Radboud University Nijmegen

Faculty of Science

Multi-scale graph-based clustering of images using Markov theory

Thesis MSc Data Science

Author:
Anneloes Ernest
s4579259

Supervisor:
Elena Marchiori

Second reader:
Twan van Laarhoven

12 July, 2021

Abstract

In this thesis, image clustering at multiple scales using the Markov clustering framework is experimentally analysed. In particular, the usability of variational autoencoder's (VAE) latent space representations and graph construction methods are investigated, as well as the ability of the Markov clustering framework to detect hierarchically meaningful clusters across different levels of coarseness in the considered image domain. Results of extensive experiments on the MNIST data set show that the latent space representation of the VAE is indeed suitable for multi-scale clustering. Results on the Fashion-MNIST data set tell a different story but were still useful in investigating the relationships of the clusterings across the detected hierarchy.

Contents

Chapter 1

Introduction

In the last decades, with the rise of the internet, the possibility to share and access data has grown tremendously. For image, picture, and photo data, social media and fora have especially been great contributors in the increasing number of data available online. The easy access to such a large number of images is not only beneficial for personal use, it has also enabled the creation of large image data sets for research. However, many of these images lack good and consistent labeling in order to be directly used for research. Image annotation [1, 2], image segmentation and classification [3], content-based image retrieval [4, 5], are all problems in the data science research area that need a way to identify which data points belong to the same label or category. This unsupervised grouping of data points is known as clustering..

There are many ways to cluster data. The K-means algorithm [6–9] is a very popular method which uses a K number of cluster centroids and assigns the data points to their closest centroid. Other methods like DBSCAN [10] and CLIQUE [11] use the density of the data points to identify clusters. A common problem with these methods is that they need a fixed number of clusters prior to clustering. If the number is unknown, many repetitions of the experiments need to be done with different hyper-parameters in order to determine the best number of clusters.

For graph-like data, where a set of vertices is connected to each other through edges, specific graph-based clustering methods have been introduced. The Normalized Cut (N-Cut) algorithm [12] and the Laplacian Eigenmap [13] are examples of graph-based clustering methods. These methods have the ability to reveal the modular structure in the data and allow detection of clusters at different levels of coarseness.

Graph-based clustering is usually applied to graphs, or networks, already existing in the real-world such as Facebook's friend networks. Nevertheless, it is possible to represent non-graph-like data as a graph. The graph representation of the data opens up the use of a Markov chain for clustering. Markov clustering works by continuously taking random walks from one vertex to another. The longer this process goes on for, the more (sub-)clusters the algorithm finds, and finding clusterings at varying levels of detail.

To make graph representations of image data, we need a distance or similarity measure to base the adjacency matrix on. Common similarity measures include the Euclidean distance and cosine similarity. For image data specifically, it might be interesting to take spacial correlations into account and use the image Euclidean distance (IMED) [14]. Still, image data is very high-dimensional with every pixel representing one dimension. Applying similarity measures on all image data pairs in such a high-dimensional space is time and memory expensive. The similarity measure itself also becomes difficult to interpret since small deviations in pixel values can cause large differences. These problems with high-dimensional computations are also known as the curse of dimensionality. Consequently, dimensionality reduction is a common tool in image processing, with principal component analysis (PCA) [15] as one of the most well known. Another solution is the use of visual descriptors for feature extraction, like SIFT [16], HOG [17], SURF [18] and CENTRIST [19]. In more recent years, latent space representations from autoencoders have shown promising results [20–24].

There are many challenges when it comes to image clustering, especially when using a graph representation of the image data set. Several attempts have been promising [19, 25, 26]. However, hardly any research using clustering methods has been done into multi-scale detection of image clusterings. The fact that most image data sets only contain one label per image does not help as it does not offer any information about lower or higher level category membership. For example, take the Fashion-MNIST [27] data set. Some of the clothing item classes are part of the same higher level category. For instance, 'sandals', 'sneakers' and 'ankle boot' are all types of 'shoes' and it would still be a relatively good clustering if those items were clustered together.

A recent paper by Liu & Barahona (2020) [28] applied Markov Stability for graph-based clustering on multiple attribute data sets from the UCI repository. They specifically used Markov Stability to avoid the need to set a parameter for the number of clusters. The Markov Stability framework also alleviated the parameter sensitivity of the applied graph construction method, in their case the parameters of the continuous k-nearest neighbours (CkNN) graph algorithm [29]. The experiments showed that the diffusion process of the Markov Stability framework acts as a resolution parameter, without needing to set the number of clusters manually. For a geometric example, the clusterings also showed a hierarchical relationship across the levels in the detected hierarchy.

In this paper, I will investigate the usability of variational autoencoder's latent space representations and graph construction methods for graph clustering on image data using the Markov clustering framework. Additionally, I will look into the ability to detect hierarchically meaningful clusters across time scales. To this end, I will use both the MNIST and Fashion-MNIST data sets to test the performance.

The following research questions will be answered in this thesis:

- **RQ1**: Does the latent space representation provide better clustering results for image clustering than standard similarity measures, like the Euclidean distance, the image Euclidean distance (IMED) and cosine distance, directly applied to the image pixels?
  - In the case of the similarity measures directly applied to the image pixels, which measure performs best?
- **RQ2**: Which graph construction method, namely regular k-nearest neighbours (kNN) or continuous k nearest neighbours (CkNN), is more beneficial with the Markov clustering framework?
- **RQ3**: Are the hierarchies detected with Markov clustering semantically meaningful?
  - Which classes are clustered together in clusterings with a lower number of detected clusters than the true number of classes?
  - Which classes have the strongest within class similarity?

RQ3 is most relevant to the Fashion-MNIST data set since some categories are more semantically related ('sneaker' and 'ankle boot', or 'shirt' and 'pullover') than other categories ('trouser' and 'sneaker', or 'shirt' and 'bag'). For the MNIST data set I am merely looking at which digits are more similar. For example, the expectation is that the digits ‘1’ and ‘7’ are more similar to each other than they are to any other digit.

Chapter 2

Background

Research into image clustering usually consist of two stages: feature learning and clustering. These stages can be optimised separately, where first the features are learned and then those features are used in the clustering framework [5, 30–32]. To further improve the total clustering performance, research has shifted to a joint effort in learning representations while simultaneously updating and optimising the clustering algorithm [20, 33–38].

2.1 Dimensionality reduction

Image feature extraction and image representation learning are common methods to reduce dimensionality but also a way to make the clustering performance better. By only comparing relevant and distinguishable features of the images, two similar images might have an even stronger connection to each other than to other images. This greatly increases the likelihood these two images are clustered together.

There are many kinds of features one can extract from image data, each with differing complexity and suitability for clustering. One of the best known dimensionality reduction methods is principal component analysis (PCA) [15], where orthogonal linear transforms are learned to capture the direction of variance in the data set. The top k components that capture the most variance are used as a new basis on which to represent the data.

2.1.1 Neural networks

Deep learning has already proven to be very successful in image classification tasks [39]. Neural networks are more and more often used in solutions to clustering problems as well, usually as a means to learn lower-dimensional representations. The complex architecture of neural networks allows for learning non-linear representations of the images. The training of a standard deep neural network is a supervised process where labels are known during training to compute the loss function. When there is a lack of good and consistent labeling of the images, regular neural networks are not an ideal solution. However, autoencoder neural network structures do use loss functions that do not depend on the data labels.

2.1.2 Autoencoders

Autoencoders have been widely used in image clustering to reduce dimensionality. An autoencoder consists of two neural network structures: the encoder and the decoder.

The encoder reduces the input to a lower-dimensional representation, better known as the latent space representation. The decoder is fed the output of the encoder and tries to reconstruct the input image. The idea is that if the reconstruction is similar to the input image, the latent space representation must be informative enough for the decoder to decipher its unique original image back. However, when two images are similar in their original state, it does not automatically mean that the latent space representation of those two images is also similar [20, 22, 40, 41].

The disconnection between the similarity in the input data and their latent space representation is the main reason for a unified method where both the latent space representations and cluster parameters are learned simultaneously. Song et al. [20] trained an autoencoder for latent space image representation which iteratively updated the K-means centroids. By adjusting the autoencoder loss to also minimize the distance between the latent space points to the cluster centroids, the authors were able to learn a representation more suitable for clustering. Xie et al. [31] proposed deep embedded clustering (DEC), which simultaneously learns a set of K-means cluster centers in the latent space of a deep neural network (DNN) and the parameters of the DNN. The DNN is initialised with the encoder of a trained stacked autoencoder and jointly optimised with the cluster centers using Kullback-Leibler (KL) divergence minimization to an auxiliary target distribution. A more principled way to learn a latent space representation that keeps the similarity between the input images intact is the use of variational autoencoders (VAEs) as they inherently have this capability.

2.1.3 Variational autoencoders

The idea behind VAEs stems from Bayesian inference problems where the posterior probability of an event is calculated based on a prior distribution and a likelihood function. By assuming the input data comes from a particular data distribution and using this prior distribution in back-propagation, VAEs are able to learn a latent space representation adhering to the distribution of the input data. This form of regularisation enforces that two images that are similar in the input data are also similar in their latent space representations, if not more similar. Dilokthanakul et al. [22] used a mixture of Gaussians as the prior of the VAE and use the latent space representation to assign cluster labels.

The loss function for training a VAE consists of two parts. The first part of the loss function describes the difference between the approximate Gaussian posterior given the input distribution $q(z|x)$ and the standard Gaussian prior $p(z)$. The difference between these two distributions is calculated using Kullback-Leibler (KL) divergence, which calculates the difference between the cross-entropy of p and q and the entropy of q:

$$D_{KL}\left(q\left(z\middle|x\right), p\left(z\right)\right) = -\int q\left(z\middle|x\right) \log p\left(z\right) dz - \left(-\int q\left(z\middle|x\right) \log q\left(z\middle|x\right) dz\right) \tag{2.1}$$

The reparametrization trick is applied and the prior is sampled from a fixed standard Gaussian $p\left(z\right) \sim \mathcal{N}\left(0, 1\right)$. We assume the posterior approximates a Gaussian function $q\left(z\middle|x\right) \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and we want this posterior to be as close as possible to the prior $\mathcal{N}\left(0, 1\right)$. Therefore, we can sample the latent space z from the fixed Gaussian $\mathcal{N}\left(0, 1\right)$ and construct $z = \mu\left(x\right) + \sigma\left(x\right) \cdot \epsilon$. By random (ϵ) sampling z, back-propagation is possible with simple gradient descent.

In Eq. 2.2 and Eq. 2.3 the formulas for $q\left(z\middle|x\right) \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and its log equivalent are stated respectively.

$$\begin{aligned} q\left(z \middle| x\right) &\sim \mathcal{N}\left(\mu, \sigma^2\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(x-\mu\right)^2}{2\sigma^2}\right). \end{aligned} \tag{2.2}$$

$$\begin{aligned} \log q\left(z \middle| x\right) &= \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(\exp\left(-\frac{\left(x-\mu\right)^2}{2\sigma^2}\right)\right) \\ &= \log\left(\left(2\pi\sigma^2\right)^{-\frac{1}{2}}\right) + \left(-\frac{\left(x-\mu\right)^2}{2\sigma^2}\right) \\ &= -\frac{1}{2}\log\left(2\pi\sigma^2\right) - \frac{\left(x-\mu\right)^2}{2\sigma^2}. \end{aligned} \tag{2.3}$$

In Eq. 2.4 and Eq. 2.5 below, the formulas for $p\left(z\right) \sim \mathcal{N}\left(0,1\right)$ and its log equivalent are stated respectively.

$$\begin{aligned} p\left(z\right) &\sim \mathcal{N}\left(0,1\right) \\ &= \frac{1}{\sqrt{2\pi 1}} \exp\left(-\frac{\left(x-0\right)^2}{2\cdot 1^2}\right) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right). \end{aligned} \tag{2.4}$$

$$\begin{aligned} \log p\left(z\right) &= \log\left(\frac{1}{\sqrt{2\pi}}\right) + \log\left(\exp\left(-\frac{1}{2}x^2\right)\right) \\ &= \log\left(\left(2\pi\right)^{-\frac{1}{2}}\right) + \left(-\frac{1}{2}x^2\right) \\ &= -\frac{1}{2}\log\left(2\pi\right) - \frac{1}{2}x^2 \\ &= -\frac{1}{2}\left(\log\left(2\pi\right) + z^2\right). \end{aligned} \tag{2.5}$$

Circling back to the KL divergence (Eq. 2.1), using Eq. 2.5, the cross-entropy of p and q, can now be written as:

$$\begin{aligned} -\int q\left(z \middle| x\right) \log p\left(z\right) dz &= -\int q\left(z \middle| x\right) \left(-\frac{1}{2}\left(\log\left(2\pi\right) + z^2\right)\right) dz \\ &= \frac{1}{2}\log\left(2\pi\right) \int q\left(z \middle| x\right) dz + \frac{1}{2}\int z^2 q\left(z \middle| x\right) dz \\ &= \frac{1}{2}\log\left(2\pi\right) + \frac{1}{2}\left(\mu^2 + \sigma^2\right) \\ &= \frac{1}{2}\left(\log\left(2\pi\right) + \left(\mu^2 + \sigma^2\right)\right), \end{aligned} \tag{2.6}$$

where the integral over a distribution is always one: $\int q\left(z \middle| x\right) dz = 1$, and the expectation over the square of a random variable is equivalent to the sum of square of mean and variance: $\int z^2 q\left(z \middle| x\right) dz$. Now using Eq. 2.3. the entropy part of the KL divergence can be written as follows:

$$\begin{aligned}
-\int q\left(z|x\right)\log q\left(z|x\right)dz &= -\int q\left(z|x\right)\left(-\frac{1}{2}\log\left(2\pi\sigma^2\right)-\frac{\left(z-\mu\right)^2}{2\sigma^2}\right)dz \\
&= \frac{1}{2}\log\left(2\pi\sigma^2\right)\int q\left(z|x\right)dz+\frac{1}{2\sigma^2}\int\left(z-\mu\right)^2 q\left(z|x\right)dz \\
&= \frac{1}{2}\log\left(2\pi\sigma^2\right)+\frac{1}{2\sigma^2}\sigma^2 \\
&= \frac{1}{2}\log\left(2\pi\right)+\frac{1}{2}\log\left(\sigma^2\right)+\frac{1}{2} \\
&= \frac{1}{2}\left(log\left(2\pi\right)+\log\left(\sigma^2\right)+1\right).
\end{aligned} \tag{2.7}$$

Combining Eq. 2.6 and Eq. 2.7 gives us the KL divergence formula:

$$\begin{aligned}
D_{KL}\left(q\left(z|x\right),p\left(z\right)\right) &= \frac{1}{2}\left(\log\left(2\pi\right)+\left(\mu^2+\sigma^2\right)\right)-\frac{1}{2}\left(log\left(2\pi\right)+\log\left(\sigma^2\right)+1\right) \\
&= -\frac{1}{2}\left(1-\mu^2-\sigma^2+\log\left(\sigma^2\right)\right).
\end{aligned} \tag{2.8}$$

In practice, however, it is preferable to use the exponent instead of the logarithm as it is more numerically stable while computing. Therefore, the final KL divergence formula used in this thesis for part of the loss function is:

$$\begin{aligned}
\mathcal{L}_{KL} &= D_{KL}\left(q\left(z|x\right),p\left(z\right)\right) \\
&= -\frac{1}{2}\left(1-\mu^2-\sigma^2+\exp\left(\sigma^2\right)\right).
\end{aligned} \tag{2.9}$$

The reconstruction loss is also used as part of the training process and is calculated with the negative loglikelihood (NLL):

$$\begin{aligned}
\mathcal{L}_{NLL} &= = -E_{z\sim q}\left[\log p\left(x|z\right)\right] \\
&= \sum_{i=1}^{MN} x_i\log y_i+\left(1-x_i\right)\log\left(1-y_i\right),
\end{aligned} \tag{2.10}$$

where x are the encoder's input image pixels and y are the decoder's output image pixels. Note that the input of the decoder is the sampled latent space representation.

2.2 (Dis)similarity

As mentioned previously in the Introduction, there are many measures to compute the similarity. The chosen similarity also depends on the kind of data that is being used. When comparing two vectors of features, some standard measures like the Euclidean distance and cosine distance come to mind. These are actually more a measure of dissimilarity since a higher value represents a larger distance between two nodes, thus implying a lower level of similarity.

$$D_{eucl}^2\left(x,y\right)=\sum_{i=1}^{N}\left(x_i-y_i\right)^2 \tag{2.11}$$

$$D_{cos}(x,y) = 1 - \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2}\sqrt{\sum_{i=1}^{N} y_i^2}} \tag{2.12}$$

In a high-dimensional feature space, the use of the Euclidean distance is not as ideal since small deviations can result in large differences, making the measure difficult to interpret. The cosine distance is a bit more suited in this case as it stays within the range $[-1, 1]$.

The Euclidean distance and cosine distance can also be applied to images. The image is flattened and each pixel represents a feature in the vector. For image data specifically, it might be interesting to take spacial relationships of the pixels into account and use the image Euclidean distance (IMED) [14]. IMED adjusts the metric coefficient matrix (G), which is used to describe the distance and angle from one point on a surface to another. Let P_i, P_j, for $i, j = 1, 2, \ldots, MN$ be pixels on a MxN image's grid. The distance between two pixel locations is denoted as $|P_i - P_j|$. If we denote the location of pixel P_i as (k, l) and the location of pixel P_j as (k', l'), the pixel distance may be calculated as follows:

$$|P_i - P_j| = \sqrt{(k - k')^2 + (l - l')^2}. \tag{2.13}$$

With f describing the dependency between two pixel locations, the metric coefficients of the regular Euclidean distance are defined as:

$$g_{ij} = f\left(|P_i - P_j|\right) = \begin{cases} 1 & \textit{if } i = j \\ 0 & \textit{otherwise} \end{cases}. \tag{2.14}$$

Looking at the metric coefficients of the Euclidean distance, only the pixels in the same position (where $i = j$) are compared and thus no information about the surrounding pixels is taken into account. Where IMED differs from the regular Euclidean distance is that the metric coefficients are constructed using a Gaussian function:

$$g_{ij} = f\left(|P_i - P_j|\right) = \frac{1}{2\pi\sigma^2} \exp\left\{ -\frac{|P_i - P_j|^2}{2\sigma^2} \right\}. \tag{2.15}$$

Therefore, the image Euclidean distance of two images $x = (x_1, x_2, \ldots, x_{MN})$ and $y = (y_1, x_2, \ldots, y_{MN})$ of size MxN can be computed as follows:

$$\begin{aligned} D_{IME}^2(x,y) &= \frac{1}{2\pi\sigma^2} \sum_{i,j=1}^{MN} \exp\left\{ -\frac{|P_i - P_j|^2}{2\sigma^2} \right\} (x_i - y_i)(x_j - y_j) \\ &= \sum_{i,j=1}^{MN} g_{ij} (x_i - y_i)(x_j - y_j). \end{aligned} \tag{2.16}$$

2.3 Graph construction

Data that is not graph-like can be transformed into a graph representation by interpreting the similarity matrix of the data as an adjacency matrix. In such a constructed graph, the vertices represent the data points and the weight on the edges between nodes represent the similarity between two data points. It is also possible to have an unweighted graph where the simple existence of an edge between two nodes represents they are more similar to each other than to nodes they are not connected to.

The constructed graph is usually highly dependent on the chosen (dis)similarity measure and the used construction method. Multiple distance measures for image similarity were mentioned in the previous paragraph, namely the Euclidean distance, cosine distance and the image Euclidean distance. One of these similarity measures is used in combination with a graph construction method to create a graph's adjacency matrix. For simplicity, all constructed graphs have unweighted and undirected edges.

A popular graph construction method is the k-nearest neighbour (kNN) graph. In kNN, each data point is connected to its k most similar neighbours based on the chosen distance measure. In mathematical terms, two data points x and y are connected if:

$$\begin{aligned} &d(x,y) \leq d(x,x_k) \\ &or \\ &d(x,y) \leq d(y,y_k), \end{aligned} \tag{2.17}$$

where x_k and y_k represent the k-th nearest neighbours of x and y respectively.

The kNN graph construction algorithm has some issues in terms of efficiency and scalability [42]. Usually, approximate methods for kNN graph construction are used when dealing with larger data sets because of the large computational cost [42].

An alternative version of kNN is the Continuous k-nearest neighbours (CkNN) algorithm [29]. By adding a free floating parameter to the equation, the maximum distance to connect two vertices can be continuously altered instead of discretely:

$$d(x,y) \leq \delta\sqrt{d(x,x_k)\,d(y,y_k)}. \tag{2.18}$$

As you can see, the distance is still dependent on the number of nearest neighbours chosen prior, but because of the δ parameter the distance can easily be increased or decreased. The δ parameter allows more flexibility in the creation of the graphs. For simplicity, this value is fixed to 1. The distance at which two nodes are connected is therefore smaller than the largest distance, either from x to x_k or from y to y_k. This means that the resulting CkNN graph has less connections than the kNN graph with the same number of k-neighbours.

A minimal spanning tree (MST) is computed and used to ensure all emerging subgraphs are connected. The connectivity is necessary for the applicability of the Markov clustering framework since taking a random walk to a disconnected group of vertices is not possible.

2.4 Markov clustering framework

The graph representation of the data opens up the use of a Markov chain for clustering. A Markov chain is a stochastic model that describes a process where each next state only depends on a transition from the current state to a neighbouring state. In a weighted graph structure, the vertices represent the states and the edge weights can be used to define the transition probabilities. The more similar two data points are, the higher the transition probability is of moving from that one vertex to the other. In unweighted graphs, the transition to the next vertex is chosen randomly among neighbouring vertices. Consequently, the vertices within the same cluster as the initial vertex are more likely to be visited early in the diffusion process. Also, when reaching a vertex in another cluster, it is more likely to visit other vertices in that cluster first before leaving again. This process of continuously taking random walks is what constitutes a Markov process.

Let A be the adjacency matrix of an undirected and unweighted graph G. The Markov matrix $\mathcal{T}$, or transition matrix, of G equals the column normalized adjacency matrix A. Let d be a diagonal matrix where each element on the diagonal is the sum of the elements in the corresponding column in A:

$$d_{ij} = \begin{cases} \sum_{i=0}^{N} A_{ij} & if\ i = j \\ 0 & otherwise \end{cases}. \tag{2.19}$$

Then the Markov matrix $\mathcal{T}$ can be computed by multiplying A with the inverse of d:

$$\mathcal{T} = Ad^{-1}. \tag{2.20}$$

The probability of ending up in a particular vertex after a single random walk from one random node to a neighbouring one is calculated by squaring the transition probability matrix $\mathcal{T}$: $\mathcal{T}\text{x}\mathcal{T} = \mathcal{T}^2$. For a second random walk, we simply multiply the previously calculated probabilities with the transition probabilities again: $\mathcal{T}^2\text{x}\mathcal{T}$, etc. By multiplying the matrix with itself over and over again until hardly any change in the resulting transition matrix $\mathcal{T}(t)$ after t steps has occurred, we can identify clusters in the graph.

A common word in graph theory is flow, which in terms of the Markov clustering framework describes the location and direction of the random walks in the graph. Van Dongen [43] defines Markov graph clustering based on two phenomena: the disappearance of flow on edges between sparsely connected dense regions and the creation of flow within dense regions. The creation of flow is simulated with expansion of the transition matrix: $\mathcal{T}\text{x}\mathcal{T}$. However, to make the algorithm converge faster, small transition probabilities are made even smaller by using an extra inflation step. Inflating the transition matrix means that first the column probabilities are multiplied with a factor r and then again column-normalized. Let Γ_r be the inflation with power r, then the inflation step can be formalized as:

$$(\Gamma_r \mathcal{T})_{pq} = \frac{(\mathcal{T}_{pq})^r}{\sum\limits_{i=1}^{N} (\mathcal{T}_{iq})^r}, \tag{2.21}$$

To decrease the computation time, van Dongen [43] also introduces pruning after a certain number of steps. If some of the transition probabilities after an expansion+inflation step are below a particular threshold, these values are set to zero. It is also possible to prune even more strictly by only allowing every node to have a maximum number of neighbours after each step, of course only keeping those with the highest transition probability. To allow more precise results, only the first pruning measure is adopted in the rest of this thesis.

As mentioned earlier, the expansion step allows flow through the network and connects regions with each other, making the number of detected clusters at convergence smaller. Inflation, on the other hand, has the opposite effect where flow is contracted. A higher inflation power r increases the contraction of information flow and consequently increases the number of detected clusters at convergence. A higher inflation power r also helps to converge the algorithm sooner since small values become smaller faster. To maintain a balance between allowing flow and contracting flow such that the right number of clusters is detected, multiple expansion steps can be performed first before applying a single inflation step.

In this thesis, I will experiment with graph construction methods kNN and CkNN both directly applied to the pixel values and to the variational autoencoder latent space representations. The Markov clustering algorithm described by [43] is used to find clusterings at different levels of coarseness to see how the elements correlate to each other across the hierarchy. The MNIST and Fashion-MNIST data sets are used as our image data sets for the experiments.

Chapter 3

Methods

3.1 Data

For the experiments, the MNIST [44] and the Fashion-MNIST [27] data sets are used. The MNIST is a classic handwritten digit data set. The Fashion-MNIST data set is a clothing item data set. Both are comprised of a training set of 60.000 images and a test set of 10.000 images, they also each have 10 classes. All images have a size of 28x28 pixels and are represented in grayscale. For Fashion-MNIST specifically, the 10 classes can reasonably be divided in 5 classes based on our semantic interpretation of which clothing items are similar [41]:

1. Short-sleeved: {T-shirt/top, Dress}
2. Long-sleeved: {Pullover, Coat, Shirt}
3. Bottoms: {Trouser}
4. Shoes: {Sandal, Sneaker, Ankle boot}
5. Accessories: {Bag}

So, for the Fashion-MNIST data we assume a clustering hierarchy with one level consisting of the 10 original classes and the other level of the 5 groups mentioned above. This new division helps to calculate the cluster performance and to investigate the cluster assignments when a lower number of clusters is detected than the true number of classes. For MNIST, I am more interested in which digits are clustered together by the Markov clustering framework in clusterings with a lower numbers of clusters than 10, and which digits are therefore more similar to each other. Observe that the Markov clustering framework is applied in an adjusted way, where the performances are mainly based on a fixed number of 10 or 5 clusters in the clusterings. More on how exactly the resulting clusterings are compared can be found in Methods section 3.3.3 and section 3.3.4.

3.2 Clustering metrics

The goodness of the resulting clusterings is computed using the normalized mutual information (NMI) [45] and the adjusted rand index score (ARI) [46]. Both compute how well the predicted clustering approximates the true clustering. Also, NMI and ARI do not need similar labels for similar clusters since the measures are based on which elements are grouped together rather than which exact label each element has. The

		Clustering Y				
	Class	y_1	y_2	$\dots$	y_S	Sums
Clustering C	c_1	n_{11}	n_{12}	$\dots$	n_{1S}	a_1
	c_2	n_{21}	n_{22}	$\dots$	n_{2S}	a_2
	.	.	.		.	.
	.	.	.		.	.
	c_R	n_{R1}	n_{R2}	$\dots$	n_{RS}	a_R
	sums	b_1	b_2	$\dots$	b_S	

Table 3.1: Contingency matrix of clusterings Y and C. Each element n_{ij} represents the number of elements present in both cluster C_i and cluster Y_j.

variation of information (VI) [47] is computed for the multi-scale clusterings to allow self-detection of a good number of clusters.

3.2.1 Normalized mutual information

The mutual information (MI) is a measure to determine how much information the observation of one random variable can give about another random variable. By normalizing the MI, we can compare the values for different clusterings more intuitively since a higher value will always represent a better clustering performance than a lower value. There are many ways to normalize the MI: by joint entropy [48], minimum entropy [49] or average (geometric) entropy [45]. In this thesis, the NMI is based on the average entropy and is calculated as follows:

$$NMI\left(Y,C\right) = \frac{I\left(Y;C\right)}{\text{mean}\left(H\left(Y\right),H\left(C\right)\right)}, \tag{3.1}$$

where $Y = \{Y_1, Y_2, \dots, Y_R\}$ denotes the predicted clustering, $C = \{C_1, C_2, \dots, C_S\}$ denotes the true clustering, I represents the mutual information and H is the entropy function. The mutual information I is computed as follows:

$$I\left(Y;C\right) = \sum_{i}^{R}\sum_{j}^{S} r_{ij}\log\left(\frac{r_{ij}}{p_i q_i}\right), \tag{3.2}$$

where $r_{ij} = \frac{|Y_i \cap C_j|}{n}$, $p_i = \frac{|Y_i|}{n}$, and $q_j = \frac{|C_j|}{n}$.

3.2.2 Adjusted rand index score

The adjusted Rand index (ARI) score is a measure that computes the clustering performance based on agreements and/or disagreements about the cluster assignment of each pair of data points. The ARI measure is also adjusted for chance whereas the NMI is not. The ARI is calculated by:

$$ARI\left(Y,C\right) = \frac{RI\left(Y,C\right) - E\big[RI\left(Y,C\right)\big]}{\max\left(RI\left(Y,C\right)\right) - E\big[RI\left(Y,C\right)\big]}, \tag{3.3}$$

where RI is the Rand Index score. Using the contingency matrix in Table 3.1 where n_{ij} is the number of elements in both cluster C_i and Y_j, the ARI measure can be written as follows:

$$ARI\left(Y,C\right) = \frac{\sum_{i=1}^{R}\sum_{j=1}^{S}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{R}\binom{a_i}{2}\sum_{j=1}^{S}\binom{b_j}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{R}\binom{a_i}{2} + \sum_{j=1}^{S}\binom{b_j}{2}\right] - \left[\sum_{i=1}^{R}\binom{a_i}{2}\sum_{j=1}^{S}\binom{b_j}{2}\right]\Big/\binom{n}{2}}. \tag{3.4}$$

3.2.3 Variation of information

To give an idea how the clusterings differ over the inflation range, the variation of information (VI) is calculated between the multi-scale clusterings [47]. For this paper's purposes, I am using the VI measure based on the local neighborhood which gives the distances at which the nearest neighbours of a clustering X lie (see [47], property 5). This VI measure is dependent on the clustering X, and is computed for all pairs of clusterings in the hierarchy. Calculating the VI across the hierarchy can give insight into which clusterings are stable and therefore allow the possibility to self-detect a good number of clusters for the data sets.

The formula for the VI measure looks a lot like the formula for the mutual information (Eq. 3.2) and therefore a similar notation is used. Mind that the clustering C is also a predicted clustering and not the true one:

$$VI\,(Y, C) = -\sum_{i}^{K}\sum_{j}^{L} r_{ij}\left[\log\left(\frac{r_{ij}}{p_i}\right) + \log\left(\frac{r_{ij}}{q_j}\right)\right], \tag{3.5}$$

again where $r_{ij} = \frac{|Y_i \cap C_j|}{n}$, $p_i = \frac{|Y_i|}{n}$, and $q_j = \frac{|C_j|}{n}$.

3.3 Experimental setup

In the experiments, the clustering performances of a few different techniques are compared. The main interest is in whether the latent space representation outperforms the similarity measures directly applied to the pixel values of the images. Since it is unknown how the latent space representations from the VAE's are going to vary across the different models, I decided to only measure the similarity of the elements in the latent space representations with the cosine distance. The cosine distance is the only similarity measure of the three that has a fixed scale, namely (-1, 1). Also, since there are quite a lot of different VAE models to test, it is preferable to only test one similarity measure. Thus, the following four techniques are compared:

- VAE latent space representation with cosine distance
- Pixel values with Euclidean distance
- Pixel values with image Euclidean distance (IMED)
- Pixel values with cosine distance

All of these techniques are applied to generate a similarity matrix before graph construction with either kNN or CkNN. The comparison of the clusterings are based on the NMI and ARI scores between the predicted clusterings and the true classes. For the MMNIST data set, only one level at 10 true classes is used whereas for the Fashion-MNIST data set another level at 5 categories is used as well.

3.3.1 Training the VAE's

The VAE's are trained using the training data sets of MNIST and Fashion-MNIST, which are both comprised of 60.000 samples. After training the VAE's, the test data set, with 10.000 samples, is put through the trained encoder part of the VAE's and the latent space representations are computed and saved to use later for clustering. Even though the labels of the training data sets are not directly used for the training of the VAE's, the input images are still used to compute how well the reconstruction of the decoder is. Therefore, the training data set is not used for clustering and only for training of the VAE's.

The encoder and decoder consist of fully connected dense layers, each with the same number of hidden nodes per layer. The number of nodes in the encoder's last layer, the latent space representation layer, can vary from the number of nodes per hidden layer. All hidden layers have relu activation, whereas the output layer of the decoder has sigmoid activation. For these experiments, no batch normalization or dropout were used. The stochasticity of the batch normalization on top of the stochasticity of the distribution sampling in the latent space layer can be problematic during the training of the VAE. Meanwhile, dropout layers are mainly used for regularization which is already accomplished by using KL divergence.

To find the best combination of the number of layers, the number of hidden nodes in each layer and the number of nodes in the latent space representation, multiple VAE models were trained with varying hyper-parameter combinations. The following values were used for the hyper-parameters:

- number of nodes in latent space = [16, 32, 64, 128]
- number of layers in encoder and decoder = [2, 4, 8]
- number of hidden nodes per layer = [32, 64, 128]

In total, $4 * 3 * 3 = 36$ different VAE models were trained for both the MNIST data set as the Fashion-MNIST data set individually. The VAE's are trained for the MNIST data set and the Fashion-MNIST data set separately, meaning they both have their own VAE's. The loss function is calculated by summing the KL divergence (Eq. 2.9) and negative log likelihood (Eq. 2.10). The weights are initialized using the Xavier initialization and optimised with Adam, learning rate: 0.001 and weight decay: 0.001, for 50 epochs.

3.3.2 kNN, CkNN and MST

As explained earlier, both kNN as CkNN are overlayed on a MST to ensure there are no separate sub-graphs. This allows the random walk mechanism of Markov clustering to also reach the vertices which were previously separated. The MST algorithm is also taken into account individually during the experiments to see the difference with the other methods.

The kNN and CkNN algorithms are tested with a varying number of k neighbours: $k = \{3, 5, 7, 9\}$. For CkNN, we do not adjust the δ value (Eq. 2.18) and have fixed its value to the default value of 1 since Liu and Barahona [36] showed that the use of Markov clustering alleviates the dependency on the δ value for detecting clusters at different levels of coarseness.

3.3.3 Best model and graph construction method selection

Since it is quite computationally expensive and time costly to perform Markov clustering on a large graph with 10.000 nodes, I first determined which latent space representation model and graph construction method performed best on a smaller number of images of the test data set. The images are taken from the test data set specifically since, as mentioned earlier, the training data sets are used only for the training of the VAEs. The small subset for these initial experiments consists of the first 50 samples from each class. The selection of the graph construction methods for the similarity measures directly applied to the pixel values is also based on the same small subset of the test data set to keep consistency across the experiments.

The selections are based on the highest NMI value where the number of detected clusters is either 9, 10, or 11, where 10 clusters is the true number of clusters in both data sets. I have chosen to also include the values surrounding 10 because the clustering framework does not always find exactly 10 clusters in any of the clusterings with the given step size for the inflation range (more on this later). Nonetheless, I have done more experiments with values in between to try and find a value for exactly 10 clusters, but this was not always successful. Luckily, the NMI measure can also be used to compare two clusterings with a differing number of clusters, and therefore a reasonable estimate of the best model and graph construction method can be made.

3.3.4 Experiments on the larger test data set

Now that we have selected a VAE model for the latent space representation and a graph construction method for each of the techniques for both data sets, the techniques are compared to each other on the remainder of the test data sets. The remainder of the test data sets is comprised of 9500 samples.

The goal is to find clusterings at different hierarchical levels, i.e. where the number of detected clusters varies across clusterings. To this end, multiple parameters in the Markov clustering framework are adjusted such that the number of clusters in the clusterings across the hierarchy cover a particular range. For both data sets, this range is set with a minimum of 5 clusters and a maximum of 10 clusters. In the next section, I will mention exactly how this range is set and how the Markov clustering parameters affect the number of clusters detected within a single clustering.

The NMI and ARI measures are used to determine which technique, namely the Euclidean distance, IMED, the cosine distance or the VAE latent space representation, performs best in combination with the Markov clustering framework on the MNIST and Fashion-MNIST data set. For the best performing technique, the multi-scale clusterings are evaluated in terms of the variation of information (VI) measure. Additionally, the cluster assignments of the data points are compared at the different levels of the hierarchy to see whether the clusterings are semantically consistent.

3.3.5 Overview

All in all, there are quite a few steps to get to the final experiments. A short overview is shown below.

1. Train different VAE's for latent space representation on training data sets.
2. Select a VAE model and graph construction method combination on the small subset of 500 samples from the test data sets.
3. Select a graph construction method for the similarity measures directly applied to the pixel values on the small subset of 500 samples from the test data sets.
4. Apply all selected models/graph construction methods with Markov clustering on the remainder of the test data set, which is 9500 samples, to find multi-scale clusterings.
5. Evaluate the clusterings of the best performing technique.

3.4 Markov stability framework

A Python implementation of Van Dongen [43] is used[1].

3.4.1 Influence of the parameters

The Markov clustering framework [43] has a couple of parameters, each with their own influence on the resulting clustering.

- Expansion: As mentioned previously, the expansion parameter allows flow through the network. A higher expansion value, i.e. more matrix multiplications in one step, represents multiple random walks and thus allows the flow to go further than one node. Therefore, the flow reaches more nodes in the graph and allows the ability to merge smaller clusters, which results in a lower number of detected clusters in the final clustering.
- Inflation: The inflation parameter does the exact opposite of the expansion paramter by directing flow to nodes with the higher transition probabilities. It contracts the flow of information which results in the flow reaching a fewer number of nodes. The higher the inflation value, the fewer nodes are reached and therefore the number of detected clusters at convergence is larger. Also, a higher inflation value helps reach convergence in less iterations, where a single iteration consists of a number of expansion steps in combination with a single inflation step.
- Pruning threshold: The pruning threshold helps to convergence even faster by setting the transition probabilities to zero when the value is lower than the threshold. Increasing the threshold value results in even more transition probabilities that are removed, and thus even faster convergence. However, by removing transition probabilities, less nodes are able to receive the flow from the expansion step which results in a higher number of clusters detected in the resulting clustering.

3.4.2 Multi-scale detection of clusterings

Since a higher inflation value allows to find clusterings with a higher number of clusters, a range of inflation values is used in each experiment to detect clusterings at different levels of coarseness. The expansion and inflation parameters are adjusted such that in each experiment the same range of detected clusters is found.

For the initial experiments on the smaller subset of the data, to get the best performing VAE model and graph construction method, the default expansion value of 2 in combination with a range of inflation values was enough to find clusterings where the number of clusters equals 9, 10, or 11 for all models and graph construction methods. The inflation range from 1.15 to 1.45 with a step size of 0.01 was used in these experiments. The selection of the VAE models and graph construction methods is based on the highest NMI value over all results where the predicted clustering has exactly 9, 10, or 11 clusters.

Unfortunately, the standard expansion value of 2 is not high enough for the Markov clustering framework to find a minimum of 5 clusters on the larger data sets. Since the graph has more nodes, we need to increase the flow of information to reach those nodes. The expansion value is highly increased to 10, 15 or 20 to achieve detection of a minimum of 5 clusters in the clustering with a low inflation value, which is also determined for each experiment individually. The inflation value is increased with step sizes of 0.01

[1]Implementation available at https://github.com/guyallard/markov_clustering and details about the algorithm available at https://micans.org/mcl/

Data set	Method	Best graph	Expansion	Inflation range (step = 0.01)
MNIST	VAE (2, 64, 16)	CkNN9	15	1.30 - 2.50
	COSINE	CkNN7	15	1.30 - 1.80
	EUCL	CkNN9	15	1.30 - 1.95
	IMED	CkNN9	20	1.20 - 1.85
Fashion-MNIST	VAE (2, 32, 16)	kNN5	10	1.30 - 1.80
	COSINE	CkNN9	20	1.20 - 1.70
	EUCL	CkNN5	15	1.30 - 1.70
	IMED	CkNN5	15	1.10 - 1.70

Table 3.2: Expansion value and inflation range for the Markov clustering framework on the larger graphs (N=9500). The VAE model's notation represents the number of layers, the number of hidden nodes per layer and the number of nodes in latent space respectively.

until the resulting clustering reaches 10 clusters or more. In Table 3.2, the expansion value and inflation ranges are shown for each method applied to the larger data set. The selected VAE models and graph construction methods are also mentioned in the table for completeness. These results will also be more elaborated on in the remainder of this thesis.

Chapter 4

Results

4.1 Selection of VAE model and graph construction method

In this section, the results on the smaller subset to select a VAE model and a graph construction method for each technique are outlined. The selected VAE model and graph construction methods are then used to perform multi-scale Markov clustering on the larger test data set.

MNIST

Graph	**CkNN**				**kNN**				**MST**	**mean**
k	**3**	**5**	**7**	**9**	**3**	**5**	**7**	**9**		
COSINE	0.6152	0.6316	**0.6907**	0.6760	0.5880	0.6411	0.6213	0.6652	0.6023	0.6368
EUCL	0.5261	0.5347	0.6260	**0.6407**	0.5566	0.6109	0.5982	0.5871	0.5315	0.5791
IMED	0.4693	0.6147	0.6518	**0.6916**	0.6239	0.6377	0.6451	0.6452	0.5935	0.6192

Table 4.1: MNIST: maximum NMI values at the detection of 9,10 or 11 clusters of the experiments on 500 samples to select best graph construction method for similarity measures applied directly to the pixel values.

Fashion-MNIST

Graph	**CkNN**				**kNN**				**MST**	**mean**
k	**3**	**5**	**7**	**9**	**3**	**5**	**7**	**9**		
COSINE	0.6357	0.6702	0.6567	**0.6725**	0.6650	0.6671	0.6709	0.6653	0.5628	0.6518
EUCL	0.5935	**0.6572**	0.6203	0.6533	0.5957	0.6135	0.6153	0.5980	0.5695	0.6129
IMED	0.5946	**0.6341**	0.6227	0.6276	0.5540	0.6112	0.6237	0.6219	0.5365	0.6029

Table 4.2: Fashion-MNIST: maximum NMI values at the detection of 9,10 or 11 clusters of the experiments on 500 samples to select best graph construction method for similarity measures applied directly to the pixel values.

4.1.1 Similarity directly applied on the image data

In Table 4.1 and Table 4.2 above, you can find the NMI values for the experiments where the similarity measure is directly applied to the MNIST data set and the Fashion-MNIST data set respectively. For the MNIST data set, the cosine distance performed best in combination with the CkNN graph construction method and 7 neighbours

(NMI=0.6907). The Euclidean distance and IMED both performed best with CkNN where the number of neighbours is 9 (NMI=0.6407 and NMI=0.6916 respectively).

On the Fashion-MNIST data set, the Euclidean distance and IMED both perform best with CkNN and 5 neighbours (NMI=0.6572 and NMI=0.6341 respectively). Note that the difference for the Euclidean distance between CkNN5 and CkNN9 is very small (diff=0.0039). The cosine distance performs best with CkNN and 9 neighbours on the Fashion-MNIST data set (NMI=0.6725), again with a very small difference between CkNN5 and CkNN9 (diff=0.0023).

4.1.2 VAE latent space representation

In Table 4.3 and Table 4.4, the top 5 results are shown for the experiments to select a VAE model and a graph construction method for the latent space representation. The entire result tables can be found in Appendix section 7.1. According to these experiments, the VAE model with 2 hidden layers, 32 nodes per hidden layer, and 16 nodes in the latent space layer performs best on the 500 samples from the MNIST data set. The accompanying best graph construction method is CkNN with 9 neighbours.

For the Fashion-MNIST data set, the VAE model with 2 hidden layers, 32 nodes per hidden layer and 16 nodes in the latent space layer performs best in combination with kNN and 5 neighbours.

As mentioned previously, it is quite difficult to find the exact right inflation value for the detection of exactly 10 clusters. This was especially the case for the VAE model and graph construction selection process. Therefore, some entries have missing values even though the clusterings with 9 and 11 clusters are included.

MNIST

G	**CkNN**				**kNN**				**MST**	**mean**
k	**3**	**5**	**7**	**9**	**3**	**5**	**7**	**9**		
(2, 32, 128)	0.6004	0.6366	0.6475	**0.6980**	0.6311	0.6680	0.6917	0.6436		0.6521
(2, 64, 16)	0.5730	0.6497	0.7002	**0.7373**	0.6579	0.6567	0.7350	0.7165	0.5312	0.6619
(2, 64, 32)		0.6343	0.6985	0.6860	0.6305	0.6154	0.6606	**0.7037**	0.5314	0.6451
(2, 64, 128)	0.5524	0.6741	0.7255	**0.7341**	0.6550	0.6756	0.7296	0.7261	0.5940	**0.6741**
(2, 128, 16)	0.6031	0.6145	0.6256	**0.6732**	0.6133	0.6477	0.6532	0.6294	0.5785	0.6265
mean	0.5822	0.6418	0.6795	**0.7057**	0.6376	0.6527	0.6940	0.6839	0.5588	0.6519

Table 4.3: MNIST: maximum NMI values at the detection of 9,10 or 11 clusters for the experiments on 500 samples to select a VAE model and a graph construction method for latent space representation. Notation of the model in the left column: (number of hidden layers, number of nodes per hidden layer, number of nodes in latent space layer).

Fashion-MNIST

G	**CkNN**				**kNN**				**MST**	**mean**
k	**3**	**5**	**7**	**9**	**3**	**5**	**7**	**9**		
(2, 32, 16)	0.5534	0.5780	0.5919	0.6315	0.5844	**0.6482**	0.6459	0.6292	0.5774	**0.6044**
(2, 32, 64)	0.5684	0.5984	0.6019	**0.6358**	0.6149	0.6353	0.6289	0.6325	0.4892	0.6006
(2, 32, 128)	0.5302	0.5194	0.6043	**0.6408**	0.5673	0.5789	0.6133	0.6266	0.5460	0.5808
(2, 128, 32)	0.4632	0.5677	**0.6252**	0.6046	0.5241	0.6010	0.5959	0.6103	0.5187	0.5679
(8, 128, 16)	0.5413	0.5898	**0.6284**	0.6148	0.5608	0.5665	0.4945	0.5811	0.5218	0.5665
mean	0.5313	0.5707	0.6103	**0.6255**	0.5703	0.6060	0.5957	0.6159	0.5306	0.5840

Table 4.4: Fashion-MNIST: maximum NMI values at the detection of 9,10 or 11 clusters for the experiments on 500 samples to select a VAE model and a graph construction method for latent space representation. Notation of the model in the left column: (number of hidden layers, number of nodes per hidden layer, number of nodes in latent space layer).

4.2 Markov clustering on larger test data sets

The VAE models and graph construction methods with the highest NMI in the previous section are used to perform the Markov clustering on the larger test data sets with 9500 samples. To reiterate, the following VAE models and graph construction methods were used for each technique on each data set:

- MNIST:
  - VAE + CkNN9,
    where the VAE model has 2 hidden layers, 64 hidden nodes per layer and 16 nodes in the latent space layer.
  - COSINE + CkNN7
  - EUCL + CkNN9
  - IMED + CkNN9
- Fashion-MNIST:
  - VAE + kNN5,
    where the VAE model has 2 hidden layers, 32 hidden nodes per layer and 16 nodes in the latent space layer.
  - COSINE + CkNN9
  - EUCL + CkNN5
  - IMED + CkNN5

4.2.1 Best performance at *true* number of detected clusters

The NMI and ARI scores for the final clustering experiments on the large test data sets of 9500 samples are displayed in Table 4.5. Multiple inflation values can result in the same number of clusters in the resulting clustering. Therefore, the values in Table 4.5 represent the highest values found over the multiple inflation values. The NMI and ARI values are based on the 10 true classes in both data sets. For the Fashion-MNIST data set, the maximum NMI and ARI scores at 5 clusters in a clustering are also computed with the 5 higher level categories as mentioned in section 3.1.

	MNIST@10		**Fashion-MNIST@10**		**Fashion-MNIST@5**	
	NMI	**ARI**	**NMI**	**ARI**	**NMI**	**ARI**
VAE	**0.7970**	**0.7403**	0.5318	0.3426	0.5678	0.4276
COSINE	0.6558	0.3933	**0.6301**	**0.4738**	**0.6876**	**0.5104**
EUCL	0.6327	0.3250	0.5468	0.3150	0.5759	0.3953
IMED	0.6573	0.3364	0.5681	0.3426	0.5967	0.3628

Table 4.5: NMI and ARI scores of resulting clusterings on the test data set with 9500 samples. The scores displayed are computed for the true clustering at 10 clusters for both data sets and also at 5 clusters for the Fashion-MNIST data set.

For the MNIST data set, the VAE latent space representation gives the highest NMI (=0.7931) and ARI (=0.7345) score at 10 clusters. These values are both remarkably higher than the values for the other techniques on the MNIST data set. The cosine similarity directly applied to the pixel values on the Fashion-MNIST data set performs best in terms of both the NMI (=0.6301) and ARI (=0.4738) scores in comparison to the other techniques. The same holds for the performance on Fashion-MNIST with the 5 higher level categories, where both the NMI (=0.6876) and ARI (=0.5104) are highest for the cosine distance directly applied to the pixel values.

	MNIST		Fashion-MNIST	
	NMI	**ARI**	**NMI**	**ARI**
VAE	**0.8025 @9**	**0.7826 @12**	0.5901 @6	0.4268 @6
COSINE	0.7022 @11	0.4697 @11	**0.6512 @8**	**0.5193 @8**
EUCL	0.7433 @12	0.6052 @13	0.5699 @9	0.3644 @9
IMED	0.7271 @6	0.4916 @6	0.6232 @7	0.4500 @7

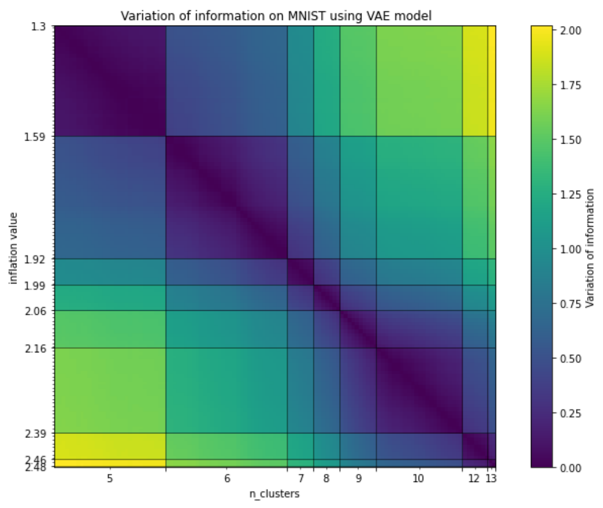
Table 4.6: Highest NMI and ARI scores for each technique over all resulting clusterings on the large test data set of 9500 samples, where @c stands for the number of detected clusters in the clustering.

4.2.2 Best performance at *variable* number of detected clusters

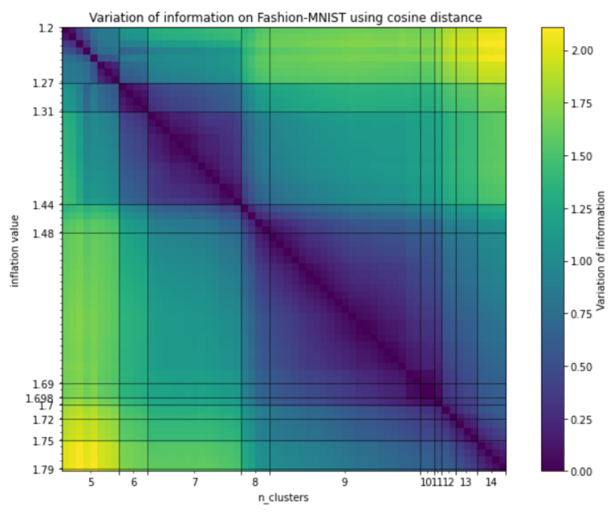
The highest NMI and ARI values regardless of the number of clusters in the resulting clustering are shown in Table 4.6. These values are a little bit higher than their corresponding values at the detection of the *true* number of classes. However, the same techniques, namely the VAE latent space representation for the MNIST data set and the cosine distance for the Fashion-MNIST data set, report the highest overall NMI and ARI values in both instances.

4.2.3 Variation of information of the clusterings

The variation of information (VI) is computed for the best performing techniques on the larger test data set as determined in the previous section. For the MNIST data set, the clusterings from the VAE latent space representation in combination with CkNN9 are used and for the Fashion-MNIST data set, the clusterings from the cosine distance with CkNN9 are used. For both data sets, the VI is calculated for all pairs of clusterings across the inflation value range. In Figure 4.1, the VI heatmaps for the MNIST data set and Fashion-MNIST data set are displayed. To provide extra information, the x-axis of the figures are renamed to the number of clusters in the clusterings. The accompanying inflation value ranges are shown on the y-axis.



(a) MNIST



(b) Fashion-MNIST

Figure 4.1: Variation of information across different levels of hierarchy calculated for a) the clusterings resulting from the MNIST data set with VAE latent space representation and CkNN9 and b) the clusterings resulting from the Fashion-MNIST data set with the cosine distance and CkNN9.

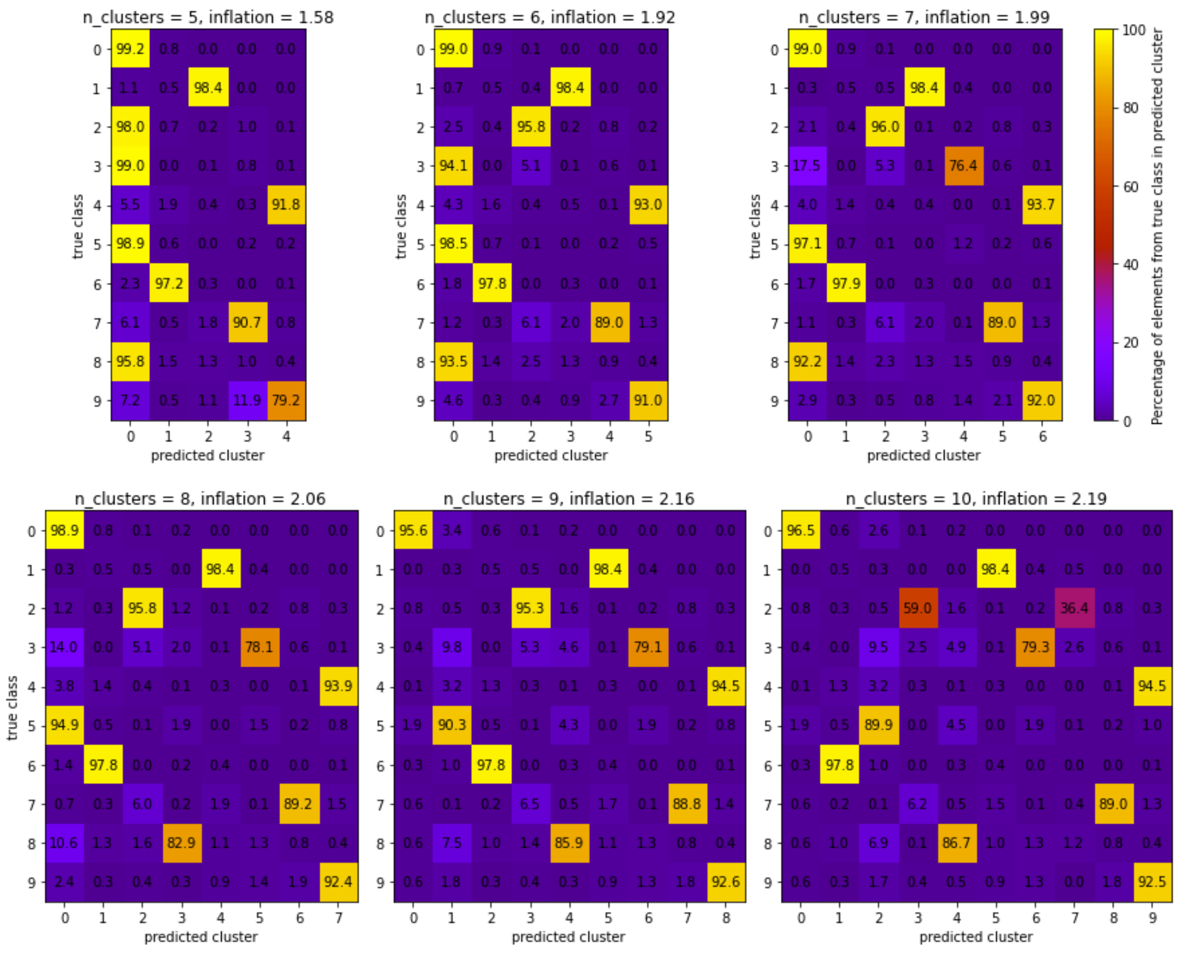


Figure 4.2: MNIST: cluster assignment of the data points in the best clusterings at different levels of the hierarchy. The values in the heatmap represents the percentage of data points from the true classes that are assigned to the predicted cluster.

The VI heatmap for the MNIST data set shows large plateaus of low VI values at 5, 6, and 10 detected clusters. For the Fashion-MNIST data set, the larger plateaus of low VI values can be found at 7 and 9 detected clusters. In the discussion, section 5.1.3, the meaning of these plateaus is discussed further.

4.2.4 Cluster assignments of the data points at different levels in the hierarchy

The quality of the multi-scale clusterings can be checked by evaluating the cluster assignments of the data points. In Figure 4.2, the cluster assignments for the MNIST data set are displayed. In Figure 4.3, the cluster assignments for the Fashion-MNIST data set can be found. In the figures, the cluster assignments with the best NMI per number of clusters is shown. The heatmaps show the percentage of elements from the true class that is clustered within a certain cluster.

At 5 detected clusters, the MNIST data set is able to differentiate class ‘1’, ‘6’ and ‘7’ from the rest with at least 90% of the corresponding class elements. Class ‘7’ is also clustered with 11.9% of elements from class ‘9’. The classes with digits ‘4’ and ‘9’

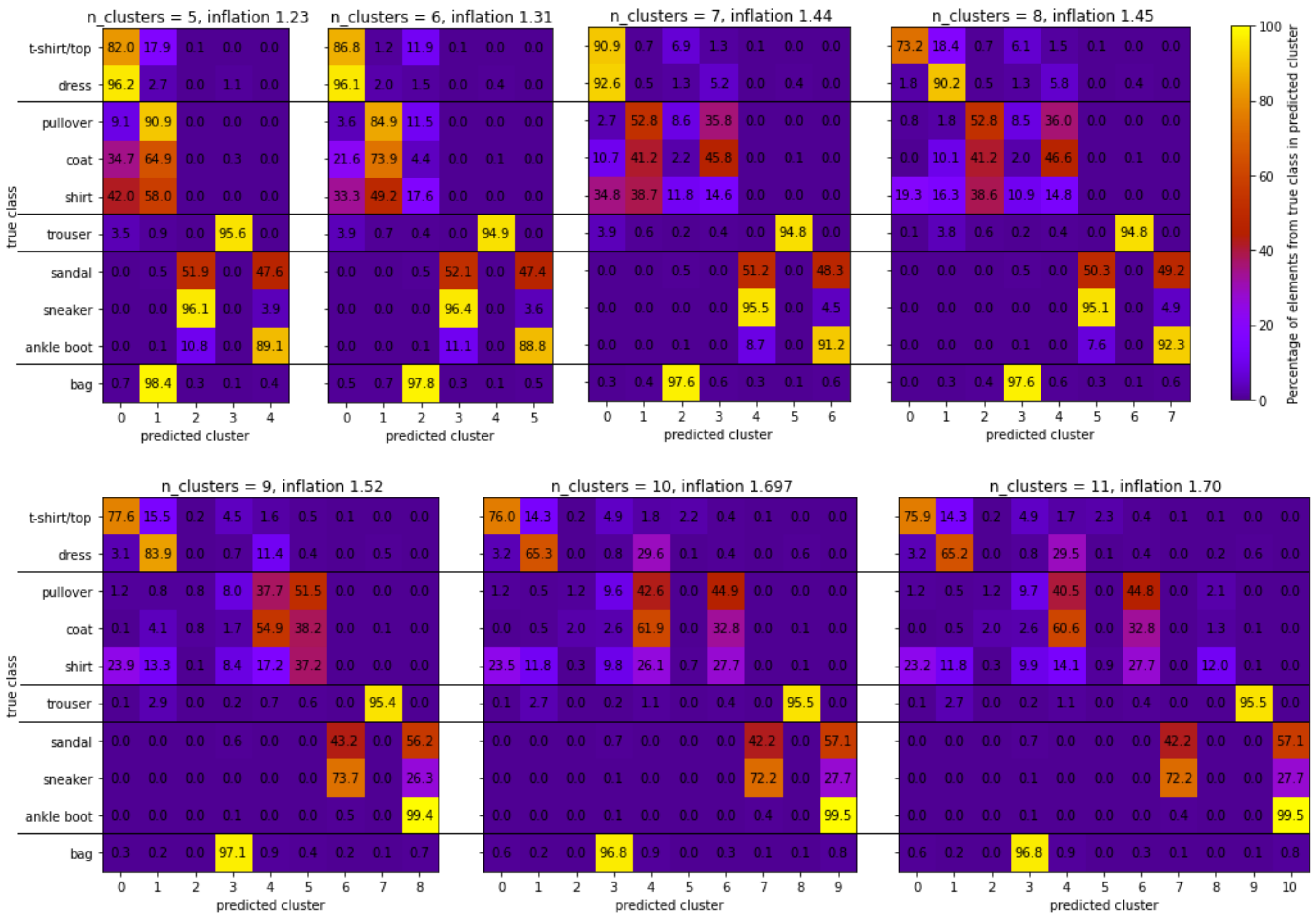


Figure 4.3: Fashion-MNIST: cluster assignment of the data points in the best clusterings at different levels of the hierarchy. The values in the heatmap represents the percentage of data points from the true classes that are assigned to the predicted cluster. The horizontal lines help distinguish the 5 higher level category membership of the 10 true classes. The order from top to bottom is 1) short-sleeved, 2) long-sleeved, 3) bottoms, 4) shoes, and 5) accessories.

are clustered together, as well as the remaining classes '0', '2', '3', '5', and '8'. At 10 detected clusters, each class roughly has its own cluster, except the classes '4' and '9' which are clustered together. Additionally, the class of digit 'two' is divided over two clusters.

For the Fashion-MNIST data set, only the class 'trouser' is clustered separately with 95.6% of the elements at 5 detected clusters. Cluster 2 and cluster 4 both do have mainly 'shoes' items which is consistent with the 5 higher level classes for the Fashion-MNIST data set. At 10 clusters, again only the class 'trouser' is distinguished from the rest, with 95.5% of the elements clustered together. The elements from the category 'shoes' are divided over cluster 7 and cluster 9, but contain hardly any elements of the other classes. Note that almost all elements (96.8%) of the class 'bag' is clustered together, but the cluster also contains some elements from the other classes. All the other detected clusters consist of elements from different true classes.

Chapter 5

Discussion

In the first part of the discussion, I would like to come back to the research questions from the Introduction. Next, some additional effects of the hyper-parameters of the graph construction methods are discussed in relation to the resulting clusterings. Finally, the limitations of this research and possible future directions are mentioned.

5.1 Revisiting the research questions

5.1.1 Latent space representation vs. directly from pixel values

Based on the performance of each technique on the larger MNIST test data set, the latent space representation of the VAE gives the highest NMI and ARI scores at exactly 10 detected clusters. Also the NMI and ARI scores are highest for this technique when the clusterings are compared with a variable number of detected clusters. The difference in NMI and especially ARI scores are large enough to conclude that in the case of the MNIST data set, the latent space representation is best suited. The selected VAE model had only 2 hidden layers with each 64 hidden nodes and the latent space representation only had 16 nodes. The input images are greatly reduced from 784 pixels to 16 features and still provide good clustering results. This illustrates the usability of VAEs to reduce dimensionality while preserving, and maybe even improving, the similarity between the input images in the latent space representation.

The results on the Fashion-MNIST data set tell a different story. Here, the cosine distance has the highest NMI and ARI, again both at exactly 10 detected clusters and at a variable number of detected clusters. The cosine distance also has the highest NMI and ARI scores for the higher level categories at 5 clusters. The VAE latent space representation performs about the same as the Euclidean distance and IMED.

Unfortunately, the use of a similarity measure specifically designed for image data, namely IMED, does not significantly improve the clustering performance. For both data sets, IMED performs approximately the same as the regular Euclidean distance, with maybe slight improvements every now and then. These improvements are negligible when taking the computing time of the IMED measure into account. The IMED measure is computed for all pairs of samples in the data set, with the coefficient matrix being of size $MN\text{x}MN$ for images of size $M\text{x}N$. Especially for large data sets, the computation time increases significantly. There is a way to reduce the computation time by first applying the Standardizing transform (ST) on the images themselves and then calculate the Euclidean distance for all image pairs [14]. In this thesis, the ST was not applied on the images due to a lack of availability of the code implementation.

5.1.2 Graph construction

Although the graph construction methods have not been thoroughly tested on the larger test data sets, there still can be said something about the results of the experiments on the small subsets. For the MNIST data set, all selected graph construction methods were of the CkNN kind. In addition, a higher number of 7 and 9 neighbours was preferred. For the Fashion-MNIST data set, the results varied a bit more and kNN was also amidst the selected graph construction methods. The majority of the selected methods had 5 neighbours and only one had a preference for 9 neighbours.

The results on the experiments for the VAE model selections can also tell use more about which graph construction method is most robust across the VAE models with differing hyper-parameters. In the Appendix section 7.1, Table 7.1 and Table 7.2, the NMI scores are reported for each combination of VAE parameters and graph construction method. The mean values are also shown across each VAE model and across each graph construction method. Out of all the graph construction methods, the CkNN graph with 9 neighbours shows the highest average NMI value across the different VAE models for both data sets.

As previously mentioned in section 3.3.2, the parameters of the graph construction methods are somewhat alleviated by the use of Markov clustering. Therefore, the best graph construction method suited for Markov clustering is quite ambiguous.

5.1.3 Relation between clusterings across the hierarchy

The multi-scale detection of clusters allows us to investigate the category membership of the data at different levels of abstractness. The clusterings in the hierarchy can help find similarity relationships between the true classes as well. Another reason for multi-scale detection of clusters is the ability to self-detect the best number of clusters for the data set. To this end, the variation of information (VI) is computed for all clusterings across the hierarchy. As seen in Figure 4.1, the plateauing of the VI measure for a particular range of inflation values indicates a robust clustering since the inflation value needs to be raised more in order to make a difference in the resulting clustering. If the number of true classes in the data is unknown, the plateauing of the VI measure at a certain number of detected clusters provides an estimate of the true number of classes. Especially for data with hierarchical classes, the VI measure has the ability to detect the different levels of category membership. The cluster assignments in Figure 4.2 and Figure 4.3 are used to evaluate these particular clusterings.

MNIST

The VI results for the MNIST data set show relatively large plateaus at 5, 6, and 10 detected clusters. Thus, the results imply a detected hierarchy of 3 different levels of abstractness. Since there is no real hierarchy within the MNIST data set, these results are difficult to interpret. Instead, we will look more into the relationships between the elements of the true classes that are clustered together at these levels.

Firstly, at 5 detected clusters, the classes '1' and '6' each have a large percentage of their elements in their own cluster (98,4% and 97,4% respectively) with only a small percentage (<2.0%) of elements from the other classes. Consequently, digit '1' and digit '6' are most distinguishable from the other digits. The majority of the elements of class '7' (90.7%) is clustered together, but also with a fair percentage of the elements from class '9'. The elements from class '4' and class '9' are mostly clustered together. The digits '4' and '9' have a similar construction as both mainly consist of a circle with a vertical line on the right side, where the circle is more angular for the '4' and the vertical line rounds of to the left for the '9'. Therefore, it is expected that the elements are clustered together at a higher level in the hierarchy. For the remaining classes, ‘0’, ‘2’, ‘3’, ‘5’, and ‘8’, a large majority of the elements (>95%) from each class is clustered together in the same cluster but not in their own cluster. Therefore, we can conclude that it is more difficult to separate these classes from each other than the classes that do have their own predicted cluster.

Secondly, at 6 detected clusters, the cluster assignments are fairly similar to the cluster assignments at 5 detected clusters. One main difference is that now the majority of class ‘2’ (95.8%) has its own cluster with some elements from class ‘3’ and class ‘7’. Additionally, more elements from class ‘9’ (now 91.0%) are clustered together in the cluster also containing more elements from class ‘4’ (now 93.0%). So, with more clusters detected, the digit ‘2’ is separated from the large mixed cluster, implying the elements from class ‘2’ are more distinguishable than the elements from class ‘0’, ‘3’, ‘5’ and ‘8’.

Lastly, at 10 detected clusters, most digits have their own cluster. The clusters containing the majority of elements for class ‘0’, ‘1’, ‘3’, ‘6’ and ‘7’ only contain a small percentage (<2.0%) of elements from other classes. Again, the majority of the elements from digits ‘4’ and ‘9’ are clustered together (94.5% and 92.5% respectively), again with each a higher percentage than at the previous hierarchy level. The elements of class ‘2’ are divided over two clusters, one also containing a few elements from class ‘7’. The cluster with the majority of class ‘5’ (89.9%) also has some elements of class ‘3’ and class ‘8’. In the cluster with the majority of class ‘8’ (86.7%), a few elements from class ‘3’ and class ‘5’ are found. In these two latter clusters, mostly elements from the same three classes (‘3’, ‘5’ and ‘8’) are found, indicating a similarity relationship between them.

In summary, the elements in the classes ‘0’, ‘1’, ‘3’, ‘6’ and ‘7’ are each most distinguishable from the elements of the other classes. The elements from the classes ‘4’ and ‘9’ have the highest similarity relationship since most elements from these classes are always clustered together at each level in the hierarchy. There is a similarity correlation between elements from class ‘3’, ‘5’ and ‘8’, as shown by the cluster assignments at 10 detected clusters. The elements of class ‘2’ have the highest inter-class dissimilarity as they are divided over two clusters at the lower level in the hierarchy.

Fashion-MNIST

On the Fashion-MNIST data set, the VI results show two interesting plateaus. The first plateau at 7 detected clusters is a relatively smaller than the second plateau at 9 detected clusters. Unfortunately, the VI results showed no real plateau at 5 detected clusters for the evaluation of the higher level categories as mentioned in section 3.1. Nonetheless, I will still compare the cluster assignments of the best clustering at 5 detected clusters to validate the higher level categories. These higher level categories are also used as a guideline for the clusterings at the two detected hierarchies of 7 and 9 clusters. In Figure 4.3, as a visual aid, the division of classes by the horizontal lines represent the higher level category memberships.

At 5 detected clusters, only the class ‘trouser’ has its own cluster with 95.6% of its elements clustered together and hardly any elements from the other classes. Almost all elements from the classes ‘sandal’, ‘sneaker’ and ‘ankle boot’, all part of the higher level category ‘shoes’, are clustered together over two clusters. Interestingly, the elements from the classes ‘sneaker’ and ‘ankle boot’ are more distinguished from each other than from the elements of the class ‘sandal’. Since sneakers and ankle boots are both closed shoe types, the expectation was that these two would be more similar to each other than to the open sandals. Alternatively, the division of the elements from the ‘sandal’ class over the two clusters might have something to do with the height of the sandals, as sneakers are generally lower than the ankle, whereas ankle boots are a little higher. The majority of the elements from the class ‘dress’ (96.2%) are clustered together, but also with a large percentage of elements from the classes ‘shirt/top’, ‘coat’, ‘shirt’, and a bit smaller percentage of elements from the class ‘pullover’. The joined clustering of the majority of elements from the classes ‘dress’ and ‘t-shirt/top’ is in accordance to the higher level category membership. For the class ‘bag’, 98.4% of the elements are clustered together, but not on its own. The cluster also contains the majority of the classes ‘pullover’ (90.9%), ‘coat’ (64.9%) and ‘shirt’ (58.0%).

Next in the hierarchy, at 7 detected clusters, the division of the elements over the clusters seems to adhere more to the higher level categories. The results for the ‘shoes’ category are almost identical to the clustering at 5 detected clusters. The same goes for the class ‘trouser’. On the other hand, more elements of the class ‘t-shirt/top’ (90.9%) are clustered together with elements of the class ‘dress’ (92.6%), with less elements from the classes ‘coat’ and ‘shirt’. The elements for the ‘long-sleeved’ category are mostly clustered together, but similar to the ‘shoes’ category, are divided over two clusters. The most noteworthy change for this clustering is that the elements from the class ‘bag’ are better separated from the elements of the ‘long-sleeved’ category with smaller percentages of the elements from the classes ‘pullover’, ‘shirt’ and ‘t-shirt/top’.

Lastly, at 9 detected clusters, there is quite a lot of variability in terms of the elements from the true classes that are assigned to one clustering. Still, the elements from the class ‘trouser’ are very well distinguished from the elements of the other classes. The majority of the elements from the classes ‘t-shirt/top’ (77.6%) and ‘dress’ (83.9%) are no longer part of the same cluster, and especially the class ‘dress’ shares a cluster with a fair portion of the elements from the class ‘t-shirt/top’. Both also share a cluster with a decent percentage of elements from the class ‘shirt’. The elements for the class ‘bag’ are slightly better separated from the other classes in the previous clustering at 7 detected clusters. The two clusters covering the ‘long-sleeved’ category are still approximately the same and quite ambiguous. Some elements of the class ‘dress’ are also included. For the ‘shoes’ clusters, almost all elements from the class ‘ankle boot’ are assigned to the same cluster. Note that one cluster, number 2, has a very small number of elements and contains no majority of elements from any of the classes. The cause for this almost empty cluster is unknown as it was not a part of the experiments to investigate which exact elements are in the clusters. One reason might be that the elements are highly specific cases, called outliers, where even the most similar image is dissimilar. This can result in odd connections between the image nodes in the graph and give the Markov framework a difficult task of assigning them to clusters.

In summary, the cluster assignments of the data points are quite well related to the higher level category memberships. Especially the class ‘trousers’ is well distinguished from the other classes at all levels in the hierarchy. The used technique in combination with Markov clustering does not perform well enough to separate the classes involving upper-body items, although some division can be found in terms of the ‘short-sleeved’ and ‘long-sleeved’ classes. The cluster involving the majority of elements from the class ‘bag’ also has some elements from the other clusters, but these percentages were smaller at 7 and 9 detected clusters than at 5 detected clusters.

5.2 Effect of k in CkNN and kNN on the number of detected clusters

The inflation value to detect a specific number of clusters in the clustering is highly dependent on the number of neighbours chosen for the graph construction methods. A node in a graph constructed with a lower number of neighbours has a lower number of possible transitions. In the overall graph, there are less connections, for which the transition probabilities are reduced faster with the inflation parameter and pruning threshold. Consequently, a low inflation value on a graph with a smaller number of neighbours per node results in a higher number of detected clusters than on a graph with a higher number of neighbours per node. In other words, there is a negative correlation between the number of neighbours in a graph and the number of detected clusters in the resulting clustering.

5.3 Computational complexity of the Markov clustering framework

The Markov clustering is the most complex part of the experiments. Assuming N is the number of nodes in the graph, the inflation step, regardless of the value, can be done in $\mathcal{O}\left(N^2\right)$. One inflation step combined with one expansion step (NxN) is of $\mathcal{O}\left(N^3\right)$. However, with more matrix multiplications per expansion step, i.e. with a higher expansion value, the complexity increases with every additional matrix multiplication. Assuming exp is the expansion value (which is 2 by default), the complexity of the Markov clustering framework becomes of $\mathcal{O}\left(N^{1+exp}\right)$. Van Dongen [43] does provide the option of using sparse transition matrices for clustering, meaning the default algorithm is of $\mathcal{O}\left(Nk^2\right)$, where k is the average number of neighbours per node in the graph.

5.4 Limitations

The current research has a couple of limitations. Firstly, the VAE model and graph construction method selections were done on a small subset of 500 samples of the test data sets. These 500 samples consisted of the first 50 samples from each class in the test data sets. In hindsight it would have been better to randomly sample these 50 samples from each class and base the final selection on the best average performance over multiple repetitions. However, due to time restrictions this was no longer possible. The consequence is that the selected VAE models and graph construction methods are based entirely on one fixed sampling and could cause a bias in the final results. For MNIST specifically, the authors [44] have stated that the first 5000 samples in the test data set are less difficult than the latter 5000 samples. The bias relating the used graph construction methods should be limited as the sensitivity for the graph construction parameters are alleviated by the use of Markov clustering [36].

Another limitation is that only one application, namely the use of VAE models, were tested to reduce the dimensionality of the input images. Additionally, other (dis)similarity measures or image descriptors were not investigated in this thesis. There is no comparison to current state-of-the-art methods for image clustering or to standard clustering methods like K-means. The main focus in this thesis is on the use of the VAE latent space representation and their ability to still find good and meaningful multi-scale clusterings.

The range of number of clusters at which the multi-scale clusterings were detected is restricted from 5 to 10 clusters. The clusterings with a number of detected clusters outside of this range are not taken into account even though these clusters could also have helped to investigate the relationships of the clusterings across the hierarchy.

5.5 Future directions

For the future directions, one goal would be to negate the first limitation mentioned in the previous section. The sampling problem can easily be tackled by taking multiple random samplings of 500 nodes and averaging the resulting NMI scores. Also, with more computing power and time, all techniques and graph construction methods could be applied to the entire test data sets of MNIST and Fashion-MNIST, both comprising of 10.000 samples. In this case, there is no need to first select a method that performs best on the small subset of 500 samples. The constructed graphs can be further optimised for image clustering as well by adjusting the δ parameter of the CkNN algorithm.

Secondly, the performance of other trending dimensionality reduction methods like Generative Adversarial Networks (GANs) [41] could be investigated for multi-scale image clustering.

Lastly, the investigated techniques could be applied on other more complex image data sets, like the object data set COIL100 [50] and maybe even the hierarchical real-world data set ImageNET [51]. The results on these data sets could give a better idea of how helpful the techniques are at multi-scale clustering of real-world examples.

Chapter 6

Conclusion

In this thesis, the usability of VAE latent space representations for multi-scale image clustering was investigated on the MNIST data set and the Fashion-MNIST data set. To this end, the performances of the VAE latent space representations were compared to the performances of the Euclidean distance, image Euclidean distance (IMED) and the cosine distance directly applied to the image pixels. For each of these techniques, a graph construction method was selected beforehand on a small subset of the test data sets. For the VAE latent space representations, a VAE model was selected beforehand as well from multiple VAE models with different hyper-parameters. Multi-scale Markov clustering was performed with the selected VAE models and graph construction methods on the remainder of the test data sets.

For the selection of the graph construction methods, there was a slight preference for the continuous k-nearest neighbour (CkNN) algorithm compared to the regular k-nearest neighbour (kNN) algorithm. The exact number of neighbours varied for each technique. The results show that the VAE latent space representation performs best on the MNIST data set but not on the Fashion-MNIST data set. The Fashion-MNIST data set had the highest NMI scores with the cosine distance as the similarity measure.

The Markov clustering framework was able to cluster most of the elements of each true class in the same cluster for the MNIST data set. Obviously, at higher levels in the hierarchy, not all true classes were clustered separately from the rest. Although no real hierarchy exists for the MNIST data set, the clusterings across the hierarchy showed that the elements in the digit classes '4' and '9' are most similar to each other. They also showed a similarity relationship between the digit classes '3', '5' '8'. For the Fashion-MNIST data set, the true classes were also assigned a higher level category membership ('short-sleeved', 'long-sleeved', 'bottoms', 'shoes', 'accessories'). The results of the multi-scale clusterings on the Fashion-MNIST data set showed relatively good results for these higher level categories in comparison to the cluster assignments of the true classes. Only the elements of the class 'trouser' (also known as 'bottoms') were assigned their own cluster at all levels in the hierarchy.

Bibliography

[1] J. Jeon, V. Lavrenko, and R. Manmatha, “Automatic Image Annotation and Retrieval using Cross-Media Relevance Models Categories and Subject Descriptors,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 119–126, 2003.

[2] G. Nasierding, G. Tsoumakas, and A. Z. Kouzani, “Clustering based multi-label classification for image annotation and retrieval,” *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, no. October, pp. 4514–4519, 2009.

[3] M. Peikari, S. Salama, S. Nofech-Mozes, and A. L. Martel, “A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification,” *Scientific Reports*, vol. 8, no. 1, pp. 1–13, 2018.

[4] Y. Chen, J. Z. Wang, and R. Krovetz, “Content-based image retrieval by clustering,” *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2003*, pp. 193–200, 2003.

[5] D. Cai, X. He, Z. Li, W. Y. Ma, and J. R. Wen, “Hierarchical clustering of WWW image search results using visual, textual and link information,” *ACM Multimedia 2004 - Proceedings of the 12th ACM International Conference on Multimedia*, pp. 952–959, 2004.

[6] H. Steinhaus, “Sur la division des corp materiels en parties,” *Bull. Acad. Polon. Sci. IV (C1.III)*, vol. IV, no. 12, pp. 801–804, 1956.

[7] G. H. Ball and D. J. Hall, “ISODATA, a novel method of data analysis and pattern classification,” tech. rep., Menlo Park: Stanford Research institute, 1965.

[8] S. P. Lloyd, “Least Squares Quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[9] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–296, University of California Press, 1967.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[11] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, “Automatic subspace clustering of high dimensional data for data mining applications,” *ACM SIGMOD Record*, vol. 27, pp. 94–105, jun 1998.

[12] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[13] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, pp. 585–591, 2002.

[14] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1334–1339, 2005.

[15] I. T. Jolliffe, "Principal Components in Regression Analysis," pp. 129–155, Springer, New York, NY, 1986.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, pp. 91–110, 2004.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[19] Jianxin Wu and J. M. Rehg, "CENTRIST: A Visual Descriptor for Scene Categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1489–1501, aug 2011.

[20] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8258 LNCS, no. PART 1, pp. 117–124, 2013.

[21] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," *arXiv preprint arXiv:1511.05644*, nov 2015.

[22] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders," no. 2016, pp. 1–12, 2016.

[23] Q. Meng, D. Catchpoole, D. Skillicorn, and P. J. Kennedy, "Relational Autoencoder for Feature Extraction," *International Joint Conference on Neural Networks*, pp. 364–371, 2017.

[24] S. Dong, H. Xu, X. Zhu, X. F. Guo, X. Liu, and X. Wang, "Multi-view deep clustering based on autoencoder," *Journal of Physics: Conference Series*, vol. 1684, no. 1, 2020.

[25] S. Papadopoulos, C. Zigkolis, G. Tolias, Y. Kalantidis, P. Mylonas, Y. Kompatsiaris, and A. Vakali, "Image clustering through community detection on hybrid image similarity graphs," *Proceedings - International Conference on Image Processing, ICIP*, pp. 2353–2356, 2010.

[26] A. Bosaghzadeh and F. Dornaika, "Incremental and dynamic graph construction with application to image classification," *Expert Systems with Applications*, vol. 144, p. 113117, 2020.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," aug 2017.

[28] Z. Liu and M. Barahona, "Graph-based data clustering via multiscale community detection," *Applied Network Science*, vol. 5, no. 1, 2020.

[29] T. Berry and T. Sauer, "Consistent manifold representation for topological data analysis," *Foundations of Data Science*, vol. 1, no. 1, pp. 1–38, 2019.

[30] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Y. Liu, "Learning deep representations for graph clustering," *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, pp. 1293–1299, 2014.

[31] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised Deep Embedding for Clustering Analysis," in *International conference on machine learning*, pp. 478–487, 2016.

[32] X. Li, G. Cui, and Y. Dong, "Graph Regularized Non-Negative Low-Rank Matrix Factorization for Image Clustering," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3840–3853, 2017.

[33] J. Yang, D. Parikh, and D. Batra, "Joint Unsupervised Learning of Deep Representations and Image Clusters," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2016.

[34] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep Adaptive Image Clustering," in *2017 IEEE International Conference on Computer Vision (ICCV)*, vol. 2017-Octob, pp. 5880–5888, IEEE, oct 2017.

[35] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognition*, vol. 83, pp. 161–173, 2018.

[36] Z. Liu, Z. Lai, W. Ou, K. Zhang, and R. Zheng, "Structured optimal graph based sparse feature extraction for semi-supervised learning," *Signal Processing*, vol. 170, p. 107456, 2020.

[37] D. J. Trosten, M. C. Kampffmeyer, and R. Jenssen, "Deep Image Clustering with Tensor Kernels and Unsupervised Companion Objectives," no. 303514, pp. 1–12, 2020.

[38] Y. Yan, H. Hao, B. Xu, J. Zhao, and F. Shen, "Image Clustering via Deep Embedded Dimensionality Reduction and Probability-Based Triplet Loss," *IEEE Transactions on Image Processing*, vol. 29, pp. 5652–5661, 2020.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[40] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep Spectral Clustering using Dual Autoencoder Network," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4066–4075, 2019.

[41] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent Spece Clustering in Generative Adversarial Networks," in *Proceedings of the AAAI conference on artifical intelligence*, pp. 4610–4617, 2019.

[42] W. Dong, M. Charikar, and K. Li, "Efficient K-nearest neighbor graph construction for generic similarity measures," *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, pp. 577–586, 2011.

[43] S. M. van Dongen, *Graph Clustering by Flow Stimulation.* PhD thesis, 2000.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] A. Strehl and J. Ghosh, "Cluster ensembles - A knowledge reuse framework for combining multiple partitions," in *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[46] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, dec 1985.

[47] M. Meilă, "Comparing clusterings-an information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[48] G. Bouma, "Normalized (Pointwise) Mutual Information in Collocation Extraction," *Proceedings of German Society for Computational Linguistics*, pp. 31–40, 2009.

[49] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[50] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-100)," tech. rep., 1996.

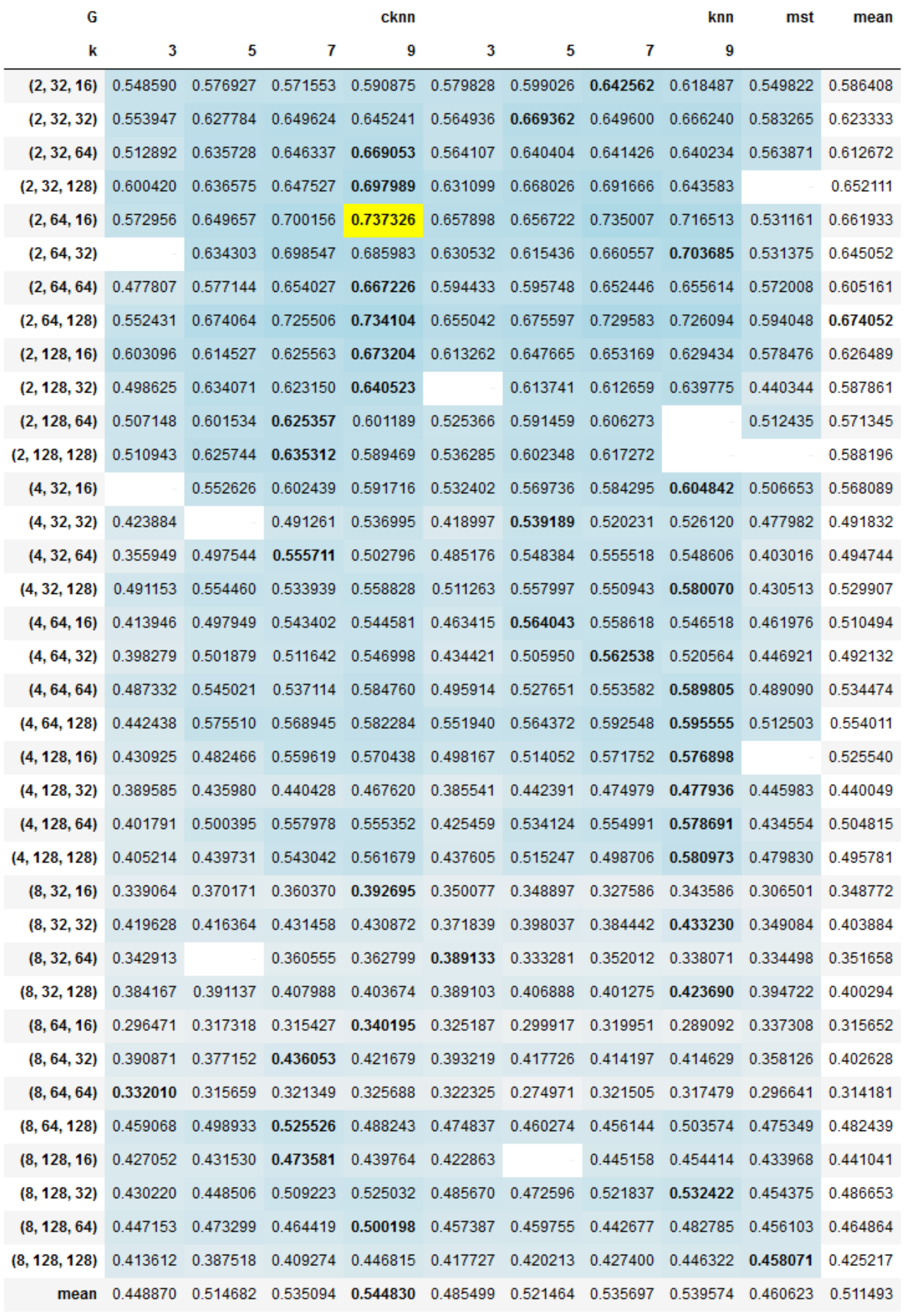
[51] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," pp. 248–255, 2009.

Chapter 7

Appendix

7.1 Model and graph testing

To determine the best variational autoencoder model and graph construction method, multiple tests were done on 500 balanced samples from the MNIST and Fashion-MNIST dataset. The NMI for each combination of model and graph construction method is reported below for all individual feature possibilities. For the model notation: {number of layers, number of hidden nodes in each layer, number of nodes in latent space}.



G	cknn				knn				mst	mean
k	3	5	7	9	3	5	7	9		
(2, 32, 16)	0.548590	0.576927	0.571553	0.590875	0.579828	0.599026	**0.642562**	0.618487	0.549822	0.586408
(2, 32, 32)	0.553947	0.627784	0.649624	0.645241	0.564936	**0.669362**	0.649600	0.666240	0.583265	0.623333
(2, 32, 64)	0.512892	0.635728	0.646337	**0.669053**	0.564107	0.640404	0.641426	0.640234	0.563871	0.612672
(2, 32, 128)	0.600420	0.636575	0.647527	**0.697989**	0.631099	0.668026	0.691666	0.643583		0.652111
(2, 64, 16)	0.572956	0.649657	0.700156	**0.737326**	0.657898	0.656722	0.735007	0.716513	0.531161	0.661933
(2, 64, 32)		0.634303	0.698547	0.685983	0.630532	0.615436	0.660557	**0.703685**	0.531375	0.645052
(2, 64, 64)	0.477807	0.577144	0.654027	**0.667226**	0.594433	0.595748	0.652446	0.655614	0.572008	0.605161
(2, 64, 128)	0.552431	0.674064	0.725506	**0.734104**	0.655042	0.675597	0.729583	0.726094	0.594048	**0.674052**
(2, 128, 16)	0.603096	0.614527	0.625563	**0.673204**	0.613262	0.647665	0.653169	0.629434	0.578476	0.626489
(2, 128, 32)	0.498625	0.634071	0.623150	**0.640523**		0.613741	0.612659	0.639775	0.440344	0.587861
(2, 128, 64)	0.507148	0.601534	**0.625357**	0.601189	0.525366	0.591459	0.606273		0.512435	0.571345
(2, 128, 128)	0.510943	0.625744	**0.635312**	0.589469	0.536285	0.602348	0.617272			0.588196
(4, 32, 16)		0.552626	0.602439	0.591716	0.532402	0.569736	0.584295	**0.604842**	0.506653	0.568089
(4, 32, 32)	0.423884		0.491261	0.536995	0.418997	**0.539189**	0.520231	0.526120	0.477982	0.491832
(4, 32, 64)	0.355949	0.497544	**0.555711**	0.502796	0.485176	0.548384	0.555518	0.548606	0.403016	0.494744
(4, 32, 128)	0.491153	0.554460	0.533939	0.558828	0.511263	0.557997	0.550943	**0.580070**	0.430513	0.529907
(4, 64, 16)	0.413946	0.497949	0.543402	0.544581	0.463415	**0.564043**	0.558618	0.546518	0.461976	0.510494
(4, 64, 32)	0.398279	0.501879	0.511642	0.546998	0.434421	0.505950	**0.562538**	0.520564	0.446921	0.492132
(4, 64, 64)	0.487332	0.545021	0.537114	0.584760	0.495914	0.527651	0.553582	**0.589805**	0.489090	0.534474
(4, 64, 128)	0.442438	0.575510	0.568945	0.582284	0.551940	0.564372	0.592548	**0.595555**	0.512503	0.554011
(4, 128, 16)	0.430925	0.482466	0.559619	0.570438	0.498167	0.514052	0.571752	**0.576898**		0.525540
(4, 128, 32)	0.389585	0.435980	0.440428	0.467620	0.385541	0.442391	0.474979	**0.477936**	0.445983	0.440049
(4, 128, 64)	0.401791	0.500395	0.557978	0.555352	0.425459	0.534124	0.554991	**0.578691**	0.434554	0.504815
(4, 128, 128)	0.405214	0.439731	0.543042	0.561679	0.437605	0.515247	0.498706	**0.580973**	0.479830	0.495781
(8, 32, 16)	0.339064	0.370171	0.360370	**0.392695**	0.350077	0.348897	0.327586	0.343586	0.306501	0.348772
(8, 32, 32)	0.419628	0.416364	0.431458	0.430872	0.371839	0.398037	0.384442	**0.433230**	0.349084	0.403884
(8, 32, 64)	0.342913		0.360555	0.362799	**0.389133**	0.333281	0.352012	0.338071	0.334498	0.351658
(8, 32, 128)	0.384167	0.391137	0.407988	0.403674	0.389103	0.406888	0.401275	**0.423690**	0.394722	0.400294
(8, 64, 16)	0.296471	0.317318	0.315427	**0.340195**	0.325187	0.299917	0.319951	0.289092	0.337308	0.315652
(8, 64, 32)	0.390871	0.377152	**0.436053**	0.421679	0.393219	0.417726	0.414197	0.414629	0.358126	0.402628
(8, 64, 64)	**0.332010**	0.315659	0.321349	0.325688	0.322325	0.274971	0.321505	0.317479	0.296641	0.314181
(8, 64, 128)	0.459068	0.498933	**0.525526**	0.488243	0.474837	0.460274	0.456144	0.503574	0.475349	0.482439
(8, 128, 16)	0.427052	0.431530	**0.473581**	0.439764	0.422863		0.445158	0.454414	0.433968	0.441041
(8, 128, 32)	0.430220	0.448506	0.509223	0.525032	0.485670	0.472596	0.521837	**0.532422**	0.454375	0.486653
(8, 128, 64)	0.447153	0.473299	0.464419	**0.500198**	0.457387	0.459755	0.442677	0.482785	0.456103	0.464864
(8, 128, 128)	0.413612	0.387518	0.409274	0.446815	0.417727	0.420213	0.427400	0.446322	**0.458071**	0.425217
mean	0.448870	0.514682	0.535094	**0.544830**	0.485499	0.521464	0.535697	0.539574	0.460623	0.511493

Table 7.1: MNIST: The NMI values are reported for all experiments to select a VAE model and graph construction method.

G	cknn				knn				mst	mean
k	3	5	7	9	3	5	7	9		
(2, 32, 16)	0.553401	0.577988	0.591916	0.631523	0.584379	**0.648224**	0.645938	0.629241	0.577359	**0.604441**
(2, 32, 32)	0.531260	**0.609972**	0.595953	0.597626	0.496504	0.568809	0.596692	0.596572	0.492977	0.565152
(2, 32, 64)	0.568418	0.598390	0.601929	**0.635798**	0.614907	0.635316	0.628879	0.632481	0.489166	0.600587
(2, 32, 128)	0.530186	0.519392	0.604340	**0.640785**	0.567273	0.578910	0.613341	0.626605	0.545988	0.580758
(2, 64, 16)	0.490787	**0.559974**	0.507411	0.551941	0.479240	0.481320	0.549050	0.530950	0.501832	0.516945
(2, 64, 32)	0.485438	0.557815	0.578546	0.554937	0.521700	0.573889	0.576625	**0.593610**	0.464123	0.545187
(2, 64, 64)	0.484747	0.527373	0.548558	0.589100	0.527490	0.547967	**0.598388**	0.593201	0.459630	0.541828
(2, 64, 128)	0.480113	**0.595526**	0.590674	0.573505	0.521766	0.561940	0.550522	0.585568	0.506286	0.551767
(2, 128, 16)	0.475336	0.566466	0.556107	0.575333	0.545940	0.555125	**0.597761**	0.588102		0.557521
(2, 128, 32)	0.463225	0.567720	**0.625177**	0.604565	0.524097	0.601044	0.595941	0.610290	0.518676	0.567859
(2, 128, 64)	0.421829	0.586900	0.624171	**0.624871**	0.540306	0.576708	0.583816	0.587184	0.514439	0.562247
(2, 128, 128)	0.471561	0.500506	0.543736	**0.554598**	0.430294	0.532606	0.551807	0.529144	0.423984	0.504249
(4, 32, 16)	0.519323	0.572896	0.579629	0.591183	0.477507	0.573208	0.599226	**0.604967**	0.493524	0.556829
(4, 32, 32)	0.512156	0.522672	0.547397	**0.583093**	0.530986	0.556651	0.570798	0.552051	0.452448	0.536472
(4, 32, 64)	0.429618	0.507335	0.510552	0.492849	**0.524420**	0.516231	0.512741		0.507732	0.500185
(4, 32, 128)	0.457686	0.496352	0.531863	**0.580226**	0.510405	0.545397	0.562453	0.579413	0.447798	0.523510
(4, 64, 16)	0.473884	0.510719	0.517060	**0.568524**	0.449369	0.559413	0.556780	0.533592	0.428353	0.510855
(4, 64, 32)	0.425373	0.389903	0.442196	0.440584	0.400307	0.465860	0.453020	**0.524909**	0.397000	0.437684
(4, 64, 64)	0.492230	0.490206	0.523832	0.526469	**0.540813**	0.517400	0.525892	0.535436	0.426338	0.508735
(4, 64, 128)	0.447540	0.489435	0.490023	0.524270	0.425678	**0.527784**	0.503581	0.516258	0.421993	0.482952
(4, 128, 16)	0.441021	0.478820	0.483512	**0.511744**	0.467415	0.478034	0.510409	0.508311	0.476744	0.484001
(4, 128, 32)	0.483591	0.481753	**0.525614**	0.523519	0.480772	0.512441	0.508385	0.495399	0.424170	0.492849
(4, 128, 64)	0.435529	0.464363	0.496793	**0.535833**	0.413369	0.509825	0.515778	0.527901		0.487424
(4, 128, 128)	0.416558	0.486446	0.491934	0.503852	0.437679	0.479783	0.490499	**0.510294**	0.472023	0.476563
(8, 32, 16)	0.549716	0.571855	0.588653	**0.619092**	0.522201	0.562214	0.507950	0.527833	0.569734	0.557694
(8, 32, 32)	0.574487	0.543632	0.606425	**0.606503**	0.529207	0.556236	0.600183	0.590064	0.557686	0.573825
(8, 32, 64)	0.580697	0.556873	0.565261	**0.619335**	0.521272	0.583428	0.593582	0.593905		0.576794
(8, 32, 128)	0.539962	0.592578	0.572043	0.583391	**0.600625**	0.558208	0.532837	0.582638	0.524482	0.565196
(8, 64, 16)	0.551074	0.596072	**0.622493**	0.604549		0.594312	0.591381	0.602859	0.543367	0.588263
(8, 64, 32)	0.471911	0.477498	0.537894	**0.570516**	0.450804	0.509542	0.503756	0.503767	0.515946	0.504626
(8, 64, 64)	0.465696	0.516159	0.527148	**0.529451**	0.444787	0.522480	0.513104	0.494496	0.491991	0.500590
(8, 64, 128)	0.533153	0.534432	**0.570349**	0.565728	0.552738	0.555072	0.563316	0.554174	0.504128	0.548121
(8, 128, 16)	0.541258	0.589788	**0.628355**	0.614803	0.560812	0.566524	0.494502	0.581122	0.521768	0.566548
(8, 128, 32)	0.560412	**0.601913**	0.597356	0.581745	0.547433		0.570312	0.591944	0.587988	0.579888
(8, 128, 64)	0.511848	0.483397	**0.532069**	0.504862	0.497910	0.466299	0.471874	0.447908	0.479546	0.488413
(8, 128, 128)	0.479399	0.539149	0.576579	0.560227	0.483922	0.554260	**0.578213**	0.556659	0.507074	0.537276
mean	0.495845	0.535063	0.556487	**0.568804**	0.506409	0.546642	0.553315	0.560539	0.492312	0.535662

Table 7.2: Fashion-MNIST: The NMI values are reported for all experiments to select a VAE model and graph construction method.