yann Le Cun and Françoise Fogelman-Soulié. Modèles connexionnistes de l'apprentissage. *Intellectica*, 2(1):114–143, 1987.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[8] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin, and Cheng Zhang. EDDI: efficient dynamic discovery of high-value information with partial VAE. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4234–4243. PMLR, 2019.

[9] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231. AAAI Press, 1996.

[11] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, pages 195–199, 1969.

[12] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognit. Lett.*, 31(8):651–666, 2010.

[13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[14] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[15] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[16] Chris H. Q. Ding and Xiaofeng He. K-means clustering via principal component analysis. In *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[17] Chris H. Q. Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and K-means clustering. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 521–528. ACM, 2007.

[18] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *IJCAI*, pages 1925–1931. IJCAI/AAAI Press, 2016.

[19] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[20] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759. ijcai.org, 2017.

[21] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3861–3870. PMLR, 2017.

[22] Lukas Miklautz, Lena G. M. Bauer, Dominik Mautz, Sebastian Tschiatschek, Christian Böhm, and Claudia Plant. Details (don't) matter: Isolating cluster information in deep embedded spaces. In *IJCAI*, pages 2826–2832. ijcai.org, 2021.

[23] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A. Bilmes. On deep multi-view representation learning: Objectives and optimization. *CoRR*, abs/1602.01024, 2016.

[24] Zhaoyang Li, Qianqian Wang, Zhiqiang Tao, Quanxue Gao, and Zhaohua Yang. Deep adversarial multi-view clustering network. In *IJCAI*, pages 2952–2958. ijcai.org, 2019.

[25] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, pages 19–26. IEEE Computer Society, 2004.

[26] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Min. Anal.*, 1(2):83–107, 2018.

[27] Fanghua Ye, Zitai Chen, Hui Qian, Rui Li, Chuan Chen, and Zibin Zheng. New approaches in multi-view clustering. *Recent Applications in Data Clustering*, 195, 2018.

[28] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *CoRR*, abs/1807.03653, 2018.

[29] Lukas Miklautz, Dominik Mautz, Muzaffer Can Altinigneli, Christian Böhm, and Claudia Plant. Deep embedded non-redundant clustering. In *AAAI*, pages 5174–5181. AAAI Press, 2020.

[30] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org, 2016.

[31] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *AAAI*, pages 4610–4617. AAAI Press, 2019.

[32] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156. IEEE Computer Society, 2016.

[33] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[34] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[35] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[36] Dongdong Chen, Jiancheng Lv, and Yi Zhang. Unsupervised multi-manifold clustering by learning deep representation. In *AAAI Workshops*, volume WS-17 of *AAAI Workshops*. AAAI Press, 2017.

[37] Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. Deep embedding network for clustering. In *ICPR*, pages 1532–1537. IEEE Computer Society, 2014.

[38] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014.

[39] Galen Andrew, Raman Arora, Jeff A. Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1247–1255. JMLR.org, 2013.

[40] Mahdi Abavisani and Vishal M. Patel. Deep multimodal subspace clustering networks. *IEEE J. Sel. Top. Signal Process.*, 12(6):1601–1614, 2018.

[41] Ga Wu, Justin Domke, and Scott Sanner. Conditional inference in pre-trained variational autoencoders via cross-coding. *CoRR*, abs/1805.07785, 2018.

[42] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1558–1567. PMLR, 2017.

[43] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *ICCV*, pages 5880–5888. IEEE Computer Society, 2017.

[44] Chih-Chung Hsu and Chia-Wen Lin. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Trans. Multim.*, 20(2):421–429, 2018.

[45] Lei Zhou, Xiao Bai, Dong Wang, Xianglong Liu, Jun Zhou, and Edwin R. Hancock. Latent distribution preserving deep subspace clustering. In *IJCAI*, pages 4440–4446. ijcai.org, 2019.

[46] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. pages 5747–5756. IEEE Computer Society, 2017.

[47] Liang Duan, Charu C. Aggarwal, Shuai Ma, and Saket Sathe. Improving spectral clustering with deep embedding and cluster estimation. In *ICDM*, pages 170–179. IEEE, 2019.

[48] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR*, abs/1611.02648, 2016.

[49] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *ICCV*, pages 6439–6448. IEEE, 2019.

[50] Dominik Maria Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Trans. Inf. Theory*, 49(7):1858–1860, 2003.

[51] Warith Harchaoui, Pierre-Alexandre Mattei, and Charles Bouveyron. Deep adversarial gaussian mixture auto-encoder for clustering. 2017.

[52] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *ICLR (Poster)*, 2016.

[53] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180, 2016.

[54] Pan Zhou, Yunqing Hou, and Jiashi Feng. Deep adversarial subspace clustering. In *CVPR*, pages 1596–1604. IEEE Computer Society, 2018.

[55] Handong Zhao, Zhengming Ding, and Yun Fu. Multi-view clustering via deep matrix factorization. In *AAAI*, pages 2921–2927. AAAI Press, 2017.

[56] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR (2)*, pages 1735–1742. IEEE Computer Society, 2006.

[57] Zhenyu Huang, Joey Tianyi Zhou, Xi Peng, Changqing Zhang, Hongyuan Zhu, and Jiancheng Lv. Multi-view spectral clustering network. In *IJCAI*, pages 2563–2569. ijcai.org, 2019.

[58] Wenbo Gong, Sebastian Tschiatschek, Sebastian Nowozin, Richard E. Turner, José Miguel Hernández-Lobato, and Cheng Zhang. Icebreaker: Element-wise efficient information acquisition with a bayesian deep latent gaussian model. In *NeurIPS*, pages 14791–14802, 2019.

[59] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[60] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[61] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[62] Harold W. Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming*, pages 29–47. Springer, 2010.

[63] Xuan Vinh Nguyen, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, 2010.

[64] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[65] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[66] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind K. Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *SenSys*, pages 127–140. ACM, 2015.

[67] José Costa Pereira, Emanuele Coviello, Gabriel Doyle, Nikhil Rasiwasia, Gert R. G. Lanckriet, Roger Levy, and Nuno Vasconcelos. On the role of correlation and abstraction in cross-modal multimedia retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):521–535, 2014.

[68] Martijn van Breukelen, Robert P. W. Duin, David M. J. Tax, and J. E. den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

[69] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[70] Xiao Cai, Hua Wang, Heng Huang, and Chris H. Q. Ding. Joint stage recognition and anatomical annotation of *drosophila* gene expression patterns. *Bioinform.*, 28(12):16–24, 2012.

[71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[73] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010.

Appendices

A. Abstract

A.1. Abstract

In this thesis, we aim to raise the importance of deep clustering methods for multi-view and missing data. We combine two state-of-the-art methods, the Variational Deep Embedding (VaDE) initially designed for deep probabilistic clustering of single-view data and the Partial Variational Autoencoder (VAE) with the Pointnet Plus (PNP) structure, a method to predict missing data points. More precisely, we extend VaDE by an additional VAE and examine two fusion techniques for creating a shared distribution between two data views. To handle missingness in both views, we integrate the Partial VAE with PNP, which enables the definition of a partial clustering objective that depends on observed data samples only. As a result, we propose the Partial Multi-View Variational Deep Embedding (Partial MV-VaDE), a deep probabilistic clustering model targeting multi-view and missing data. We evaluate the model's performance in extensive experiments with numerous multi-view data sets for which we generate different amounts of missingness. We observe the model's changes in cluster probabilities in more detail and compare the clustering results to several baseline methods.

A.2. Kurzfassung

In dieser Arbeit heben wir die Bedeutung von Deep Clustering Methoden für multimodale und unvollständige Datensätze hervor. Dabei kombinieren wir zwei bestehende Methoden, das probabilistische Deep Clustering Modell für unimodale Daten, Variational Deep Embedding (VaDE), sowie den Partial Variational Autoencoder (VAE) mit der Pointnet Plus (PNP) Struktur, welcher erlaubt auch unvollständige Daten zu berücksichtigen. Wir erweitern die Architektur von VaDE um einen weiteren VAE und untersuchen zwei Methoden um multimodale Datensätze zu verknüpfen. Um mit fehlenden Datenpunkten in beiden Datenmodalitäten umgehen zu können, integrieren wir den Partial VAE mit dem PNP. Mit dieser Vorgehensweise definieren wir ein partielles Clustering-Ziel, welches ausschließlich von beobachteten Datenpunkten abhängt und stellen das Partial Multi-View Variational Deep Embedding (Partial MV-VaDE) Modell vor. Die Performance des Modells wird anhand von umfangreichen Experimenten an multimodalen Datensätzen, für welche fehlende Datenpunkte generiert werden, evaluiert. Außerdem schauen wir uns die Cluster-Wahrscheinlichkeiten im Detail an und vergleichen die Resultate mit verschiedenen Baseline-Methoden.

B. Glossary

B.1. Notation and Symbols

|N| Number of variables in a vector

|M| Number of variables in a second vector

$\boldsymbol{x}$ Set of random variables

$\boldsymbol{x}^{(1)}$ Set of random variables denoting the first data view describing instance $\boldsymbol{x}$

$\boldsymbol{x}^{(2)}$ Set of random variables denoting a second data view describing instance $\boldsymbol{x}$

$\boldsymbol{x}_O^{(1)}$ Subset of $\boldsymbol{x}^{(1)}$ being observed

$\boldsymbol{x}_Q^{(2)}$ Subset of $\boldsymbol{x}^{(2)}$ being observed

$\boldsymbol{x}_U^{(1)}$ Subset of $\boldsymbol{x}^{(1)}$ being unobserved

$\boldsymbol{x}_V^{(2)}$ Subset of $\boldsymbol{x}^{(2)}$ being unobserved

$\boldsymbol{z}$ Latent variables

c Cluster

$\boldsymbol{\mu}$ Mean of a probabilistic distribution, e.g., Gaussian

$\boldsymbol{\sigma}^2$ Variance of a probabilistic distribution, e.g., Gaussian

$\hat{x}_i$ Reconstruction of data sample x_i

x_i Data sample

|K| Number of variables in a vector

$D(x_i)$ The discriminator's probability estimate of x_i

$\mathbb{E}_{x_i}$ Expected value over all true data distributions

$G(z_i)$ Data samples generated by the GAN's generator

$\mathbb{E}_{z_i}$ Expected value given all generated data samples

r_j Centroid of j^{th} cluster

|C| Number of clusters

$q(x_i)$ Approximated distribution of data sample x_i

$p(x_i)$ True distribution of data sample x_i

$\boldsymbol{k}_i$ Set of k-nearest neighbors of a data sample x_i

$\mathcal{A}$ Similarity measure

$\mathcal{L}_{net}$ Network loss

$\mathcal{L}_{cluster}$ Cluster loss

λ Regularization parameter defining the trade-off between $\mathcal{L}_{net}$ and $\mathcal{L}_{cluster}$

$p(\boldsymbol{z})$ Probability distribution of latent variables $\boldsymbol{z}$

$p(\boldsymbol{x}|\boldsymbol{z})$ Likelihood of data samples $\boldsymbol{x}$ given latent variables $\boldsymbol{z}$

$p(\boldsymbol{z}|\boldsymbol{x})$ True probability distribution of latent variables $\boldsymbol{z}$ given data samples $\boldsymbol{x}$

$p(\boldsymbol{x})$ Probability distribution of data samples $\boldsymbol{x}$

$q(\boldsymbol{z}|\boldsymbol{x})$ Approximated probability distribution of latent variables $\boldsymbol{z}$ given data samples $\boldsymbol{x}$

$\boldsymbol{\epsilon}$ Noise variable following a Gaussian distribution $\mathcal{N}(0, 1)$

$p(c)$ Probability distribution of cluster c

$p(\boldsymbol{z}|c)$ Probability distribution of latent variables $\boldsymbol{z}$ given cluster c

$\boldsymbol{\pi}$ Prior cluster probability for cluster c

$Cat(\boldsymbol{\pi})$ Categorical distribution of the prior cluster probability for cluster c

$\boldsymbol{\mu}_c$ Mean of probability distribution corresponding to cluster c

$\boldsymbol{\sigma}_c^2$ Variance of probability distribution corresponding to cluster c

$\boldsymbol{I}$ Identity matrix

$f(\boldsymbol{z};\boldsymbol{\theta})$ Probabilistic decoder network

$\boldsymbol{\mu}_x$ Mean of probability distribution of data samples $\boldsymbol{x}$

$\boldsymbol{\sigma}_x^2$ Variance of probability distribution of data samples $\boldsymbol{x}$

$p(\boldsymbol{z}, c|\boldsymbol{x})$ True posterior distribution

$q(\boldsymbol{z}, c|\boldsymbol{x})$ Approximated posterior distribution

$g(\boldsymbol{x};\boldsymbol{\phi})$ Probabilistic encoder network

e_j Unknown embedding in *PNP* [8] optimized during training

$\boldsymbol{d}$ Resulting encoding of the *PNP* set function [8]

L Dimension of latent space

y_i Ground-truth label of data sample x_i

c_i Cluster assignment

m Possible mapping between cluster and label

B.2. Acronyms

DNN Deep Neural Network

VaDE Variational Deep Embedding

VAE Variational Autoencoder

PNP *Pointnet Plus*

AE Autoencoder

SAE Stacked Autoencoder

GAN Generative Adversarial Network

KL Kullback Leibler

ELBO Evidence Lower Bound

GMM Gaussian Mixture Model