UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Physics and Technology

Deep Representation-aligned Graph Multi-view Clustering for Limited Labeled Multi-modal Health Data

Erland Grimstad

FYS-3941 Master's Thesis in Applied Physics and Mathematics

June 2022

Abstract

Today, many fields are characterised by having extensive quantities of data from a wide range of dissimilar sources and domains. One such field is medicine, in which data contain exhaustive combinations of spatial, temporal, linear, and relational data. Often lacking expert-assessed labels, much of this data would require analysis within the fields of unsupervised or semi-supervised learning. Thus, reasoned by the notion that higher view-counts provide more ways to recognise commonality across views, contrastive multi-view clustering may be utilised to train a model to suppress redundancy and otherwise medically irrelevant information. Yet, standard multi-view clustering approaches do not account for relational graph data. Recent developments aim to solve this by utilising various graph operations including graph-based attention. And within deep-learning graph-based multi-view clustering on a sole view-invariant affinity graph, representation alignment remains unexplored.

We introduce Deep Representation-Aligned Graph Multi-View Clustering (DRAGMVC), a novel attention-based graph multi-view clustering model. Comparing maximal performance, our model surpassed the state-of-the-art in eleven out of twelve metrics on Cora, CiteSeer, and PubMed. The model considers view alignment on a sample-level by employing contrastive loss and relational data through a novel take on graph attention embeddings in which we use a Markov chain prior to increase the receptive field of each layer. For clustering, a graph-induced DDC module is used. GraphSAINT sampling is implemented to control our mini-batch space to capitalise on our Markov prior.

Additionally, we present the MIMIC *pleural effusion* graph multi-modal dataset, consisting of two modalities registering 3 520 *chest X-ray* images along with two static views registered within a one-day time frame: *vital signs* and *lab tests*. These making up the, in total, three views of the dataset. We note a significant improvement in terms of separability, view mixing, and clustering performance comparing DRAGMVC to preceding non-graph multi-view clustering models, suggesting a possible, largely unexplored use case of unsupervised graph multi-view clustering on graph-induced, multi-modal, and complex medical data.

Acknowledgements

I would like to extend a most sincere thank you to my supervisors Professor Robert Jenssen and Sigurd Løkse for your immensely helpful feedback and guidance throughout this semester. Our meetings always left me feeling motivated and engaged.

I would also like to thank my classmates: Ask, Håvard, Nils, Sigurd, and Synne, for the helpful discussions and the lighthearted banter that we have had throughout these five years.

A heartfelt thank you goes out to my loved ones. To my dearest Helene, who has been infinitely supportive, kind, and loving, no matter the circumstance. To my parents Hilde and Stig, for your endless and loving support. And to my grandmother, Svanhild, whose good wit, kindness, and compassion is of great inspiration to me.

Erland Grimstad,
Tromsø, June 2022.

Contents

List of Figures

List of Tables

1 Introduction

The field of medicine is famously plagued with high workloads and hectic schedules [84; 101]. Much technological research is conducted on the topic of pattern recognition to aid in medical tasks (see figs. 1.2 & 1.3) looking to ease the work loads of physicians and health care workers. Easing the work load suffered by health care workers is an noble ambition, however, due to the aforementioned high demand by the health sector, expert annotators for the sake of medical pattern recognition research are few and far in-between. Although, while *data labels* (also referred to as *data annotations*) are largely unavailable, the medical field contains a vast mass of patient data from a wide range of sources and different modalities. In this thesis we explore the topic of *machine learning* (ML) without extensive guidance by expert annotations. Instead, we aim to utilise the many forms of data available to strengthen underlying group structures for the sake of prediction and visualisation of multi-modal medical data including chest X-rays (CXRs). A form of data that we found to be an important driving force in recognising cluster structures is that of *relational information* (illustrated in fig. 1.1) in the form of *affinity graphs*. When data annotations are unavailable, having available information of similarities between data samples prove to be an excellent aid in our endeavour. In the following, we provide an introduction to the field and details on the problem areas addressed in this thesis. Namely, an unexplored approach to graph-based multi-modal unsupervised machine learning on complex medical data for prediction and cluster analysis.

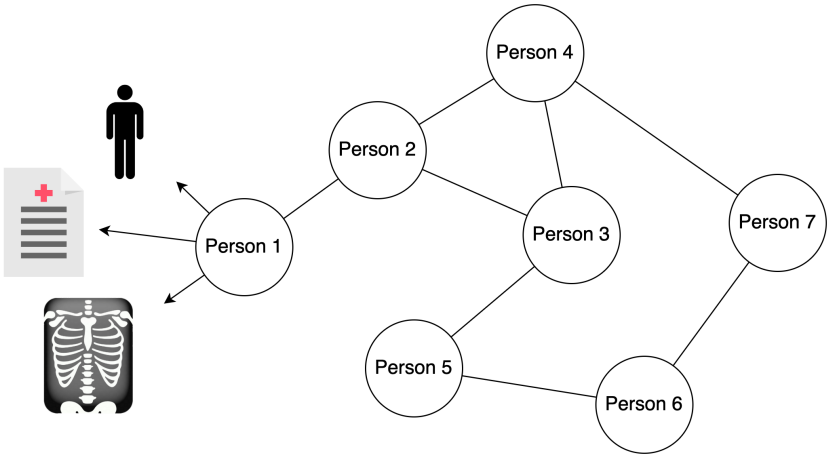
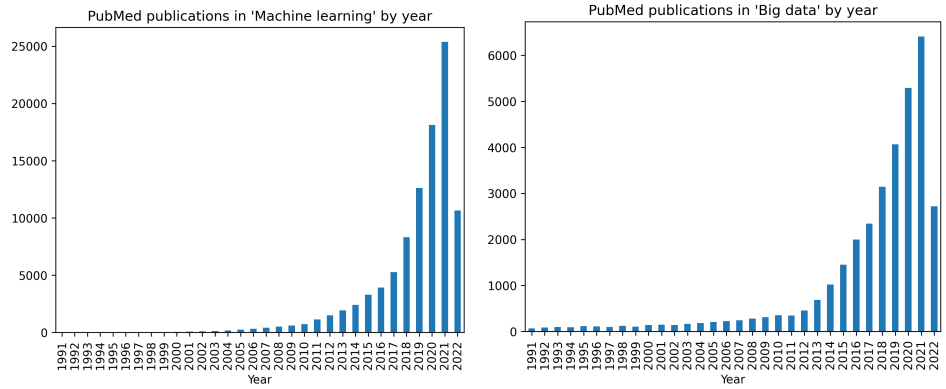


Figure 1.1: The relational information in datasets as shown by connecting lines.

1.1 Getting Information from Data

When we consider the sheer mass of accumulated data in many fields, manually analysis becomes, for the most part, infeasible without proper analysis tools. In addition to the volume, big data tends to be heterogeneous in type and significantly unstructured [20], making it yet harder to analyse. Today's solutions exist partly within the field of machine learning (ML), aiming to automate the learning process and provide useful tools to deal with and learn from large data masses. Machine learning is a cross-disciplinary subject combining computer science, mathematics, statistics, and information theory creating powerful data analysis methods of interpretation [71], generation (e.g. [26]), prediction (e.g. [79; 10]), and visualisation (e.g. [66]). This can be employed by presenting examples in a set of data points with associated *labels* in what is commonly referred to as *supervised machine learning*. The amount of research within the topic of machine learning has been increasing drastically over the last few decades, implying a strong belief in its potential as well as a demand for solutions within data analysis and pattern recognition (see figs. 1.2a & 1.2b).

Taking theory into practice will, in most cases, pose challenges. In machine learning, we see that although data is abundant, it is commonly imperfect through a combination of it being incomplete, inaccurate, and/or unlabeled. Regularly exposed to these imperfections is the field of medicine [20]. Medical data will often require hard-to-obtain expert-assessed labels, as well as devoid information, erroneous/lost lab tests, and exposure to the varying customs of caregivers and physicians. Machine learning fields such as *semi-supervised* and *unsupervised learning* (specifically clustering) attempt to learn data patterns



(a) PubMed publications containing the words ”machine learning”.

(b) PubMed publications containing the words ”big learning”.

Figure 1.2: Plots illustrating the machine learning trend in health research.

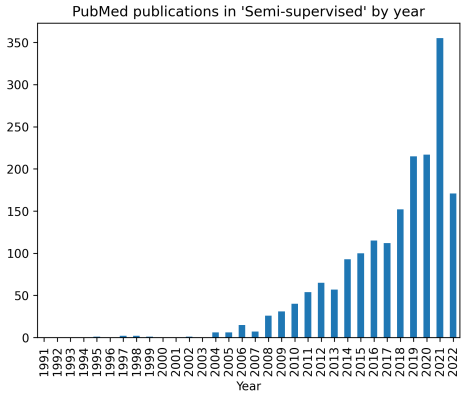


Figure 1.3: PubMed publications containing the words ”semi-supervised”.

despite the data being unlabeled — and the field is an increasingly appealing subject for medical research (see fig. 1.3). The problem of data incompleteness is commonly attempted neutralised by taking appropriate *pre-processing* measures, as will be adequately covered in chapter 6. Nevertheless, real-world data will contain bias, implicitly by default, or explicitly when attempting to neutralise the innate biases. Thus, having a fundamental understanding of the data at hand could prove beneficial both in pre-processing and *data imputation*.

1.2 Advantages of Data Diversity

Let us consider the task of teaching an algorithm to recognise a specific dog in an image by using a set of examples having annotations ”dog” or ”not dog”. Imagining that the majority of images taken view dogs located outside. Do we actually learn to recognise the dog, or do we recognise the grass it stands on, the trees in the background, or the blue sky above? We refer to this as the implicit bias of data — the presumptions made by the model implicitly through the data that was presented. Now, let us consider two approaches to deal with this bias. Firstly, to actively select a subset of the training dataset containing balanced image properties, explicitly introducing bias into the data in an attempt to neutralise the pre-existing bias. Secondly, to adjust our machine learning approach to improve its learning abilities. For instance, if we were to consider both the image as well as a newfound knowledge of the dog’s colour and size, we would have an advantage in our task by recognising the sections of the image which is more likely to contain the dog. *Multi-view learning* as a ML approach attempts to unify learning across different sources — or *views* — such as in our example. It builds on the intuition that additional *informative* and *non-redundant* views will improve machine learning models in the recognition of complex patterns by singling out the common properties. When data from a range of different sources is readily available, yet labeled training data is unavailable, *multi-view clustering* (MVC) may often prove to be a suitable solution.

Multi-view learning is well-suited for unambiguous datasets, however, when similar data features may be interpreted differently depending on latent information, we might yet find MVC to be lacking. In the case of having available similarity values between samples in the dataset, similarly valued features may yet be interpreted differently depending on their associated samples. Having these features in an affinity graph, we may unite the advantages of having multiple views and accessible graph data to the method of graph-based multi-view learning.

1.2.1 Graph Multi-View Clustering and Health Data

As we have covered, in medical data is both abundant and from a wide range of modalities and sources (e.g. figure 1.4) [20]. It is therefore a promising place to start for graph-based multi-view machine learning approaches. However, the limited capacity of medical experts for constant data labelling makes fully supervised learning hard to accomplish. We hypothesise that managing to merge information across views by recognising groupings that discovers commonality across different sources and modalities while taking sample similarity into

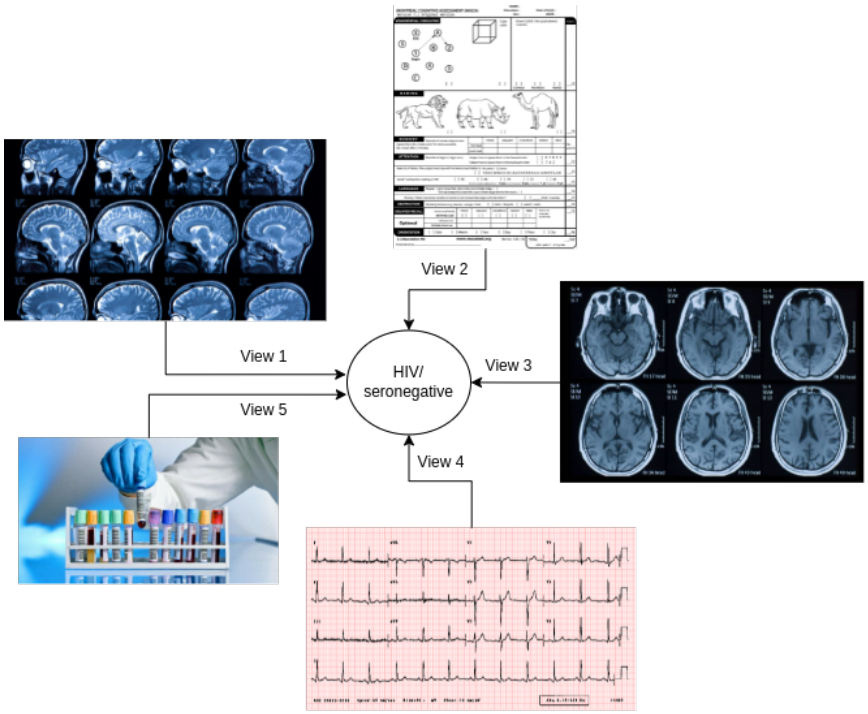


Figure 1.4: Use cases of multi-view learning in healthcare [9].

account, we reason that the dependency on a priori assessments may lessen — enabling solutions requiring less supervision on significantly complex data. Other challenges posed include (i) data selection by selecting non-redundant and informative data views; (ii) the high interconnected nature of medical diagnostics, requiring extensive datasets for noise smoothing (having sufficient data to get a accuracte portrayal of the assumed underlying diagnostic distributions) and yielding highly correlated predictions that does not necessarily isolate any singular diagnosis — simply recognising a correlated subgroup; (iii) the resulting clustering lacking interpretability.

Hypothesising that the extraction of features by recognising commonality across views on a sample-level may provide added cluster stability and performance benefits for intricate data, we introduce the model to be presented in this thesis (chapter 5). We also hypothesise that we may extract diagnostically relevant information from limited labeled data — unsupervised or partially labeled in a semi-supervised manner — by using graph-based multi-modal clustering with *contrastive learning* in the form of *representation alignment*, as in [104]. Other approaches to contrastive graph-based multi-view clustering is commonly within the fields of representation learning and/or graph learning [33; 78; 109; 95; 40]. Meanwhile, to the best of the author's knowledge, no other

graph-based MVC models employ representation alignment on a predermined graph multi-view dataset without employing representation learning and/or graph learning. Thus, in this thesis we aim to present a model adhering the following set of model features with the core intention of *performing clustering as close to the raw input data as possible*:

- The machine learning model should learn to extract diagnostically relevant information from each view separately and utilise this information in union in its clustering.
- The machine learning model should learn to nuance information by taking both data features and relational information into consideration. Each sample in each view should be fused in a representation space in which similar nodes are located close together.
- The machine learning model should learn to attend to similar neighbouring samples in the dataset by an attention mechanism.
- The model should *not* learn new views but cluster purely on the provided multi-view data (unlike multi-view approaches such as [33]).
- The model should *not* learn graphs during training, but utilise one predetermined view-invariant graph (unlike works such as [78; 109; 95; 40]).

1.2.2 No Benchmark Dataset Exists

While publicly available databases are readily available [47; 49; 24], we fail to identify any graph-based multi-modal datasets for training of graph multi-view neural networks for lung diagnostics: namely the prediction of *pleural effusion* from CXR-assisted multi-view clustering. We construct a novel medical dataset containing multi-modal and relational data for *pleural effusion/not pleural effusion* patients for classification. The dataset to be detailed (chapter 6) should comply to the following:

- The dataset should contain at least two different data modalities that provide non-redundant and relevant information (and otherwise conform to the multi-view principles as will be covered in chapter 4).
- The dataset should contain one general *homogeneous graph* shared across views.
- The dataset features should be extracted without feature selection to assess the ability of unsupervised feature selection by models.

- The underlying ground truth classes (diagnoses) should not be used in the construction of the dataset in any way.

1.3 Contributions

In this thesis, we present *DRAGMVC*: a novel representation alignment-based graph MVC model evaluated in unsupervised and semi-supervised applications. The model extends on the *representation alignment* approach in [104], i.e. by teaching a model to extract relevant features by recognising commonality across data views. In our dog image classification example this correspond to, e.g., recognising that the object in the image is both *brown* and *has a snout*, thus learning to recognise relevant commonality across data views. We reason that the sample-to-sample approach should yield more informative data representations that could prove useful in certain real-world applications in which nuance is crucial. This is an unexplored approach for medical diagnostic purposes to the best of the author's knowledge. Models such as MAGCN [13] being the established state-of-the-art within deep graph-based multi-view clustering do not align representations directly, which we assume to weaken their feature extraction capabilities on sufficiently complex datasets such as our constructed MIMIC dataset. We attribute this to the very general nature of aligning distributions, allowing for a significant degree of intra-cluster variation, which we assume as likely to be a cause of instability on their clustering performance. In the following thesis we display, by the use of DRAGMVC, that the degree of representation alignment is significantly strengthened by learning from relational information in clustering.

We evaluate the model on our novel medical dataset to evaluate its usefulness for highly complex data as well as its limited labeled data performance and compare it to previous, similar clustering models. In evaluation we consider the predictions' subgroups in terms of diagnoses, as well as the statistical significance of the resulting diagnosis splits. The thesis presents two main contributions. Firstly, DRAGMVC introduces the *graph attention convolutions using a Markov prior* improving on existing MVC-targeted attention-based graph convolutions by allowing for a more selective attention approach by using a Markov chain model as a statistical prior. This will allow for potentially further-reaching attention than simple graph masking and a more selective attention than attending to the full dataset. Due to the learned attention, this serves as a potential dampening effect to the over-smoothing problem regularly encountered in graph neural networks [76].

Our second contribution in this thesis is a novel medical dataset constructed from two publicly available databases: MIMIC-IV and MIMIC-CXR [47; 49; 24], using ground truth classes *pleural effusion/not pleural effusion*. The dataset contains relational data generated from radiology reports in a fully unsupervised manner. For our node attributes, the data uses two different modalities: vectorial and image data, and three data views in total: *vital signs*, *lab tests*, and *chest X-rays*. Our aim is to evaluate the quality of information contained in our dataset, as well as whether it complies with the non-redundancy presumption of multi-view datasets — as demonstrated by the "dog/no dog" example. The MIMIC dataset will be evaluated using state-of-the-art methods including the model presented in this thesis and its non-graph counterpart. From this analysis, we aim to assess both diagnostic capabilities and any significant effect on interpretability due to the employment of relational data. In addition to our constructed MIMIC dataset, we perform comparisons on three common graph multi-view datasets (Cora, CiteSeer, and PubMed [93; 73]) in order to assess DRAGMVC's performance in comparison to the established state-of-the-art within deep learning graph-based multi-view clustering with a single view-invariant affinity graph.

1.4 Thesis Outline

Seeing as our method employs methods from graph learning, unsupervised learning, classic neural networks, and builds on foundational multi-view concepts and advanced multi-view approaches — we will cover the necessary theory: from basic clustering of data features and graphs, to neural networks, and finally, the current state-of-the-art within graph-based multi-view learning and clustering. These make up the three chapters of *clustering* (chapter 2), *deep learning* (chapter 3), and *multi-view learning* (chapter 4), respectively. Following the background, theory detailing both of our main contributions will be covered thoroughly in chapters 5 & 6. Chapter 7 covers experiments aiming to challenge questions raised regarding our model's comparative performance and its potential for complex medical applications. The chapter opens with descriptions of datasets used, model setup, and the tools used in result analysis. Results are presented along with their interpretation. The thesis finalises with a comprehensive discussion of the results, providing a rendition of the contributions made and their strengths and weaknesses. Finally, we conclude the thesis in chapter 8 along with propositions of potential future work.

The thesis builds upon work done in the project paper preceding this thesis: *Exploring Multi-modal Medical Data with Deep Multi-view Clustering* [27]. Some of the content in the thesis, including parts of the introduction, are reused

from this paper. Namely, chapters 2-4 covering the background theory for the thesis is widely reused from [27]. Notable alterations in the background theory include the addition of sections covering graph neural networks, semi-supervised learning, graph-based multi-view networks, and many lesser or inconsequential changes. Similarly, chapter 6 covering the construction of our multi-view dataset is largely drawn from [27] with the exception of the creation of our affinity graph and distinct changes in the selection of static views and ground truth classes. Lastly, the analysis methods using in the experiments (covered in section 7.2) are also reused from [27] due to the similarities of the medical dataset in use.

Part I

Background

2

Clustering

In this chapter, clustering methods will be covered. We include these methods to build a fundamental understanding of the variety, potential, and limitations of clustering — aiming to build the knowledge necessary in order to properly cover the main contributions of this thesis. Clustering can be divided into multiple categories and sub-categories. A simple way of categorisation follows from the structure of the resulting groupings. Does it partition the data into defined clusters or a tree-like hierarchical cluster structure? These are appropriately named partitional and hierarchical clustering, respectively. It is also worth mentioning that traditional clustering methods can be used in conjunction: in an ensemble of machine learning methods, i.e. ensemble learning [23]. Naturally, this group falls into a superposition of the two clustering categories previously mentioned. Partitional clustering can be further bisected into centroid-based (as seen in fig. 2.1) and affinity-based methods. In other words, the clustering methods presented will cover pattern recognition methods on either data features or relational information.

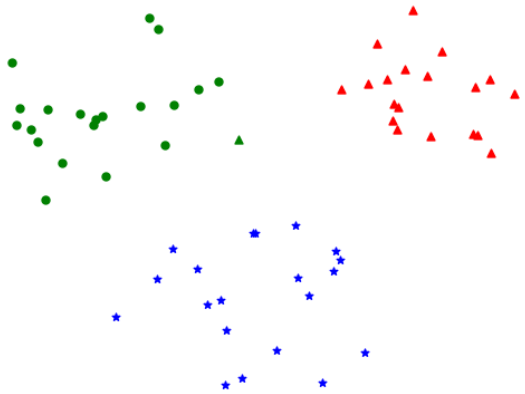


Figure 2.1: Partitional clustering on a dataset generated from three random Gassian distributions. Marker symbol denotes the distribution from which a given point was generated. Colour denotes predictions.

2.1 Clustering Definitions

2.1.1 Clusters

Let $\boldsymbol{X}$ be our data set which is to be clustered, defined as

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}. \tag{2.1}$$

Now, a partitioning of $\boldsymbol{X}$ is defined as the m clusters $C_1, C_2, ..., C_m$, if the following three conditions are met [103]:

(i) $C_i \neq \emptyset, i = 1, ..., m$

(ii) $\cup_{i=1}^{m} C_i = \boldsymbol{X}$

(iii) $C_i \cap C_j = \emptyset, i \neq j, \quad i, j = 1, ..., m$

In other words: no cluster set is allowed to be empty, the union of all clusters must add up to the complete data set, and the intersection of any two clusters must be an empty set, i.e. none of the data points may be classified as being a part of multiple clusters. In addition to these three conditions, clusters should reflect some similarity in its set. Data points contained in any given cluster should be more similar to each other and less similar to points in another cluster [103].

In the above clustering definition, each data point is assigned to a sole cluster. This is referred to as *hard memberships*. Alternatively, *fuzzy memberships* [121]

may be applied in situations where the class assignments are less clear-cut. In this case, data points' memberships are given as a partial affinity with all clusters that must add up to 1. Looking at our $\boldsymbol{X}$, 2.1, fuzzy memberships are defined as [103]:

$$u_j : \boldsymbol{X} \rightarrow [0, 1], j = 1, ..., m \tag{2.2}$$

where u_j is the membership for cluster j, with memberships in $\boldsymbol{X}$.

$$\sum_{j=1}^{m} u_j(\boldsymbol{x}_i) = 1, i = 1, ..., N \tag{2.3}$$

$$0 < \sum_{i=1}^{N} u_j(\boldsymbol{x}_i) < N, j = 1, ..., m. \tag{2.4}$$

This means that each data point's membership sums up to 1 — potentially distributed among multiple clusters — and that each cluster must partially contain at least one data point and also cannot fully contain all data points.

2.1.2 Proximity Measures

Proximity measures describe the similarity (and dissimilarity) of two points, $\boldsymbol{x}$ and $\boldsymbol{y}$, in feature space. The most common dissimilarity measure [1] is the weighted l_p metric [103]

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{l} w_i |x_i - y_i|^p \right)^{1/p} \tag{2.5}$$

for features along dimensions $i = 1, ..., l$

The simple, unweighted variants correspond to familiar distance metrics, e.g. we may recognise the unweighted l_2 metric as the Euclidean norm

$$d_2(\boldsymbol{x}, \boldsymbol{y} | \boldsymbol{W} = \boldsymbol{I}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T (\boldsymbol{x} - \boldsymbol{y})}. \tag{2.6}$$

Sometimes used when considering a difference between two identically shaped matrices, i.e. $\boldsymbol{A} = \boldsymbol{X} - \boldsymbol{Y} \in \mathbb{R}_{m \times n}$, we may use the Frobenius norm given as such

$$||\boldsymbol{A}||_F = \sqrt{\sum_{i}^{m} \sum_{j}^{n} |A_{ij}|^2}. \tag{2.7}$$

1. A measure where higher values corresponds to two points being more dissimilar

In the special case where dissimilarity is to be measured between clusters represented in feature space as probability density functions (PDFs) — a divergence measure may be more appropriate than the previous methods. In the multi-cluster ($k \geq 2$) case, a well-suited measure is the Cauchy-Schwarz (CS) divergence [46]. Defined as:

$$D_{cs}(p_1, ..., p_k) = -\log\left(\frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\int p_i(\boldsymbol{x})p_j(\boldsymbol{x})d\boldsymbol{x}}{\sqrt{\int p_i^2(\boldsymbol{x})d\boldsymbol{x}\int p_j^2(\boldsymbol{x})d\boldsymbol{x}}}\right) \tag{2.8}$$

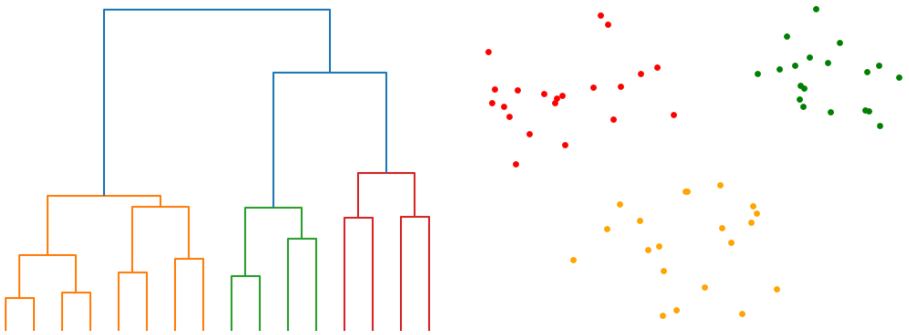
for PDFs p_i and p_j. The intuitive understanding of divergence measures is to quantify the degree with which two probability density functions overlap. This is often desired in order to optimise one's feature representation to fit a desired distribution. This approach is used in e.g. the deep divergence-based clustering module [50] employed in DRAGMVC.

2.2 Outlining Hierarchical Clustering

Hierarchical clustering can be split into two defining methods: agglomerative clustering and divisive clustering [6]. The former of which constructs a tree of clusters by initially having N separate clusters — one for each data point — and merging clusters that minimise a predetermined distance metric until a sole cluster remains. A dendrogram may then be used to display the clustering results and in turn find natural groupings in the data set (as can be seen in figs. 2.2a and 2.2b) by assigning hard memberships at an optimal threshold along the y-axis, which in this case represents Euclidean distance. Agglomerative clustering may be further differentiated by the selection of distance metric, and equally importantly its measuring method. Notable variants of the latter include the single-link [2] [98] and complete-link [3] [55] algorithms.

The process for divisive clustering can be simplified as the reverse of that in agglomerative clustering. The dataset is now initialised as a single cluster. As opposed to the merging one might associate with agglomerative clustering, the objective is now to find the cut (the *graph cut* will be covered later in this chapter) that yields maximum between-cluster distance until there is one cluster for each data point in the dataset. Similar to agglomerative clustering, the results can be displayed in a dendrogram from which natural groupings can be extracted and clustered together.

2. Measuring distance based on the pair of nearest data points in different clusters
3. Distance measured at furthest distance between points in different clusters



(a) Dendrogram of agglomerative clustering by the complete-linkage algorithm.

(b) The resulting clusters from the three coloured clusters in fig. 2.2a.

Figure 2.2: Agglomerative clustering on an identical data distribution to the one in fig. 2.1.

Hierarchical clustering is very suitable when the objective is to gather an understanding of the underlying structure of the data. However, when the objective is to obtain memberships or explicit predictions, rather than the tree-like pattern we see in a dendrogram (2.2a), partitional methods are frequently more suitable.

2.3 Partitional Methods

The idea behind partitional methods is to partition data points into predicted classes, rather than describe their data hierarchy. One such approach is our proposed model, DRAGMVC, aiming to assign predicted labels to data points.

2.3.1 Centroid-based Methods

2.3.1.1 The Generalised Hard Algorithmic Scheme

The Generalised Hard Algorithmic Scheme (GHAS) [103] is a generalised formulation of centroid-based hard clustering which can be used to describe a wide range of known clustering methods. We may use the definitions for fuzzy clustering in eqs. 2.2, 2.3, & 2.4 to describe hard clustering as well, in the case

of a one-hot vector such as [103]

$$u_{ij} \in \{0, 1\}, \quad j = 1, ..., m \tag{2.9}$$

$$\sum_{j=1}^{m} u_{ij} = 1, \quad i = 1, ..., N \tag{2.10}$$

$$\tag{2.11}$$

with m clusters and N data points. The loss function is given as a sum over dissimilarities to clusters $j = 1, ..., m$ by u_{ij}, defined as

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} d(\boldsymbol{x}_i, \boldsymbol{\theta}). \tag{2.12}$$

Equation 2.12 is minimised for the cluster $j = 1, ..., m$ that minimises the dissimilarity $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$, i.e. for a one-hot vector $\boldsymbol{x}_i$:

$$u_{ij} = \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \min_{k=1,...,m} d(\boldsymbol{x}_i, \boldsymbol{\theta}_k) \\ 0, & \text{otherwise} \end{cases}. \tag{2.13}$$

We update the clusters $\boldsymbol{\theta}_j$ by solving $\dfrac{\partial J(\boldsymbol{\theta}, U)}{\partial \boldsymbol{\theta}_j}$ for $j = 1, ..., m$ to find our loss minimum:

$$\sum_{i=1}^{N} u_{ij} \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = 0, \quad j = 1, ..., m. \tag{2.14}$$

Now, combining eqs. 2.13 and 2.14 into the GHAS clustering scheme given in algorithm 1 [103]. Its *termination criterion* can be any arbitrary criterion, e.g. no change in cluster coordinates, a set number of iterations, or no cluster assignment change.

2.3.1.2 K-means

K-means [67] is often regarded as the go-to clustering method because of its simplicity and efficiency [44]. As such, K-means will be used throughout the paper where it is deemed appropriate and sufficient in quality. Like many other popular clustering methods it is shown to be a variant of the GHAS. The dissimilarity measure used is the squared Euclidean distance, $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = (\boldsymbol{x}_i - \boldsymbol{\theta}_j)^2$. Thus, we may rewrite eq. 2.12 as

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} ||\boldsymbol{x}_i - \boldsymbol{\theta}_j||^2. \tag{2.15}$$

Algorithm 1 Generalised Hard Algorithmic Scheme (GHAS) algorithm.

Initialise $\boldsymbol{\theta}_j(0), \quad j = 1, ..., m$.
$t \leftarrow 0$
while termination criterion not met **do**
  for $i \leftarrow 1$ to N **do**
    for $j \leftarrow 1$ to m **do**

$$u_{ij}(t) \leftarrow \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{\theta}_j(t)) = \min_{k=1,...,m} d(\boldsymbol{x}_i, \boldsymbol{\theta}_k(t)) \\ 0, & \text{otherwise} \end{cases}$$

    end for
  end for
  $t \leftarrow t + 1$
  for $j \leftarrow 1$ to m **do**
    Solve the following w.r.t. $\boldsymbol{\theta}_j$ to yield solutions $\boldsymbol{\theta}'_j$

$$\sum_{i=1}^{N} u_{ij}(t-1) \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = 0$$

    $\boldsymbol{\theta}_j(t) \leftarrow \boldsymbol{\theta}'_j$
  end for
end while

The update value eq. 2.14 can be found from eq. 2.15 by first solving

$$\begin{aligned}\frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} &= \frac{\partial}{\partial \boldsymbol{\theta}_j}\left((\boldsymbol{x}_i - \boldsymbol{\theta}_j)^2\right) \\ &= -2(\boldsymbol{x}_i - \boldsymbol{\theta}_j) \end{aligned} \tag{2.16}$$

then solving the update equation 2.14

$$\begin{aligned} 0 &= -2\sum_{i=1}^{N} u_{ij}(\boldsymbol{x}_i - \boldsymbol{\theta}_j) \\ \boldsymbol{\theta}_j \sum_{i=1}^{N} u_{ij} &= \sum_{i=1}^{N} u_{ij}\boldsymbol{x}_i \\ \boldsymbol{\theta}'_j &= \frac{\sum_{i=1}^{N} u_{ij}\boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}}. \end{aligned}$$

Recognising $\sum_{i=1}^{N} u_{ij} = N_j$ as the numbers of data points of cluster C_j and $\sum_{i=1}^{N} u_{ij}\boldsymbol{x}_i$ as a summation of the data points in cluster C_j we may simplify the above equation to

$$\boldsymbol{\theta}'_j = \frac{\sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i}{N_j}. \tag{2.17}$$

This is recognised as the mean of data points $\boldsymbol{x}_i \in C_j$. The intuitive understanding of this algorithm is simple. First, you assign each data point a cluster by finding the nearest centroid by the squared Euclidean distance. Then, update the centroid positions by taking the means of the newly assigned data points within each cluster. Repeat this until convergence or until another termination criterion is achieved.

2.4 Affinity-based Transformations & Clustering

To continue building our perception of data we introduce the concept of affinity graphs. In a mathematical context, we employ graphs to define relational context. While data features may provide an arbitrary mass of data for a single sample, the data is commonly self-describing, disregarding affinity to other data points in the dataset. This thesis presents machine learning approaches uniting the sample-bound data with contextual data. Thus, we find it highly relevant to cover the topics of graphs and graph-based machine learning methods, the latter of which will be detailed further in chapter 3.3. In this section, we will also detail a dimensionality reduction method commonly used in visualisation of graph structures. Finally, we will introduce the foundational graph clustering method *spectral clustering*.

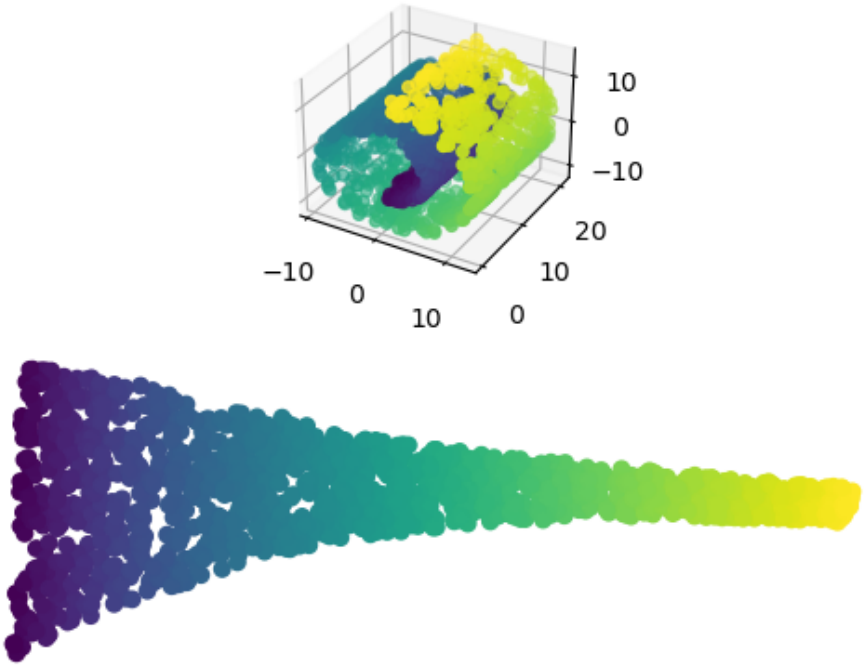


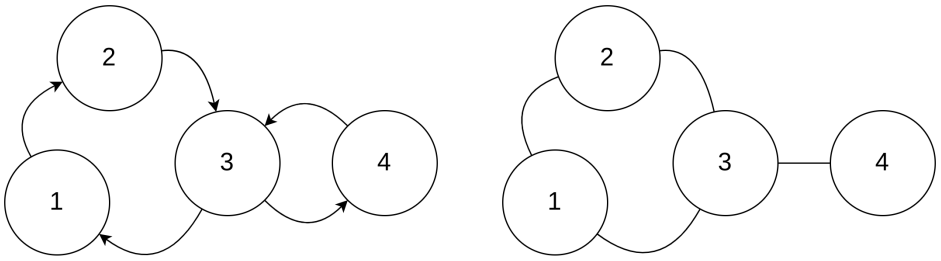
Figure 2.3: An example of a graph-based transformation [89].

2.4.1 Graph Definitions

Graph-based transformations may be used in cases where patterns are required to take some grander context into consideration (as illustrated in fig. 2.3). This underlines the necessity of graph data in certain case, but prior to covering graph transformations and graph clustering, we will detail a few underlying definitions.

A graph $G = (V, E)$ consists of vertices (or nodes) and edges. It defines a similarity of a connection from vertex v_i representing data point $\boldsymbol{x}_i$ by an edge — representing the vertex's connections to $\boldsymbol{x}_j$ by a weight w_{ij}. The edge weights define proximity or similarity, and are commonly stored in an *affinity matrix* $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. In the case that w_{ij} has no associated floating point value, the graph is defined as *unweighted* [74]. In these cases edges connecting two vertices are simply defined as present or not present. Graphs may further be categorised by properties of their vertices and edges, for example [124]:

- *Directed / undirected graphs* : Edges in directed graphs have a direction, meaning that any node is directed to another (see fig. 2.4a). This is



(a) A directed graph. **(b)** An undirected graph.

Figure 2.4: Examples of directed/undirected graphs.

not the case for undirected graphs (fig. 2.4b), however, each edge in an undirected graph may be represented as a set of directed edges (e.g. node 3 in fig. 2.4a directing to node 1 and node 4).

- *Homogeneous / heterogeneous graphs* : A graph is homogeneous if vertex-edge relation type described by the graph is the same for all nodes. In contrast, relations in heterogeneous graphs may consist of different types for various nodes[4].

- *Static / dynamic graphs* : Whether the graph topology is fixed or vary with time.

With cases such as distance matrices including every edge may prove excessive, thus, the affinity matrix itself may be aptly filtered according to its application. An example of this is seen in the ”swiss roll” figure (fig. 2.3) in which we commonly aim to purposefully disregard datapoints on the other side of the space separating the layers of the roll. For the standard machine learning case where graph vertices are associated with data in some vector space, we define the following graph alterations

- *k Nearest neighbour* (k-NN) : Considering only the k nearest points for each vertex v_i, setting the weight of all remaining edges to zero $w_{ij} = 0$ if data point $\boldsymbol{x}_j$ is not a k nearest neighbour. Using a similarity measure $w_{ij} = s_{ij}$ (such as Euclidean distance) for the neighbours.

- *ε neighbourhood* : Considering all points within a threshold ε. Setting the weights of all remaining data points $w_{ij} = 0$.

4. Example of a heterogeneous graph: A marked place graph may consist of nodes *seller*, *buyer*, and *product*. These are connected by edges *selling*, *buying*, *in-marked-for*, and *has-bought*. In this example, both vertices and edges vary in type.

- *Fully connected graph* : Using all points in a graph.

2.4.2 Graph Laplacian

From the definitions we have established, we may utilise graph structures to transform or cluster data. We define the graph Laplacian as a matrix given by

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \tag{2.18}$$

having $\boldsymbol{A}$ be our affinity matrix and $\boldsymbol{D}$ is the diagonal *degree matrix* with elements

$$D(i,j) = \begin{cases} \deg(v_i) & i = j \\ 0 & i \neq j \end{cases},$$

with $\deg(v_i) = \sum_{j=1}^{N} w_{ij}$.

Data patterns found in graphs may be transformed to lower dimensions by applying *Laplacian eigenmaps* [103]. Having $y_i \in \mathbb{R}^l$ where $l \ll p$, let $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_N\}$ be a set of transformed representation of high-dimensional data $\boldsymbol{X} \in \mathbb{R}^{N \times p}$. The loss function aiming to optimise $\boldsymbol{y}$ should be set so that high-valued weights w_{ij} correspond to small squared distances (in transformed space) $(y_i - y_j)^2$, while small weights correspond to well-separated nodes. As such, the loss function is defined as follows

$$J(\boldsymbol{y}, \boldsymbol{A}) = \sum_{i=1}^{N} \sum_{j=1}^{N} (y_i - y_j)^2 w_{ij}. \tag{2.19}$$

We have the equality

$$\begin{aligned}
\boldsymbol{y}^T \boldsymbol{L}\boldsymbol{y} &= \boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{y} \\
&= \boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} \\
&= \sum_{i=1}^{N} y_i^2 d_i - \sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j w_{ij} \\
&= \frac{1}{2}\left(\sum_{i=1}^{N} y_i^2 d_i - 2\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j w_{ij} + \sum_{j=1}^{N} y_j^2 d_j\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} y_i^2 w_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N} 2y_i y_j w_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N} y_j^2 w_{ij}\right) \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(y_i^2 - 2y_i y_j + y_j^2) w_{ij} \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(y_i - y_j)^2 w_{ij}. \qquad (2.20)
\end{aligned}$$

Thus, by eqs. 2.19 & 2.20

$$J(\boldsymbol{y}, \boldsymbol{A}) \propto \boldsymbol{y}^T \boldsymbol{L}\boldsymbol{y}. \qquad (2.21)$$

By optimising eq. 2.21 with regards to $\boldsymbol{y}$ subject to $\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y} = 1$ in order to find its minimum, the eigenmap may be derived as follows by eigen-decomposition

$$\boldsymbol{L}\boldsymbol{a} = \lambda \boldsymbol{D}\boldsymbol{a}. \qquad (2.22)$$

Consider $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_l$ be the smallest $l+1$ eigenvalues[5]. By using the l eigenvectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_l$ as our basis[6] a Laplacian eigenmap $\boldsymbol{Y} \in \mathbb{R}^{N\times l}$ is produced, e.g. fig. 2.3.

The map yields a new representation where the graph connections are restricted such that highly weighted edges are mapped to be close, while nodes corresponding to smaller edge weights are pushed away. Using filtering methods such as k-NN, only local structures are considered and put in a grander context, allowing structures such as the on in fig. 2.3 to be solved properly by

5. Note that unlike the notation used in PCA, the eigenvalues are ordered in ascending rather than descending order.
6. Ignoring $\lambda_0 = 0$ which corresponds to the zero mapping solution.

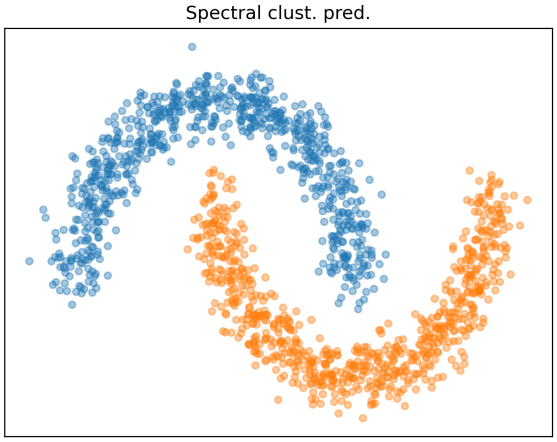


Figure 2.5: Spectral clustering predictions for complex data distributions.

using Laplacian eigenmaps.

In many cases it is useful to consider the *normalised graph Laplacian*, defined as

$$\tilde{\boldsymbol{L}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I}_N - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}. \tag{2.23}$$

The normalised graph Laplacian is a symmetric positive semidefinite matrix[7] [124], making it useful in eigen-decomposition problems such as the one in eq. 2.22.

2.4.3 Spectral Clustering

As opposed to partitional clustering, affinity-based clustering methods are based by minimising the *graph cut*. Having two cluster candidates, A and B, we define the graph cut as [103]

$$\text{cut}(A, B) = \sum_{i \in A} \sum_{j \in B} w_{ij}. \tag{2.24}$$

The optimal cut is one where where the sum of weights connecting points in A with points in B is minimised. This does, however, often favour small and sparse groupings [103]. In order to avoid this issue, the normalised cut was introduced [96]. This variant attempt to correct the original graph cut's bias

7. Matrix $\boldsymbol{A}$ is positive semidefinite if $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq 0$ for all real and non-zero $\boldsymbol{x}$ [1].

towards sparse groupings and has become one of the most used criteria in spectral clustering [103]. The normalised cut is defined as [96]

$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{V(A)} + \frac{\text{cut}(A, B)}{V(B)}, \tag{2.25}$$

where we define the *volume* as [96]

$$V(A) = \sum_{i \in A, j \in V} w_{ij}, \tag{2.26}$$

having $A \cup B = V$.

To ease computation we approximate the normalised cut, defining a label vector $\boldsymbol{y} = [y_1, ..., y_N]^T$ with elements

$$y_i = \begin{cases} \dfrac{1}{V(a)} & \boldsymbol{x}_i \in A \\ \dfrac{1}{V(B)} & \boldsymbol{x}_i \in B \end{cases}. \tag{2.27}$$

Using eq. 2.27 this (as well as the definition for $\boldsymbol{y}^T \boldsymbol{L}\boldsymbol{y}$ derived in eq. 2.20), we may define

$$\begin{aligned} \boldsymbol{y}^T \boldsymbol{L}\boldsymbol{y} &= \frac{1}{2} \sum_{i \in A} \sum_{j \in B} \left(\frac{1}{V(A)} + \frac{1}{V(B)} \right)^2 w_{ij} \\ &= \frac{1}{2} \left(\frac{1}{V(A)} + \frac{1}{V(B)} \right)^2 \text{cut}(A, B) \end{aligned} \tag{2.28}$$

and

$$\begin{aligned} \boldsymbol{y}^T \boldsymbol{D}\boldsymbol{y} &= \sum_{i \in A} y_i^2 d_i + \sum_{j \in B} y_j^2 d_j \\ &= \frac{1}{V(A)^2} \sum_{i \in A} d_i + \frac{1}{V(B)^2} \sum_{j \in B} d_j \\ &= \frac{1}{V(A)} + \frac{1}{V(B)}. \end{aligned} \tag{2.29}$$

Finally, we get

$$\begin{aligned}\frac{\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}} &= \frac{1}{2}\frac{\left(\frac{1}{V(A)}+\frac{1}{V(B)}\right)^2\text{cut}(A,B)}{\frac{1}{V(A)}+\frac{1}{V(B)}} \\ &= \frac{1}{2}\left(\frac{1}{V(A)}+\frac{1}{V(B)}\right)\text{cut}(A,B) \\ &\propto \text{Ncut}(A,B). \end{aligned} \tag{2.30}$$

The optimisation problem becomes

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}}. \tag{2.31}$$

Allowing any value for $\boldsymbol{y}$ in eq. 2.27 and setting the constraint that

$$\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y} = \boldsymbol{y}_1^T\boldsymbol{D}^{-1/2}\boldsymbol{D}^{-1/2}\boldsymbol{y}_1 = \boldsymbol{z}^T\boldsymbol{z} = 1,$$

we find that eq. 2.31 reduces to a generalised eigenvalue problem

$$\begin{aligned}\boldsymbol{L}\boldsymbol{y} &= \lambda\boldsymbol{D}\boldsymbol{y} \\ \boldsymbol{D}^{-1}\boldsymbol{L}\boldsymbol{y} &= \lambda\boldsymbol{y} \\ \tilde{\boldsymbol{L}}\boldsymbol{y} &= \lambda\boldsymbol{y}, \end{aligned} \tag{2.32}$$

where $\tilde{\boldsymbol{L}}$ is the normalised graph Laplacian from eq. 2.23. As with Laplacian eigenmaps, the eigenvector corresponding to the second smallest eigenvalue, $\boldsymbol{y}_1$, is used to perform the final spectral clustering by dicretising the components of $\boldsymbol{z}$ = according to some fitting threshold value.

Similar to Laplacian eigenmaps, spectral clustering allow graph relations to consider both local and global similarities in a way that allows for a higher degree of variation in the shape of the data distributions, see figs. 2.3 and 2.5. Thus, unexplored and unfamiliar data could potentially benefit from incorporating these relations into the tools used. A disadvantage of this method is its lack of robustness in the presence of noise, see figure 2.6. Due to a randomly generated ”bridge” between the two natural groupings, the resulting partitions are not discovered by the k nearest neighbour affinity graph used in this particular case. Thus, a significant motivation of this thesis — being within the field of graph-based multi-view clustering — builds on the intuition that a unity of graph information and traditional data adds much needed robustness into spectral clustering, as well as much needed relational information into traditional clustering.

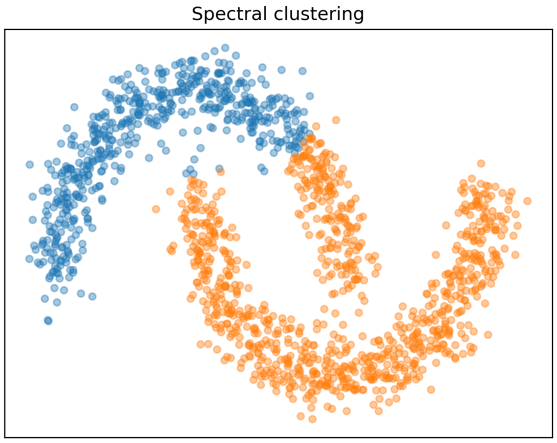


Figure 2.6: Spectral clustering performance may be affected by noisy data distributions.

3

Deep Learning

This chapter will cover a few current deep learning models, as well as a selection of the methods which led us to the state-of-the-art. This includes the multilayer perceptron (MLP), convolutional neural networks (CNN), and the graph neural network (GNN). This will set the stage for newer auto-encoder and deep clustering methods, which will be covered towards the end of the chapter. The categories covered in this chapter set the stage for a wide range of deep learning research, including the model presented in this paper. To begin, let us get familiar with the idea of *data features* by introducing the concept of the simple *data transformation* which will set the foundation for numerous machine learning approaches.

Let $x_1, x_2, ..., x_N$ be a set of data values making up some vector $\boldsymbol{x}$. Given a unitary[1] $N \times N$ matrix $\boldsymbol{A}$, a transformation $\boldsymbol{x} \longrightarrow f(\boldsymbol{x})$ where $\boldsymbol{y} = f(\boldsymbol{x})$ can be defined as [103]

$$\boldsymbol{y} = \boldsymbol{A}^T \boldsymbol{x} \equiv \begin{bmatrix} \boldsymbol{a}_0^H \\ \vdots \\ \boldsymbol{a}_N^H \end{bmatrix} \boldsymbol{x}. \tag{3.1}$$

The vectors $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ are referred to as the *basis vectors* of the transformation f [103].

1. A matrix in the complex domain is unitary iff $\boldsymbol{A}^{-1} = \boldsymbol{A}^H$. This corresponds to an orthogonal matrix for real matrices, i.e. $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$.

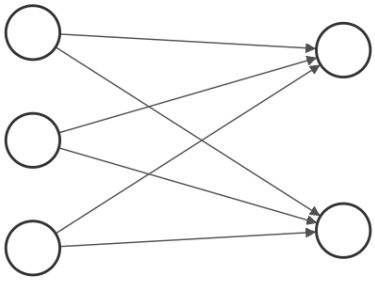


Figure 3.1: A simple perceptron — the building block of a neural network. This perceptron has three input nodes and two output nodes.

3.1 Perceptrons

The original perceptron is a simple classifier which is based on our understanding of a neuron of the brain [86]. This inspired the grew to become the multilayer perceptron, appropriately referred to as a neural network [34]. Although the theory behind neural networks run tens of years back into the previous century, the primary bottleneck of AI development has been the limited computational power [25]. However, due to the exponentially increasing computing power (Moore's law), today's computers are well able to train neural networks that are tens — and even hundreds — of layers deep.

The simple perceptron [86] is a linear transformation of an input vector $\boldsymbol{x}$ to a predicted output $\hat{y}$, as such:

$$\hat{y} = g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0, \tag{3.2}$$

where w_0 is referred to as the *bias* of the function.

In the original paper, the perceptron was used as a binary classifier. Thus, the output $g(\boldsymbol{x})$, is passed through a continuous activation function, $f(\cdot)$, from which the output would be $f(g(\boldsymbol{x})) \in (-1, 1)$. However, in many applications it is desired to have the output $f(g(\boldsymbol{x})) \in (0, 1)$, which may also be achieved by using another activation function.

For supervised machine learning, the perceptron's weights w_j for $j = 0, 1, ..., k$,

can be estimated by an iterative process aimed to minimise a given loss function, $J(\boldsymbol{x}|\boldsymbol{y})$ defined as

$$J = \sum_{i=1}^{N} \varepsilon(i), \tag{3.3}$$

where $\varepsilon(i)$ is the loss value for training pair $(\boldsymbol{x}_i, y_i)$.

The properties of the activation function will often play a role in the quality of the model, as well as its loss value. Two of these important properties are their derivative and output span. The sigmoid activation function is given by

$$\sigma(\hat{y}) = \frac{1}{1 + e^{-\hat{y}}} \tag{3.4}$$

and spans $\sigma \in (0, 1), \quad \hat{y} \in (-\infty, \infty)$.

Another popular activation function is the hyperbolic tangent:

$$\tanh(\hat{y}) = \frac{e^{\hat{y}} - e^{-\hat{y}}}{e^{\hat{y}} + e^{-\hat{y}}} \tag{3.5}$$

which spans $\tanh \in (-1, 1), \quad \hat{y} \in (-\infty, \infty)$.

A simple, yet effective activation function is the Rectified Linear Unit (ReLU) given by:

$$\mathrm{ReLU}(\hat{y}) = \begin{cases} \hat{y} & \hat{y} \geq 0 \\ 0 & \hat{y} < 0 \end{cases} \tag{3.6}$$

having $\mathrm{ReLU} \in (0, 1), \quad \hat{y} \in (-\infty, \infty)$.

An alternative to the ReLU function is the LeakyReLU [65]:

$$\mathrm{LeakyReLU}(\hat{y}; a) = \begin{cases} \hat{y} & \hat{y} \geq 0 \\ a\hat{y} & \hat{y} < 0 \end{cases}, \tag{3.7}$$

using parameter $0 < a < 1$ defining the leaking parameter, yielding a slight gradient leakage for negative values.

If the aim is to produce a probability-like output, the softmax activation function is applicable. In this case, we look at a one-hot label vector for data pairs $(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)$,

Notation :

- $v_j^r(i)$: Layer output for node $j \in \{1, ..., k_r\}$, layer $r \in \{1, ..., L\}$, and sample $i \in \{1, ..., N\}$ pre-activation.
- $a_j^r(i) = f(v_j^r(i))$: Post-activation output for node j in layer r for sample i.
- $\hat{y}_j(i) = f_{out}(v_j^L(i))$: Prediction, post-activation for $v_j^L(i)$.
- $\boldsymbol{w}_j^r$: The weight vector of shape k_{r-1} relating the output of layer $r-1$ to node j in layer r.
- w_{j0}^r : The bias of node j in layer r.

where $\hat{\boldsymbol{y}}_i = (\hat{y}_{i,m}, ..., \hat{y}_{i,k_L})$ are binary values

$$\text{softmax}(\hat{y}_{i,m}) = \frac{e^{\hat{y}_{i,m}}}{\sum_{n=1}^{k_L} e^{\hat{y}_{i,n}}} \tag{3.8}$$

which spans $\text{softmax}(\hat{y}_{i,m}) \in [0, 1]$, $\hat{y}_{i,m} \in (-\infty, \infty), i = 1, ..., k_l, m \in \{1, ..., k_L\}$.

3.1.1 The Multilayer Perceptron

The multilayer perceptron (MLP) [34] takes advantage of the property that an arbitrary number of repeated linear transformations may create any complex hyperplane to be used in classification [103]. The architecture of such a network is split in three, the input layer, the hidden layer(s), and the output layer. Their tasks are, respectively, to form the hyperplanes used in classification, form the class regions by one or multiple repeated transformations, and give neuron weightings that correspond to the output prediction [103]. By increasing the number of hidden layers in the network (thus increasing the depth of a neural network) its potential in recognising complex patterns increases as well. However, the number of parameters to be learned increase as well.

By utilising the transformation of equation 3.2, sequential transformation of the data are applied by the input layer and hidden layers as such

$$v_j^r(i) = \left(\boldsymbol{w}_j^r\right)^T \boldsymbol{a}^{(r-1)} + w_{j0}^r, \tag{3.9}$$

and $\boldsymbol{v}^r(i) = [v_1^r(i), v_2^r(i), ..., v_{k_r}^r(i)]^T$. This is calculated for $r = 1, ..., L$ where $\boldsymbol{v}^L(i) = \hat{\boldsymbol{y}}(i)$ is the output vector.

The final layer transforms the data into the same shape as the labels to which the data corresponds. This is where we apply the loss function (eq. 3.3). An example of such a loss function is the cross-entropy loss

$$\varepsilon(i) = -\sum_{k=1}^{k_L} y_k(i) \ln \hat{y}_k(i), \tag{3.10}$$

where $y_k(i) \in \{0, 1\}$ is the binary value of one-hot vector $\boldsymbol{y}(i)$ for training pair i.

Or in the binary case

$$\varepsilon = -\sum_{k=1}^{k_L} y_k(i) \ln \hat{y}_k(i) + (1 - y_k(i)) \ln (1 - \hat{y}_k(i)), \tag{3.11}$$

where $y_k(i) \in \{0, 1\}$ and $\hat{y}_k(i) \in (0, 1)$ are, respectively, the true labels and the predicted (soft) labels for output node k and training pair i.

In order to iteratively create an optimal MLP model $g : \boldsymbol{x} \rightarrow \hat{\boldsymbol{y}}$, we want to find the weights that minimise this loss function, i.e.

$$J^* = \min_{\boldsymbol{W}} \Big\{ \sum_{i=1}^{N} \varepsilon(i) \Big\}, \tag{3.12}$$

where $\boldsymbol{W}$ is the set of all parameters in MLP g. The the vast majority of loss landscapes has no closed-form solution, as such finding the minimum of a loss function such as eq. 3.12 is done iteratively.

3.1.1.1 Gradient Descent and Backpropagation

The way neural network weights are optimised is through a process called gradient descent (described in [54]). This is an iterative process in which the gradient of the loss function with regards to model parameters is used to fine-tune the parameters of the model.

Let us look at a set of simple perceptrons — one transformation in a MLP, with input dimension k_0 and output dimension k_1 (see fig. 3.1 for reference). For each of the two layers $r \in \{0, 1\}$, we look at nodes $j = 1, ..., k_r$ — the number of variables in the given layer. The transformation has one set of weights $\boldsymbol{w}_j^1$ The process of finding the optimal weight change begins by solving

$$\nabla_{\boldsymbol{w}_j^1} J = \sum_{i=1}^{N} \frac{\partial \varepsilon(i)}{\partial \boldsymbol{w}_j^1}. \tag{3.13}$$

Given that we know this value, the updated weights for layer r (in our case $r = 1$ as this is our only weight vector) can be defined by taking a small step in the correct direction, as such

$$\boldsymbol{w}_j^r(\text{new}) = \boldsymbol{w}_j^r(\text{old}) + \Delta \boldsymbol{w}_j^r \tag{3.14}$$

with

$$\Delta \boldsymbol{w}_j^r = -\mu \cdot \nabla_{\boldsymbol{w}_j^r} J, \tag{3.15}$$

where μ is a step size parameter that determines how fast the model weights will change throughout the iterative process.

The process described for the simple perceptron extends to the MLP. At layer $r \leq L$ we extend eq. 3.13 by applying the chain rule

$$\begin{aligned} \nabla_{\boldsymbol{w}_j^L} J &= \sum_{i=1}^{N} \frac{\partial \varepsilon(i)}{\partial \boldsymbol{w}_j^L} \\ &= \sum_{i=1}^{N} \frac{\partial \varepsilon(i)}{\partial v_j^r(i)} \nabla_{\boldsymbol{w}_j^r} v_j^r(i), \end{aligned} \tag{3.16}$$

where $v_j^r(i)$ is the j'th node output of a layer r in the MLP, as given in eq. 3.2. For each node j this can be written as:

$$v_j^r(i) = \sum_{k=1}^{k_{r-1}} w_{jk}^r a_k^{r-1}(i) + w_{j0}^r, \tag{3.17}$$

where $v_k^r(i)$ is the output of layer r. For $r = L$, $a_k^r(i) = \hat{y}_k(i)$. And for the input layer $r = 1$ we have $v_k^r(i) = x_k(i)$.

Recalling eqs. 3.14 and 3.15, the task now consists of finding the two factors from eq. 3.16, namely $\frac{\partial \varepsilon(i)}{\partial v_j^r(i)}$ and $\nabla_{\boldsymbol{w}_j^r} v_j^r(i)$, for nodes $j = 1, ..., k_r$ and layers $r = 1, ..., L$.

For simplicity of notation, we define

$$\frac{\partial \varepsilon(i)}{\partial v_j^r(i)} \equiv \delta_j^r(i). \tag{3.18}$$

Starting off, by looking at eq. 3.17, $\nabla_{\boldsymbol{w}_j^r} v_j^r(i)$ for $j = 0, ..., k_{r-1}$ can be solved

simply as following [103]

$$\nabla_{\boldsymbol{w}_j^r} v_j^r(i) \equiv \begin{bmatrix} \dfrac{\partial}{\partial w_{j1}^r} v_j^r(i) \\ \vdots \\ \dfrac{\partial}{\partial w_{jk_{r-1}}^r} v_j^r(i) \end{bmatrix} = \boldsymbol{a}^{r-1}(i) \tag{3.19}$$

and for the bias:

$$\frac{\partial}{\partial w_{j0}^r} v_j^r(i) = +1. \tag{3.20}$$

The other factor, $\delta_j^r(i)$, will be dependent on the activation function $f(\cdot)$ and the loss function $\varepsilon(i)$. Using the multi-class cross-entropy loss function from eq. 3.10, the calculation for layer $r = L$ is derived as follows

$$\begin{aligned} \delta_j^L(i) &= \frac{\partial \varepsilon(i)}{\partial v_j^L(i)} \\ &= \frac{\partial}{\partial v_j^L(i)} \left\{ - \sum_{k=1}^{k_L} y_k(i) \ln\left(\hat{y}_k(i)\right) \right\} \\ &= \sum_{k=1}^{k_L} \left[\frac{\partial}{\partial \hat{y}_k(i)} \Big(- y_k(i) \ln\left(\hat{y}_k(i)\right) \Big) \cdot \frac{\partial \hat{y}_k(i)}{\partial v_j^L(i)} \right] \\ &= \sum_{k=1}^{k_L} \left[\left(- \frac{y_k(i)}{\hat{y}_k(i)} \right) \cdot \frac{\partial f(v_k^L(i))}{\partial v_j^L(i)} \right]. \end{aligned} \tag{3.21}$$

In the case of softmax output the derivative is defined as

$$\begin{aligned} \frac{\partial f_s(v_k^L(i))}{\partial v_j^L(i)} &= \frac{\partial}{\partial v_j^L(i)} \left(\frac{e^{v_k^L(i)}}{\sum_{m=1}^{k_L} e^{v_m^L(i)}} \right) \\ &= \frac{\dfrac{\partial e^{v_k^L(i)}}{\partial v_j^L(i)} \cdot \left(\sum_{m=1}^{k_L} e^{v_m^L(i)} \right) - e^{v_k^L(i)} e^{v_j^L(i)}}{\left(\sum_{m=1}^{k_L} e^{v_m^L(i)} \right)^2} \\ &= f_s(v_k^L(i)) \cdot \left(\tilde{\delta}_{kj} - f_s(v_j^L(i)) \right), \end{aligned} \tag{3.22}$$

where

$$\tilde{\delta}_{kj} = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$$

is the Kronecker delta function.

Yielding out final answer in the case of cross-entropy loss and softmax output:

$$\begin{aligned}\delta_j^L(i) &= \sum_{k=1}^{k_L}\left[\left(-\frac{y_k(i)}{\hat{y}_k(i)}\right)\cdot\hat{y}_k(i)\big(\tilde{\delta}_{kj}-\hat{y}_j(i)\big)\right]\\ &= -y_j(i)\big(1-\hat{y}_j(i)\big)+\sum_{k\neq j}y_k(i)\hat{y}_j(i)\\ &= \hat{y}_j(i)\Big(y_j(i)+\sum_{k\neq j}y_k(i)\Big)-y_j(i).\end{aligned}$$

As, in a one-hot vector $\sum_k y_k(i) = y_j(i) + \sum_{k\neq j} y_k(i) = 1$, we get

$$\delta_j^L(i) = \hat{y}_j(i) - y_j(i) \tag{3.23}$$

Now, for the lower layers $r < L1$:

$$\begin{aligned}\delta_j^{r-1}(i) &= \frac{\partial\varepsilon(i)}{\partial v_j^{r-1}(i)}\\ &= \sum_{k=1}^{k_r}\frac{\partial\varepsilon(i)}{\partial v_k^r(i)}\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}\\ &= \sum_{k=1}^{k_r}\delta_k^r(i)\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}\end{aligned} \tag{3.24}$$

The first factor $\delta_k^r(i)$ is given by eq. 3.23. The second factor is found as such, using activation function $f(\cdot)1$

$$\begin{aligned}\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} &= \frac{\partial}{\partial v_j^{r-1}(i)}\left(\sum_{m=1}^{k_{r-1}}w_{km}^r a_m^{r-1}(i)\right)\\ &= \frac{\partial}{\partial v_j^{r-1}(i)}\left(\sum_{m=1}^{k_{r-1}}w_{km}^r f(v_m^{r-1}(i))\right).\end{aligned}$$

By assuming the use of an independent activation function such as ReLU, we get

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = w_{kj}^r f'(v_j^{r-1}(i)), \tag{3.25}$$

which yields our final $\delta_j^{r-1}(i)$1

$$\delta_j^{r-1}(i) = \left(\sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r \right) f'(v_j^{r-1}(i)). \tag{3.26}$$

Using ReLU activation function $f_R(\cdot)$ with derivative $f_R'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$ yields

$$\delta_j^{r-1}(i) = \begin{cases} \left(\sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r \right) & v_j^{r-1}(i) \geq 0 \\ 0 & v_j^{r-1}(i) < 0 \end{cases}. \tag{3.27}$$

This completes the algorithm for the standard MLP. Many of the backpropagation algorithm for other deep learning methods build on ideas borne of the algorithm above, as we will discover throughout the remainder of this chapter.

3.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN or ConvNet) is a neural network designed for spatial or sequential data. There are many examples that fall within this category, being both one- and two-dimensional in nature. A few examples are object detection in images [58] and videos [52], time series data [16], spectograms [21], high- and low-frequency images (e.g. images from medical [117] or remote sensing devices [99]), and sensor data (e.g. PET images [53]). These applications differ in architecture, data augmentation, output format, and in learning approach. The main similarity of the CNN is that its main component for feature extraction is the convolution operation. The high pixel count in the average image make linear operations, such as the ones seen in the standard MLP, not a reliable method of feature extraction due to the high parameter count this would entail. In addition, such a network would in most cases not be translation invariant, i.e. its classification ability would depend on the objects placement in the image. The reason for this is due to the behavior of the convolution operation.

The convolution operation is a mathematical operation that applies a kernel of arbitrary size to weigh an image, see fig. 3.2. The operation can be compared to a weighted mean. However, in convolutions the spatial relations are highly relevant. The formula of the one-dimensional convolution operation is as

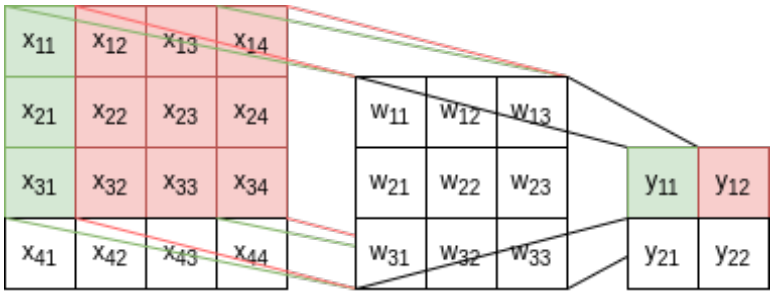


Figure 3.2: Visualisation of the convolution operation on a weight kernel $\boldsymbol{W}$.

follows [25]

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{3.28}$$

In order to apply convolutions to two-dimensional data such as images, the function needs to be extended to two or more dimensions as such [25]

$$(\boldsymbol{I} * \boldsymbol{K})(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n) \tag{3.29}$$

$$(\boldsymbol{I} * \boldsymbol{K})(i,j,...) = \sum_{m,n,...} I(m,n,...)K(i-m, j-n, ...). \tag{3.30}$$

The convolution operation can be extended to an arbitrary number of dimensions, so if volumetric data such as CT images are to be processed by a CNN [97], the operation can be extended to three dimensions.

Each one of these convolutions provide a sole value for a convolution of an image section with an equally dimensioned weight matrix, $M \times N$. Such a weight matrix is slid across the image so that each $M \times N$ tile on the input corresponds to one pixel in the output. Additional parameters such as stride: the number of pixels to jump from one tile to the next (the default 1 will imply that for each convolution operation, the next section is 1 square away), and padding: how many pixels to fill the space surrounding the image and what to fill it with. The default stride value is 1, and padding is usually simply done by inserting 0. As an example, if a 3×3 weight matrix is applied in a convolutional operation with stride 1 and 1 pixel pooling, the output will equal the input in size, as each pixel in the original image is surrounded by at least one pixel in every direction.

A block (referred to as *layer* in [25]) in a CNN typically consists of more than just the convolution operation [25]. Similar to the feature extraction approach in the MLP, CNNs also apply activation functions. Common activation functions such as the ReLU work in a similar fashion, on a pixel-to-pixel basis [25].

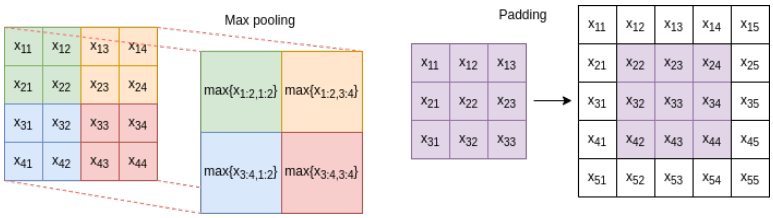


Figure 3.3: Max pooling and padding.

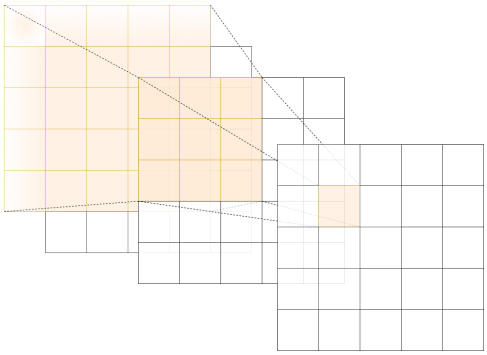


Figure 3.4: The increasing receptive field of a CNN with a 3×3 convolution operator.

Thirdly, a block consists of a *pooling layer*. The objective of the pooling layer is to compress the feature representation, as well as help in making them more translation invariant [25]. A common selection for pooling is *maximum pooling* [125]. In maximum pooling, a pooling window of size $M \times N$ "slides" across the image in a similar fashion to the convolution operation and outputs the maximum feature value in the tile. See fig. 3.3. Many recent CNN methods employ the *skip connection* introduced in the ResNet architecture [35] in which, the network performs residual learning, i.e. an arbitrary CNN operation $f(\cdot)$ producing output $f(\boldsymbol{x})$ will pass on the learned feature map by adding to an identity mapping of $\boldsymbol{x}$, as such

$$\boldsymbol{x}^{r+1} = f(\boldsymbol{x}^r) + \boldsymbol{x}^r. \tag{3.31}$$

This change allowed deep learning research to create far deeper and better neural networks. Reasoned by the fact that any arbitrarily deep network should be able to perform at least as well as a similar but shallower network, if simply by having the additional layers solely performing identity mappings (i.e., the case where additional layers produce $f(\boldsymbol{x}^r) = 0$). A core point of repeated applications of CNN blocks is the concept of increasing the network's *receptive field*. Compressing information through convolution and pooling operations will allow for information sharing across the full image space. See fig. 3.4. Ideally, this should allow the final output to process an image in its entirety.

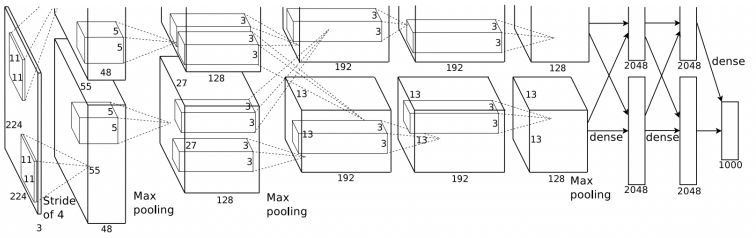


Figure 3.5: AlexNet architecture (figure taken from [58]).

Figure 3.6: Segmentation of a dash cam image.

A simple and common example of a CNN is the AlexNet architecture [58] (fig. 3.5) achieving at the time state-of-the-art (SOTA) accuracy on the ImageNet dataset [19] for image classification. Another application of a CNN is for image segmentation tasks. Tasks in which the goal is to locate various classes in an image and return a masking reflecting the discovery, see fig. 3.6. An example of such a network is U-net [85]. The architectures of CNNs can be applied to various tasks by simply adjusting the output layer. In an image segmentation task, one may set the output dimensions to be equal to the spatial dimensions of the input and compute loss on the difference to an annotated image that has been defined by annotators.

3.2.1 Backpropagation

Backpropagation for CNNs is based on the same ideas seen in backpropagation for MLPs. The idea is to find $\nabla_{\boldsymbol{W}_j^L} J = \sum_{i=1}^{N} \frac{\partial \varepsilon(i)}{\partial \boldsymbol{W}_j^L}$, similar to eq. 3.16, and iteratively improve the weight kernels by attempting to reduce the loss function by gradient descent. From the chain-rule, we have

$$\frac{\partial \varepsilon(i)}{\partial \boldsymbol{W}_j^r} = \left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}_j^r(i)} \right)^T \frac{\partial \boldsymbol{A}_j^r(i)}{\partial \boldsymbol{W}_j^r}. \tag{3.32}$$

Notation :

- $\boldsymbol{V}_j^r(i)$: Feature map $j \in \{1, ..., k_r\}$ of shape (H_{k_r}, W_{k_r}) in layer $r \in \{1, ..., L\}$, for sample $i \in \{1, ..., N\}$, pre-activation.
- $\boldsymbol{A}_j^r(i) = f(\boldsymbol{V}_j^r(i))$: Feature map j in layer r for sample i, post-activation.
- $\hat{y}_j(i) = f(v_j^L(i))$: Linear prediction for output node j from node $v_j^L(i)$ in a flattened layer $\boldsymbol{v}^L(i)$.
- $\hat{\boldsymbol{Y}}(i) = f_{out}(\boldsymbol{V}^L(i))$: Image prediction of shape (C, H_{k_L}, W_{k_L}) for from the final layer convolution output $\boldsymbol{V}^L(i)$.
- $\boldsymbol{W}_j^r$: The weight kernel of shape (k_{r-1}, H_W, W_W) relating the output from layer $r-1$ to feature map j in layer r.
- $w_{j,i,m,n}^r$: The weight value in (m, n) going from feature map i in layer $r-1$ to feature map j in layer r.
- b_j^r : The bias associated with kernel matrix $\boldsymbol{W}_j^r$ for feature map j in layer r.

For backpropagation further down the layers, we also want to derive

$$\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}_s^{r-1}(i)} \equiv \boldsymbol{\delta}_j^{r-1}(i) = \left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}^r(i)}\right)^T \frac{\partial \boldsymbol{A}^r(i)}{\partial \boldsymbol{A}_s^{r-1}(i)}. \tag{3.33}$$

Same as for MLPs, an easy way to solve this problem is to bisect the equation and solve one factor at a time. Looking to the convolution step illustrated in fig. 3.2 and taking into account eq. 3.29 and the components in 3.3, we see that an output pixel $v_{j,p,q}^r(i)$ is defined by a weighted sum with $\boldsymbol{W}_j^r$ over its corresponding tile in the previous layer $\boldsymbol{a}_{:,p',q'}^{r-1}(i)$. The convolution for output $a_{1,1}^r$ stems from a convolution of tile $\boldsymbol{v}_{1:3,1:3}$ (1 as index for padded pixels) with $\boldsymbol{W}^r$. For odd values of H_W and W_W we define $H_{off} = (H_W - 1)/2$ and $W_{off} = (W_W - 1)/2$. And for even values we have $H_{off} = H_W/2$ and $W_{off} = W_W/2$. An output value is then found to be

$$a_{i,p,q}^r(i) = f\Big(\sum_{l,m,n}\Big[w_{j,l,m,n}^r a_{l,p'-H_{off}+m,q'-W_{off}+n)}^{r-1}\Big] + b_j^r\Big) \tag{3.34}$$

In the case of equally sized input and output, i.e. $(H_{k_r}, W_{k_r}) = (H_{k_{r-1}}, W_{k_{r-1}})$, we have $(p, q) = (p', q')$.

Now, to derive the two terms in eq. 3.32. The bias is — as with the MLP — easy to derive from eq. 3.34

$$\frac{\partial \boldsymbol{A}_j^r(i)}{\partial b_j^r} = f'\big(\boldsymbol{V}_j^r(i)\big). \tag{3.35}$$

Now for the weights.

$$\begin{aligned}\frac{\partial a^r_{j,p,q}(i)}{\partial w^r_{j,s,a,b}} &= f'\big(v^r_{j,p,q}(i)\big)\frac{\partial v^r_{j,p,q}(i)}{\partial w^r_{j,s,a,b}} \\ &= f'\big(v^r_{j,p,q}(i)\big)\frac{\partial}{\partial w^r_{j,a,b}}\big((\boldsymbol{W}^r_j * \boldsymbol{A}^{r-1}(i))(p,q) + b^r_j\big).\end{aligned} \tag{3.36}$$

Using the definition in eq. 3.30

$$\begin{aligned}\frac{\partial a^r_{j,p,q}(i)}{\partial w^r_{j,s,a,b}} &= f'\big(v^r_{j,p,q}(i)\big)\sum_{l,m,n}\frac{\partial}{\partial w^r_{j,s,a,b}}\big(w^r_{j,l,m,n}a^{r-1}_{l,p-m,q-n}(i)\big) \\ &= f'\big(v^r_{j,p,q}(i)\big)a^{r-1}_{s,p-a,q-b}(i),\end{aligned} \tag{3.37}$$

yielding

$$\frac{\partial \varepsilon(i)}{\partial w^r_{j,s,a,b}} = \left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}^r_j(i)}\right)^T \begin{bmatrix} f'\big(v^r_{j,1,1}(i)\big)a^{r-1}_{s,1-a,1-b}(i) \\ \vdots \\ f'\big(v^r_{j,H_{k_{r-1}},W_{k_{r-1}}}(i)\big)a^{r-1}_{s,H_{k_{r-1}}-a,W_{k_{r-1}}-b}(i)\end{bmatrix} \tag{3.38}$$

for all corresponding elements in $\boldsymbol{A}$.

Now for the second part. We have, in eq. 3.33, the factor that is passed on throughout the neural network. This can be found by changing the variable being derivated with regards to in factor $\frac{\partial v^r_{j,p,q}(i)}{\partial w^r_{j,s,a,b}}$ in eq. 3.36 to $\frac{\partial v^r_{j,p,q}(i)}{\partial a^r_{s,a,b}}$ and summing over j, getting

$$\begin{aligned}\frac{\partial \varepsilon(i)}{\partial a^{r-1}_{s,a,b}} &= \sum_{j=1}^{k_r}\left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}^r_j(i)}\right)^T\frac{\partial \boldsymbol{A}^r_j(i)}{\partial \boldsymbol{A}^{r-1}_s(i)} \\ &= \sum_{j=1}^{k_r}\left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}^r_j(i)}\right)^T\begin{bmatrix} f'\big(v^r_{j,1,1}(i)\big)w^{r-1}_{j,s,1-a,1-b}(i) \\ \vdots \\ f'\big(v^r_{j,H_{k_{r-1}},W_{k_{r-1}}}(i)\big)w^{r-1}_{j,s,H_{k_{r-1}}-a,W_{k_{r-1}}-b}(i)\end{bmatrix}.\end{aligned} \tag{3.39}$$

For classification tasks, the feature map is flattened during forward pass into a one-dimensional feature vector following feature extraction from the convolutional layers. In this case, loss is calculated as in the MLP case and reshaped during backpropagation. Thus, we end up with matrices $\boldsymbol{\delta}^r_j(i)$ to be used throughout the initial convolutional layers. However, in the case of having spatial outputs, this is not the case. We assume that this is the case (e.g. a

segmentation task). Our label is given by $\boldsymbol{Y}(i) \in \{0,1\}$ in the shape (C,H,W), the same as the soft predictions $\hat{\boldsymbol{Y}}(i) \in [0,1]$. $\boldsymbol{\delta}_j^L(i)$ should have the shape (H_{k_L}, W_{k_L}). Using (element-wise) sigmoid activation $f_{out}(\cdot)$ (eq. 3.4) and binary cross-entropy loss (eq. 3.11) in the final layer $r = L$, we derive

$$\begin{aligned}\frac{\partial \varepsilon(i)}{\partial \boldsymbol{V}_j^L(i)} &= \frac{\partial}{\partial \boldsymbol{V}_j^L(i)}\Big\{ -\boldsymbol{Y}_j(i)\ln\hat{\boldsymbol{Y}}_j(i) + \big(1-\boldsymbol{Y}_j(i)\big)\ln\big(1-\hat{\boldsymbol{Y}}_j(i)\big)\Big\} \\ &= \frac{\partial}{\partial \boldsymbol{V}_j^L(i)}\Big(-\boldsymbol{Y}_j(i)\ln f_{out}(\boldsymbol{V}_j^L(i))\Big) \\ &+ \frac{\partial}{\partial \boldsymbol{V}_j^L(i)}\Big(\big(1-\boldsymbol{Y}_j(i)\big)\ln\big(1-f_{out}(\boldsymbol{V}_j^L(i))\big)\Big).\end{aligned}$$

Knowing that the sigmoid function is independent of other $v^L_{j,\cdot,\cdot}(i)$, we may define the elements of the matrix as such

$$\frac{\partial \varepsilon(i)}{\partial v^L_{j,p,q}(i)} = \left[-\frac{y_{j,p,q}(i)}{f_{out}(v^L_{j,p,q}(i))} + \frac{1-y_{j,p,q}(i)}{1-f_{out}(v^L_{j,p,q}(i))}\right] f'_{out}(v^L_{j,p,q}(i)). \tag{3.40}$$

The derivative of the sigmoid is given as

$$\begin{aligned}\sigma(x) &= \frac{1}{1+e^{-x}} \\ \frac{d\sigma(x)}{dx} &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} \\ &= \frac{1}{1+e^{-x}}\frac{(1+e^{-x})-1}{1+e^{-x}} \\ &= \sigma(x)(1-\sigma(x)).\end{aligned} \tag{3.41}$$

Then, finalising eq. 3.40:

$$\begin{aligned}\frac{\partial \varepsilon(i)}{\partial v^L_{j,p,q}(i)} &= \left[-\frac{y_{j,p,q}(i)}{f_{out}(v^L_{j,p,q}(i))} + \frac{1-y_{j,p,q}(i)}{1-f_{out}(v^L_{j,p,q}(i))}\right] f_{out}(v^L_{j,p,q}(i))(1-f_{out}(v^L_{j,p,q}(i))) \\ &= (f_{out}(v^L_{j,p,q}(i))-1)y_{j,p,q}(i) + f_{out}(v^L_{j,p,q}(i))(1-y_{j,p,q}(i)) \\ &= f_{out}(v^L_{j,p,q}(i)) - y_{j,p,q}(i)\end{aligned}$$

or, equally

$$\frac{\partial \varepsilon(i)}{\partial \boldsymbol{V}^L(i)} = f_{out}\big(\boldsymbol{V}^L(i)\big) - \boldsymbol{Y}(i). \tag{3.42}$$

Deriving $\boldsymbol{\delta}_s^{L-1}(i)$ by the same logic as in eq. 3.39

$$\boldsymbol{\delta}_s^{L-1}(i) = \sum_{j=1}^{k_L}\left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{V}_j^L(i)}\right)^T \frac{\partial \boldsymbol{V}_j^L(i)}{\partial \boldsymbol{A}_s^{L-1}(i)} = \sum_{j=1}^{k_L}\left(\frac{\partial \varepsilon(i)}{\partial \boldsymbol{V}_j^L(i)}\right)^T \begin{bmatrix} w^{r-1}_{j,s,1-a,1-b}(i) \\ \vdots \\ w^{r-1}_{j,s,H_{k_{r-1}}-a,W_{k_{r-1}}-b}(i)\end{bmatrix}. \tag{3.43}$$

Now, for the other layers $r < L$:

$$\begin{aligned}\boldsymbol{\delta}_s^{r-1}(i) &= \frac{\partial \varepsilon(i)}{\partial \boldsymbol{A}_s^{r-1}(i)} \\ &= \left(\boldsymbol{\delta}^r(i)\right)^T \frac{\partial \boldsymbol{A}^r}{\partial \boldsymbol{A}_s^{r-1}},\end{aligned} \tag{3.44}$$

which is derived and propagated for all layers, continuously updating weights and biases with eqs. 3.38 and 3.35.

3.3 Graph Neural Networks

Thus far, we have tackled some standard machine learning networks for vectorial data, as well as multi-dimensional data in both the spatial and the temporal domain. In this section, we will cover the deep learning approach for a centrepiece data structure of this paper: the graph. Graphs are unlike images and temporal data due to similarities not being recognised with respect to their position in the matrix, and unlike vectorial data as each vector does not represent a set of objective measures, but rather a subjective view from each separate data point — thus requiring the entire graph to be taken in context to understand any broader pattern that may be present.

Graph Neural Networks (GNNs) are ordinarily utilised for three task categories [124]:

1. *Node-level* tasks
2. *Edge-level* tasks
3. *Graph-level* tasks

These tasks help decide how the loss function should be set up (and as such, the ML method's objective) [124]. Node-level tasks will learn to recognise nodes in, e.g., classification or clustering tasks. Edge-level tasks are often used for classification and edge prediction tasks. Finally, graph-level tasks may be used for graph learning / graph matching, and classification. Similar to previous methods, GNNs span every supervision setting, from fully supervised to unsupervised, as well as everything in between. A selection of these semi-supervised graph methods will be covered in section 3.5 concerning semi-supervised deep learning.

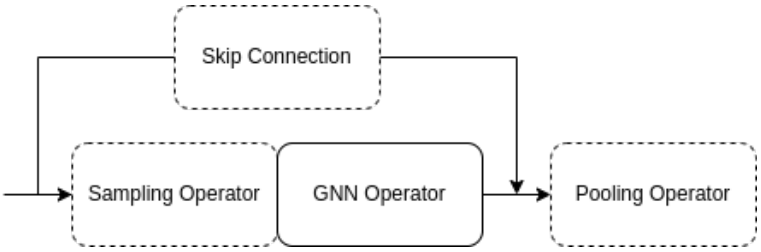


Figure 3.7: One example of a GNN layer [124].

The components of a standard GNN layer is similar to that of the CNN. Skip connections may be used used to solve the issues of vanishing gradients and allowing for far deeper neural networks, e.g. [61]. Also similar to CNNs, pooling operators may be employed to extract and compress information. Due to the poor scaling of graphs, sampling may in many cases be required for large datasets in order to avoid memory issues. However, due to the larger patterns in graphs (as discussed in section 2.4.1), sampling a graph is challenging to avoid resulting in a significant loss of contextual information. Thus, GNN sampling operators can be used used in an attempt to maximise the propagated information through sampling. Using batches (with $BS < N$) GraphSAINT [122] has been shown to increase performance compared to using the complete graph in certain cases [122].

One sampling method is found in GraphSAGE [30], which samples a fixed number of neighbours for each node, an approach which aptly may be referred to as *node sampling* [124]. *Layer sampling* is another approach to graph sampling that is applied layer-wise, sampling the graph's receptive field[2] directly[3] [124]. FastGCN [11] employs this sampling method, utilising an importance sampling technique making important nodes more likely to be sampled. Finally, *subgraph sampling* may be used to sample an entire subgraph at a time, rather than nodes or edges as was the case with the two latter sampling methods. This process is utilised in GraphSAINT [122] and ClusterGCN [14]. The former of which will be covered later in this section.

There are two main approaches one may utilise in GNN feature extraction: one is *spectral* in nature and the other is *spatial* [124]. In this paper, we will

2. Note: The receptive field of a graph is — unlike that of an image — connected by its edges rather than its spatial affinity (which is the case in the example fig. 3.4).
3. Random nodes are selected in each layer, increasing the receptive field for each layer (the GNN will converge to include all vertices with a non-zero probability as the number of layers tends to infinity). This may be understood as sampling the graph's receptive field directly.

mainly focus on spectral approaches. These approaches are characterised by applying operations in the spectral domain. At the core of this methods is the *graph convolution* operator.

3.3.1 The Graph Convolution

Firstly, a singular graph signal $\boldsymbol{x} \in \mathbb{R}^N$ is transformed to the spectral domain by the *Fourier transform* $\mathcal{F}$, as such [124]

$$\mathcal{F}(\boldsymbol{x}) = \boldsymbol{U}^T\boldsymbol{x}. \tag{3.45}$$

And its inverse is given as

$$\mathcal{F}^{-1}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{x}. \tag{3.46}$$

Here, $\boldsymbol{U}$ is defined as the matrix of eigenvectors associated with the normalised graph Laplacian $\tilde{\boldsymbol{L}}$ (from eq. 2.23). Convolution with a filter $\boldsymbol{g}$ may thus be achieved by performing element-wise multiplication in the spectral domain[4] as such

$$\begin{aligned}
\boldsymbol{g} * \boldsymbol{x} &= \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{g}) \odot \mathcal{F}(\boldsymbol{x})\big) \\
&= \boldsymbol{U}\big(\boldsymbol{U}^T\boldsymbol{g} \odot \boldsymbol{U}^T\boldsymbol{x}\big) && (3.47)\\
&= \boldsymbol{U}\boldsymbol{G}\boldsymbol{U}^T\boldsymbol{x}, && (3.48)
\end{aligned}$$

where the arbitrary filter $\boldsymbol{g}$ in eq. 3.47 is simplified to $\boldsymbol{G}$, a learnable diagonal matrix, in eq. 3.48, which concludes the standard graph convolution function. Hammond et al. [31] suggested that $\boldsymbol{G}$ could be approximated by use of K-th order Chebyshev polynomials. The GCN [57] was later suggested using $K = 1$, approximating eq. 3.48 by

$$\begin{aligned}
\boldsymbol{g} * \boldsymbol{x} &\approx \sum_{k=0}^{K} w_k \boldsymbol{T}_k(\tilde{\boldsymbol{L}} - \boldsymbol{I}_N)\boldsymbol{x} \\
&= w_0\boldsymbol{x} + w_1(\tilde{\boldsymbol{L}} - \boldsymbol{I}_N)\boldsymbol{x} \\
&= w_0\boldsymbol{x} - w_1\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{x}, && (3.49)
\end{aligned}$$

where $\boldsymbol{T}_0(\boldsymbol{x}) \equiv 1$ and $\boldsymbol{T}_1(\boldsymbol{x}) \equiv \boldsymbol{x}$, and $\boldsymbol{A}$ is the affinity matrix. Setting $w = w_0 = -w_1$, equation 3.49 is further reduced to

$$\boldsymbol{g} * \boldsymbol{x} \approx w\big(\boldsymbol{I}_N + \boldsymbol{D}^{-1}\boldsymbol{A}\big)\boldsymbol{x} \tag{3.50}$$

or equivalently

$$\boldsymbol{g} * \boldsymbol{x} \approx w\big(\boldsymbol{I}_N + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\big)\boldsymbol{x}. \tag{3.51}$$

4. Due to the convolution theorem [70].

In order to counteract the gradient problem, the *renormalisation trick* is applied to eq. 3.51 as such

$$\boldsymbol{I}_N + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \rightarrow \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}, \tag{3.52}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ make up the corresponding diagonal degree matrix. Finally, for a complete input matrix $\boldsymbol{X} \in \mathbb{R}^{N \times p}$ at an arbitrary layer in a GNN, we transform the data in a fashion familiar to previous deep learning methods by applying a learned weight matrix $\boldsymbol{W} \in \mathbb{R}^{p \times p'}$ transforming each data point to $\mathbb{R}^{p'}$:

$$\boldsymbol{H} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{W}. \tag{3.53}$$

The GCN has been used as a baseline spectral approach in a range of GNNs, inspiring developments such as the Dual Graph Convolutional Network (DGCN) [127] and Adaptive Graph Convolutional Network (AGCN) [62]. The former of which is trained to combine data to incorporate both local and global consistencies into the network. AGCN is a model aiming to recognise tacit relations between nodes, while not necessarily being linked by an edge.

We will briefly cover the spatial approach to GNNs for context. Rather than applying convolution operations in spectral domain, spatial approaches to GNNs apply convolutions directly on the graph topology. A significant contribution within this category is GraphSAGE [30] (formerly referred to for its method of sampling). The extraction process is solved through feature aggregation of neighbourhoods of sampled nodes. The method of aggregation may vary in type, namely *mean aggregator*, *LSTM* (which requires a predetermined order of nodes), and *pooling aggregator*. Providing an adequate solution to the sampling problem, this method is a simplistic and reliable GNN for many common applications. Other approaches within the spatial category is the Graph Attention network GAT [107]. GAT is an attention-based method, learning to recognise the importance of neighbours for each node using *self-attention*[5]. Due to its ability to neglect certain uninformative nodes, this approach can be useful in cases where edges may be noise-prone or diffuse.

3.3.2 Graph Sampling

Graph SAmpling based INductive learning meThod (GraphSAINT) is a sampling algorithm attempting to solve the aforementioned scaling issue of GNNs [122].

5. Learning to recognise each node's similarity by understanding their significance in regards to the other nodes in their respective neighbourhoods.

GraphSAINT aims to directly sample the graph, rather than the opposing node and layer-wise sampling. The subgraphs should retain maximal information as well as select complementary batches building well-learned representations for the whole graph. In addition, normalisation should be performed to eliminate sampling biases and variance [122]. Assuming a GCN of shape

$$\boldsymbol{X}^{(l)} = \boldsymbol{A}\boldsymbol{X}^{(l-1)}\boldsymbol{W}^{(l)}$$

or equally

$$\boldsymbol{x}_v^{(l)} = \sum_{u \in \mathcal{V}} A_{v,u}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)},$$

Zeng et al. solves the bias and variance problem by finding the unbiased estimator of the aggregated feature of nodes $v \in \mathcal{V}$

$$\zeta_v^{(l)} = \sum_{u \in \mathcal{V}} \frac{A_{v,u}}{\alpha_{u,v}}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}\mathbb{1}_{u|v}, \tag{3.54}$$

with normalisation constant $\alpha_{u,v}$ and, for each subgraph of nodes $\mathcal{V}_s$ and edges E_s, we have

$$\mathbb{1}_{u|v} = \begin{cases} 0 & v \in \mathcal{V}_s \wedge (u,v) \notin E_s \\ 1 & (u,v) \in E_s \end{cases}.$$

Now, $p_{v,u} = p_{u,v}$ is defined as the probability that edge $(u,v) \in E$ is sampled, and p_v the probability of node $v \in \mathcal{V}$ being sampled. Equation 3.54 may be shown to be unbiased for an arbitrary subgraph sampling method if $\alpha_{u,v} = \dfrac{p_{u,v}}{p_v}$ when v is sampled in subgraph s as such:

$$\begin{aligned}
\mathbb{E}\left[\zeta_v^{(l)}\right] &= \mathbb{E}\Big[\sum_{u \in \mathcal{V}} \frac{A_{v,u}}{\alpha_{u,v}}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}\mathbb{1}_{u|v}\Big] \\
&= \sum_{u \in \mathcal{V}} \frac{A_{v,u}}{\alpha_{u,v}}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}\mathbb{E}[\mathbb{1}_{u|v}] \\
&= \sum_{u \in \mathcal{V}} \frac{A_{v,u}}{\alpha_{u,v}}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}P\big((u,v)|v\big),
\end{aligned}$$

using Bayes' theorem, we may rewrite $P\big((u,v)|v\big) = \dfrac{P\big(v|(u,v)\big)P\big((u,v)\big)}{P\big(v\big)}$, where $P\big(v|(u,v)\big) = 1$ by the condition that v is sampled in a subgraph. Thus, we get

$$\begin{aligned}
\mathbb{E}\left[\zeta_v^{(l)}\right] &= \sum_{u \in \mathcal{V}} \frac{A_{v,u}}{\alpha_{u,v}}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}\frac{p_{u,v}}{p_v} \\
&= \sum_{u \in \mathcal{V}} A_{v,u}\boldsymbol{x}_u^{(l-1)}\boldsymbol{W}^{(l)}.
\end{aligned} \tag{3.55}$$

Finally, loss normalisation is achieved by dividing node loss by normalisation value $\lambda_v = \left|\mathcal{V}\right| \cdot p_v$ so that the expectation of the batch loss

$$L_{batch} = \sum_{v \in G_s} \frac{L_v}{\lambda_v}$$

has the unbiased expectation

$$\mathbb{E}\left[L_{batch}\right] = \frac{1}{\left|\mathcal{V}\right|} \sum_{v \in \mathcal{V}} L_v.$$

Zeng et al. proves that the edge probabilities $p_{u,v}$ minimises the sum of variances of each dimension

$$\zeta = \sum_{l} \sum_{v \in G_s} \frac{\zeta_v^{(l)}}{p_v}$$

when the edge probabilities are given as

$$p_{u,v} = \frac{m}{\sum_{(u,v)'} \left\| \sum_l \boldsymbol{b}_{(u,v)'}^{(l)} \right\|} \left\| \sum_l \boldsymbol{b}_{(u,v)}^{(l)} \right\|, \tag{3.56}$$

with $\boldsymbol{b}_{(u,v)}^{(l)} = A_{v,u} \boldsymbol{x}_u^{(l-1)} \boldsymbol{W}^{(l)} + A_{u,v} \boldsymbol{x}_v^{(l-1)} \boldsymbol{W}^{(l)}$ and some budget parameter m. Avoiding the issue of requiring $\boldsymbol{x}_u^{(l-1)}$ for sampling, the edge probabilities are in practice estimated by

$$p_{(u,v)} \propto A_{v,u} + A_{u,v}. \tag{3.57}$$

One GraphSAINT algorithm is the random walk-based sampler. The form of the Random walk is arbitrary, but one solution is having r root samples randomly drawn, where each node takes h steps drawn from edge probabilities. Now, viewing the graph as a Markov Chain, one may get the edge weights of L layers by $\boldsymbol{B} = \boldsymbol{A}^L$ with elements $B_{i,j}$. The interpretation of $\boldsymbol{B}$ is a value proportional to the probability of going from i to j in L steps — viewing values of $\boldsymbol{A}$ as proportional to the transition probabilities. Thus, eq. 3.57 sets the edge probabilities by $p_{(u,v)} \propto B_{u,v} + B_{v,u}$ and is interpreted as the mean probability of a random walk ending up at node v starting at node u, and vice versa, after L steps.

3.4 Autoencoders & Deep Clustering

Autoencoders are (for the most part) unsupervised neural networks trained to reconstruct its input from an encoded representation or an *embedding*

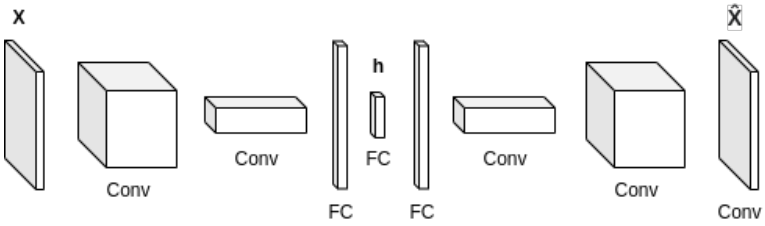


Figure 3.8: An example of an autoencoder consisting of fully connected (FC) layers and convolutional (conv) layers.

$\boldsymbol{h} = f(\boldsymbol{x})$ [25]. Deep clustering networks often take advantage of this encoder architecture by performing clustering in the embedded space. Autoencoders or more specifically its components — encoders and decoders — are often useful as tools for visualising high dimensional data, tools of compression, or as a concrete approximation of a latent space [25]. The latter is useful in e.g. generative adversarial networks (GANs) in which the application is that of generate new data by randomly drawing from the embedding space, and applying a decoder to reconstruct the latent representation into a fresh data point. Considering its usefulness and applicability in clustering, this section will provide a sufficient introduction into autoencoders, as well as one SOTA model used in our final multi-view clustering application.

3.4.1 Encoder-Decoder Networks

The standard autoencoder consists of two components: the *encoder* $f(\cdot)$ and the *decoder* $g(\cdot)$ [25]. The first of which is designed to create a latent representation of the data, $\boldsymbol{h}$, while the latter attempts to reconstruct the input image from the encoded representation, $\hat{\boldsymbol{y}} = g(f(\boldsymbol{x}))$. Commonly, the latent representation will be of lower dimensionality than the input. This is referred to as an *undercomplete autoencoder*[6]. Thus, the model is forced to learn to prioritise the aspects of the data when training. In other words: the model is forced to specialise to the data at hand. This is a useful property for clustering, as the hidden space will represent a condensed representation of the most central aspects of a data point.

Any arbitrary data type can be modelled to fit an autoencoder, e.g. images as in fig. 3.8. Mathematically, the loss function of an autoencoder is given as such [25]

$$\mathcal{L}\big(\boldsymbol{x}, g(f(\boldsymbol{x})); \boldsymbol{\theta}, \boldsymbol{\phi}\big), \tag{3.58}$$

6. It is an *overcomplete autoencoder* when the opposite is true. Commonly, these fail to learn important properties of the data as there is no need for data reduction (i.e. selection) [25].

having encoder and decoder parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. Any suitable loss function may be applied in order to calculate the difference between the original and reconstructed data, e.g. cross-entropy loss.

In certain cases[7] encoder and decoder parameters are defined to be equal, i.e. $\boldsymbol{\phi} = \boldsymbol{\theta}$. This is referred to as *tied weights*. In these cases linear transformations in the encoder is given as

$$\boldsymbol{Y}^{(r)} = \boldsymbol{W}^{(r)}\boldsymbol{Y}^{(r-1)}$$

while its corresponding decoder layer is simply given with transposed weights, as such

$$\hat{\boldsymbol{Y}}^{(r-1)} = \left(\boldsymbol{W}^{(r)}\right)^T \hat{\boldsymbol{Y}}^{(r)}. \tag{3.59}$$

Autoencoders may be constructed to fit its use case to a greater degree by altering certain properties. For example, denoising autoencoders can be trained to improve robustness, as well as eliminate noise from input by simply augmenting the second loss term in eq. 3.58 as such $g(f(\tilde{\boldsymbol{x}}))$ where $\tilde{\boldsymbol{x}}$ is the input image with added noise [25]. Other variants include sparse autoencoders, where a regulariser term $\Omega(\boldsymbol{h})$ is added to the loss term 3.58. These autoencoders can be used for interpretability or simply as an analysis tool [25]. A third use case is to learn a statistical hidden representation. These autoencoders are often referred to as *variational autoencoders* (VAEs). These models attempt to create a model that encodes and decodes data to and from a statistically meaningful hidden space. This is done by minimising the loss function divergence between the model's latent distribution and a desired latent distribution [25]. As previously mentioned, AEs may be applied as tools for clustering, this is done in Deep Embedded Clustering (DEC) [112] and Improved Deep Embedded Clustering (IDEC) [29] models. The former train an AE in order to use its encoder for deep embedded clustering. After extracting the encoder it is finetuned by optimising a divergence based clustering objective through self-training [112]. IDEC is an improvement on DEC by jointly training both the autoencoder as well as solely the encoder by utilising a reconstruction loss and clustering loss, respectively [29].

3.4.2 Deep Divergence-based Clustering

The clustering module to be used in experiments is the Deep Divergence-based Clustering (DDC) module. This will be the building block in the SOTA multi-view clustering models, and is therefore deemed central to the contents of this

7. e.g. when one wishes to reduce the number of learnable parameters of the model.

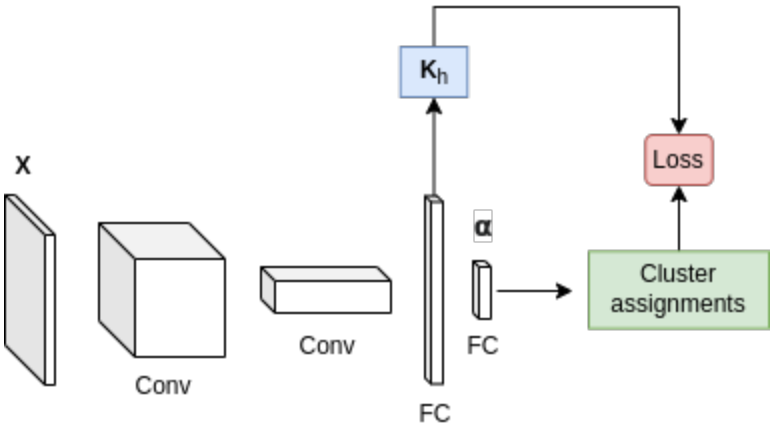


Figure 3.9: The DDC architecture for image data [50].

paper. It is also important to note that a thorough understanding of DDC will help in analysis of our experiments later on. The DDC module is based on an objective to obtain good feature representations that retain three desired properties of high-quality clustering: (i) dense (intra-cluster distance), (ii) separated (inter-cluster distance), and (iii) unambiguous (strict assignments). The model consists of an encoder which outputs cluster assignment predictions from which the loss is calculated and backpropagated to optimize the encoder (see fig. 3.9).

The loss function of the DDC module consists of three loss terms with each its separate objective. Firstly, the CS divergence (see section 2.1.2) is employed to assert the dissimilarity between distributions. In order to use the CS divergence in a clustering problem, one would first have to quantise the distributions. This can be done by applying kernel methods and in this case by using a Parzen window estimator [87] with bandwidth parameter σ. Taking the unweighted l_2 norm from eq. 2.6 we may calculate the similarity matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ with elements

$$k_{\boldsymbol{x}_i, \boldsymbol{x}_j} = \exp\left\{-d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)^2/(2\sigma^2)\right\}. \tag{3.60}$$

Using this estimate, the discretised CS divergence becomes [46]

$$D_{cs} = -\log\left(\frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q\in C_i}\sum_{l\in C_j}k_{q,l}}{\sqrt{\sum_{q,q'\in C_i}k_{q,q'}\sum_{l,l'\in C_j}k_{l,l'}}}\right). \tag{3.61}$$

Using cluster assignments $\hat{Y} = [\alpha_{q,i}] \in \mathbb{R}^{n \times k}$ denoting hard cluster memberships $\alpha_{q,i} \in \{0, 1\}$ of data point q to cluster C_i, we may rewrite eq. 3.61 as

$D_{cs} = -\log d_\alpha$ where [50]

$$d_\alpha = \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q,l=1}^{N}\alpha_{q,i}\alpha_{l,j}k_{q,l}}{\sqrt{\sum_{q,l=1}^{N}\alpha_{q,i}\alpha_{l,i}k_{q,l}\sum_{q,l=1}^{N}\alpha_{q,j}\alpha_{l,j}k_{q,l}}}, \tag{3.62}$$

given that $\sum_{q\in C_i}\sum_{l\in C_j}k_{q,l} = \sum_{q,l=1}^{N}\alpha_{q,i}\alpha_{l,j}k_{q,l}$. This corresponds to the first loss term in the DDC module: $\mathcal{L}_1 = d_\alpha$.

Now, in order to avoid trivial solutions and collapsing clusters (corresponding to the third objective), the second loss term will ensure diversity in cluster assignments [50]. This is achieved through the outer product of the cluster assignment matrix $\hat{\boldsymbol{Y}}$, $\hat{\boldsymbol{Y}}\hat{\boldsymbol{Y}}^T$. Now, for clear cluster assignments and a balanced prediction, the upper triangular element of $\hat{\boldsymbol{Y}}\hat{\boldsymbol{Y}}^T$ — denoted $\text{triu}(\hat{\boldsymbol{Y}}\hat{\boldsymbol{Y}}^T)$ and is interpreted as a sum of the upper triangle — will ensure that not all clusters winds up in the same cluster by favouring orthogonal cluster assignments. Thus, we have our second loss term [50]

$$\mathcal{L}_2 = \text{triu}(\hat{\boldsymbol{Y}}\hat{\boldsymbol{Y}}^T) = \sum_{i=1}^{N}\sum_{j=i+1}^{N}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_j. \tag{3.63}$$

Finally, our third loss term will remove the trivial solutions arising from the fact that orthogonal cluster assignments are not necessarily located along the simplex for clear classification [50]. Thus, the third loss term will push points towards the simplex $\mathbb{R}^k$, promoting clear cluster assignments. Given $\boldsymbol{e}_i \in \mathbb{R}^k$ being the ith corner of the simplex, the function

$$m_{q,i} = \exp\{-||\boldsymbol{\alpha}_q - \boldsymbol{e}_i||^2\} \tag{3.64}$$

is defined as a similarity measure that draws cluster assignments towards the simplex. This measure is implemented in the CS divergence as before, taking the place of $\alpha_{q,i}$ as such

$$\mathcal{L}_3 = \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q,l=1}^{N}m_{q,i}m_{l,j}k_{q,l}}{\sqrt{\sum_{q,l=1}^{N}m_{q,i}m_{l,i}k_{q,l}\sum_{q,l=1}^{N}m_{q,j}m_{l,j}k_{q,l}}}. \tag{3.65}$$

Resulting in the final loss function of the DDC module

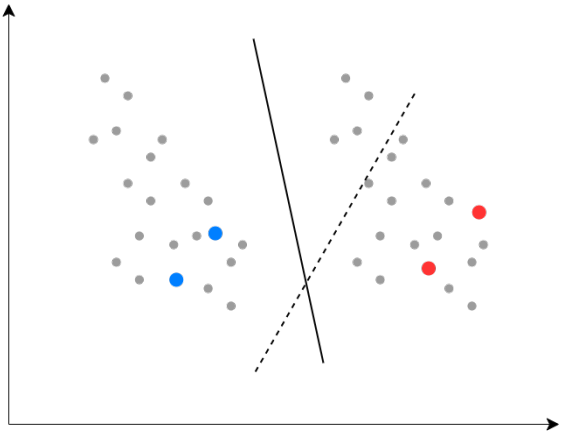
$$\begin{aligned}\mathcal{L}_{tot} &= \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \\ &= \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q,l=1}^{N}\alpha_{q,i}\alpha_{l,j}k_{q,l}}{\sqrt{\sum_{q,l=1}^{N}\alpha_{q,i}\alpha_{l,i}k_{q,l}\sum_{q,l=1}^{N}\alpha_{q,j}\alpha_{l,j}k_{q,l}}} \\ &+ \sum_{i=1}^{N}\sum_{j=i+1}^{N}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_j + \frac{1}{k}\sum_{i=1}^{k-1}\sum_{j>i}\frac{\sum_{q,l=1}^{N}m_{q,i}m_{l,j}k_{q,l}}{\sqrt{\sum_{q,l=1}^{N}m_{q,i}m_{l,i}k_{q,l}\sum_{q,l=1}^{N}m_{q,j}m_{l,j}k_{q,l}}}.\end{aligned} \tag{3.66}$$

By optimising this function by help of gradient descent, the model will optimise its ability to draw clusters from the input data.

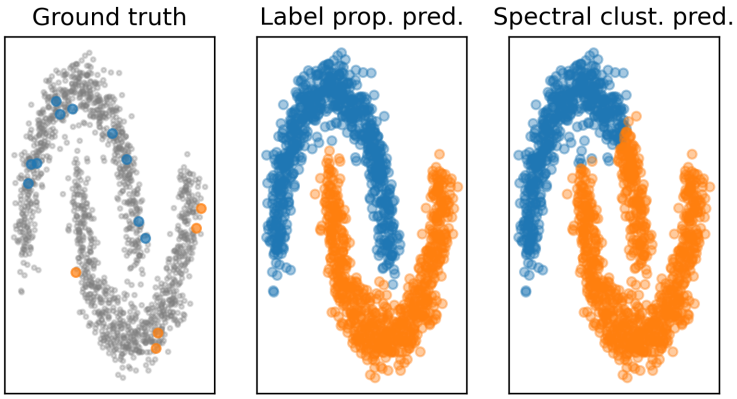
3.5 Semi-supervised Learning

Thus far we have covered both supervised and unsupervised machine learning models. Their usefulness in machine learning applications are evident. Whether the goal is to maximise model performance in a resource rich environment or its antipode: modelling entirely fresh data with no available fitting labels. The final section covering deep learning, will cover the field of semi-supervised learning (SSL). Being a unification of the two previous tasks, SSL is applied in cases where labels are available for a subset of the complete dataset. Its usefulness is the highest in applications where labels are costly to obtain and the subject matter sufficiently complex. Similarly to fully supervised and unsupervised approaches, semi-supervised machine learning range in complexity from closed-form-solution, basic machine learning methods to deep neural network approaches.

The basic methods may be categorised based on separation method and the nature of the data. *Low-density separation based*, *disagreement-based*, *graph-based*, and *self-training* are four categories of semi-supervised machine learning that will build upon the general understanding of how partially labeled data may help (and potentially harm) learning. To begin, the Transductive Support Vector Machine (TSVM) [106] is a semi-supervised, low-density separation based approach to the traditional Support Vector Machine (SVM) [15] which is a powerful linear classifier. Illustrated in figure 3.10a, the TSVM uses unlabeled data to better recognise the low-density areas that best fit a decision boundary. Disagreement-based methods is used to generate agreement by comparing different predictors on data from a single dataset. One common method within this category — co-training — will be covered in chapter 4. Within the cate-



(a) The TSVM approach of SSL. A decision boundary using only the labeled data is illustrated with a dashed line.



(b) Label propagation. A graph-based approach to SSL.

Figure 3.10: Illustrations of how unlabeled data (grey) may assist in finding an optimal decision boundary.

gory of graph-based method, one core method is label propagation [126]. Using graph-based methods in an unsupervised manner allows for similar advantages to those of spectral clustering and Laplacian eigenmaps (in section 2.4), while providing additional useful information as compared to the fully unsupervised spectral clustering — see comparison between label propagation and spectral clustering in fig. 3.10b. Self-training is the final basic category of SSL in our selection. Self-training [22] originated from the concept of labelling unlabeled data. This can be utilised by training a supervised model on limited labeled data, outputting label predictions. A set of these predictions, known as pseudo-labels, were assigned as true labels and reiterated by the supervised model, now with a larger set of feature-label pairs. The criterion of selection for the pseudo-labels is in principle arbitrary, but common practice is to select the most confident predictions [3], generally by setting some threshold or by selecting the n most confident predictions.

3.5.1 Deep Semi-supervised Learning

Extending into the deep learning domain, SSL builds upon the base principles (or categorisations) using better performing networks: graph-based methods such as GraphSAGE [30] and GCN [57], mentioned in the section on GNNs, are designed to be used in semi-supervised machine learning [115]. Methods such as co-training and self-training also extends to the deep learning domain, as they are *model agnostic*[8] in nature. Put simply, SSL may be defined as to minimise the following optimisation problem [115]:

$$\sum_{\boldsymbol{x}\in\boldsymbol{X}_L,\, y\in\boldsymbol{y}_L} \mathcal{L}_s(\boldsymbol{x}, y, \boldsymbol{\theta}) + \alpha \sum_{\boldsymbol{x}\in\boldsymbol{X}_U} \mathcal{L}_u(\boldsymbol{x}, \boldsymbol{\theta}) + \beta \sum_{\boldsymbol{x}\in\boldsymbol{X}} \mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{3.67}$$

having labels $\boldsymbol{y}_L$, input data corresponding to labeled and unlabeled data $\boldsymbol{X} = \{\boldsymbol{X}_L, \boldsymbol{X}_U\}$, and hyperparameters $\boldsymbol{\theta}$. The three terms makes up the supervised loss, unsupervised loss, and regularisation, respectively.

8. The method does not depend on any specific machine learning model.

4

Multi-view Learning

4.1 The Basic Approach

This portion of the thesis will cover basic multi-view methods. These will serve as base knowledge and provide the reader with a fundamental understanding of the ideas that make up multi-view learning as well as its many applications in the health sector.

The medical domain is one that is very attractive to the field of multi-view learning [123; 9]. This is in part due to the information contained in data from a range of different modalities. Of these include image data from medical imaging such as X-rays, time series data from medical time series such as ECG and oxygen saturation levels, as well as a range of static data such as laboratory tests or categorical data such as prior diagnoses and prescriptions. The existing methods of multi-view learning can be separated into three reigning categories [113]: co-training, multiple kernel learning, and subspace learning. These are the categories that will be covered in this chapter, acting as context for the deep-learning based multi-view method to be used in our experiments.

Multi-view learning in general consists of two principles: the *consensus principle* and the *complementary principle* [113]. The former aims to ensure agreement across views. Given that there are two hypotheses for two different views, (x_i^1, x_i^2, y_i) where y_i is the associated label, the principle can be expressed as

the inequality [17]

$$P(f^1 \neq f^2) \geq \max\left\{P_{err}(f^1), P_{err}(f^2)\right\}, \tag{4.1}$$

where f_i is the hypothesis from view i. This implies that the error rate of either hypothesis must be less or equal to the corresponding probability of disagreement [113]. In simpler terms, the maximum probability of any one view yielding an inaccurate prediction cannot be higher than the probability of the two views disagreeing. This states that the two views improve when taken in context of the other.

The second principle, being the complementary principle, states that each view may contain knowledge that is not present in the other views [113]. This is an intuitive interpretation as we are used to different observations corresponding to different information. As an example, let us look at any arbitrary oral statement. Three modalities may for example be the sentence in itself, how they say it, and their body language. These modalities may completely change the perception of what is being said by providing additional information and context. Looking at a more closely related case, multi-view clustering may be aptly used to combine information from the many modalities present in health data.

The trivial approach to multi-view learning is to create a concatenated vector containing all views. However, this approach is cause to over-fitting on small training samples and will not provide a meaningful context to the classifier due to each view containing unique statistical properties not accurately represented in this format [113].

4.1.1 Co-training

Co-training style algorithms are mostly used for semi-supervised machine learning problems, which was covered in chapter 3. These methods work by using predicted data from a given view to provide labels for other uncertain views. The consensus principle is thus achieved by alternately maximising the mutual agreement on different views of unlabeled data [113]. The general idea of co-training has been extended to other multi-view methods such as co-testing [72] and co-clustering [114].

The process of co-training begins with setting up classifiers for each view separately. The goal of the classifier is to ensure that the trainers are correlated in their predictions. Thus, any disagreement in prediction should be used to

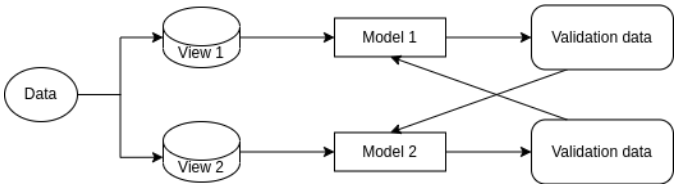


Figure 4.1: The general procedure of co-training [116].

develop a consensus among the trainers [113]. The unlabeled data should be used as a validation set (*knowledge* in fig. 4.1) where predictions from a view is used as training samples in the other view(s) (see fig. 4.1).

Yang and Gao's [114] co-clustering approach utilises information theory to adapt the methods in standard co-training to domain adaptation, where information is transferred across domains by employing transfer learning[1] methods [123].

4.1.2 Multiple Kernel Learning

This section will cover multiple kernel learning (MKL), however, prior to the multi-view variant, basic kernel methods should be covered sufficiently. Kernel methods is an expansive subject and cannot be covered in full in this section.

The use of kernel functions is often based on kernel trick [103]. The kernel trick is a method of extracting more information by adding an implicit mapping to the data [103]. Through this mapping the desire is to create a feature space that is more linearly separable along the hyperplane in which the original data lay. A good kernel function to use as an example are Radial Basis Functions (RBF) given by [103]:

$$f(||\boldsymbol{x} - \boldsymbol{c}_i||) \tag{4.2}$$

or equally

$$K(\boldsymbol{x}, \boldsymbol{c}_i). \tag{4.3}$$

These functions, centred around a point $\boldsymbol{c}_i$ can take a range of forms, e.g.

1. A machine learning field aiming to improve machine learning by transferring knowledge across different source domains [128].

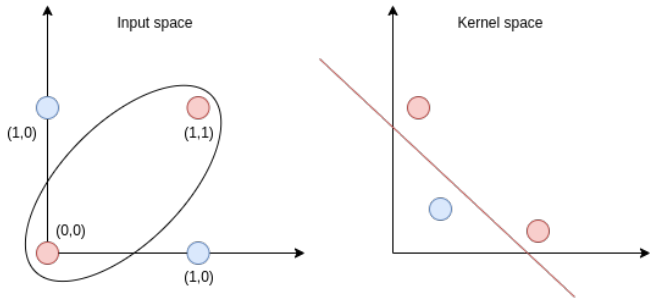


Figure 4.2: Displaying the decision curve for the XOR task — a standard non-linear classification problem where the classes are located along the diagonals ($C_1 = \{(0,0),(1,1)\}$ and $C_2 = \{(0,1),(1,0)\}$) — in input space (left) and the transformed kernel space (right) [103].

[103]

$$f(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{c}_i) = \exp\left(-\frac{1}{2\sigma_i^2}||\boldsymbol{x} - \boldsymbol{c}_i||^2\right) \tag{4.4}$$

$$f(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{c}_i) = \frac{\sigma^2}{\sigma^2 + ||\boldsymbol{x} - \boldsymbol{c}_i||^2} \tag{4.5}$$

of which the Gaussian variety (eq. 4.4) is the most commonly used [103]. The aim of these types of functions is to transform variable $\boldsymbol{x}$ into a transformed space $\boldsymbol{y} = f(\boldsymbol{x})$ in which a classifier can linearly separate classes [103].

A kernel function given by $K(\cdot,\cdot)$ associated with the implicit mapping $\boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x})$ is the building block of such kernel methods. $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ will yield a measure of the effect of point $\boldsymbol{x}_j$ on point $\boldsymbol{x}_i$. Thus, we may use the effect of all points in a dataset $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$ to decide the altered weighting in the mapped space. An example of such mapping is seen in Parzen window estimates [87; 103]. A decision line made in kernel space can be visualised as shown in figure 4.2.

The multiple kernel approach used in multi-view learning is based on the idea that instead of selecting a single kernel function to transform data in different domains, a superior solution would be to use a set of kernels and allow the classifier to decide which kernel function to use for each variety [113]. The data is split by view and transformed into a set of kernels and combined into a unified kernel representing the multi-view data element [113]. A number of combination methods may be used to combine the set of kernels, e.g. a *linear*

combination method [113] as such

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{M} d_k K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{M} d_k f_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{4.6}$$

where d_k is the weighting of kernel K_k on the unified kernel. This kernel is referred to as a *composite kernel*.

Having transformed the multi-view data into the unified kernel space, a high-grade linear classifier such as a Support Vector Machine (SVM) can be used to classify a MKL problem, such as in *SILP* [100] or the *simple MKL* [83]. Unsupervised approaches using multi-kernel learning in the natural sciences has also been suggestet by e.g. Mariette and Villa-Vialaneix [69] on multi-omics[2] data. Other MKL algorithms worth mentioning are within the *group-LASSO* approach [120]. The group-LASSO approach is an extension of the LASSO regression model[3] which aims to regularise the set of kernels as to create a more sparse representation. Considering the kernels as groups, these methods provide a way of analysing the relevance of various views [113].

4.1.3 Subspace Learning

The final category of classical multi-view learning algorithms is subspace learning. These methods aim to obtain a shared latent subspace from which it is assumed that all the input views were generated from [113]. One advantage of using this latent subspace is to reduce the dimensionality of the input data to lessen the effect of the *curse of dimensionality*[4] [113]. Having obtained this unified subspace, the ensuing analysis tool should be straight-forward by applying standard classification or clustering methods [113].

These methods are often used in collusion with dimensionality reduction methods based on eigendecomposition. An example of this is the approach of canonical correlation analysis (CCA) [38] or its kernel variant KCCA [2]. These methods obtain the unified subspace by applying a basis by which the correlation between projections is maximised. Another approach within subspace learning is that of multi-view metric learning [113]. For these learning algorithms, the aim is to produce a unified embedding by utilising meaningful distance measures for every view [113]. One such approach is that of Quadrianto

2. Analysis on data from multiple ”omes”, e.g. genomics, proteomics, and methylomics.
3. A regression model which employs a regularisation term l_1 to create sparser weights.
4. A classification problem that arises in high dimensions as the density of data points decreases and prediction boundaries become less clear-cut.

and Lampert [81] which is based on the straightforward idea to pull similar samples together while pushing dissimilar samples apart [113].

4.2 Multi-view Clustering

4.2.1 Simple Multi-view Clustering

This chapter in its entirety builds on the paper covering Simple- and Contrastive Multi-view Clustering (SiMVC and CoMVC, respectively) models by Trosten et al. [104]. The idea of this method builds on the understanding that aligning the views in an embedded feature space create a view-invariant feature representation. This will have the added benefit of being able to disregard information that only exists in certain views, while extracting the common information. Given an assumption of the complementary principle[5] covered in chapter 4, we understand that the model should not learn to disregard views instantly. This leads to the idea that aligning the representation will force the model to find patterns by having it try to employ all views. This will contain the drawback of making it harder to prioritise views in representation space. Another drawback of multi-view cluster alignment is the issue of not having an equal number of separable clusters in each view, and thus making it harder to separate in the combined representation space. A third drawback in cluster alignment comes from the problem of mis-aligning the innate "ground truth" distributions as a result of different view representations aligning with each other, creating a combined representation space that misrepresents the underlying natural groupings. To avoid these common drawbacks of cluster alignments, Trosten et al. propose the contrastive variant of their deep multi-view clustering (MVC) algorithm by using contrastive learning to align data points at sample-level.

Using what we know of MLPs and CNNs and encoder networks, feature encoders are created for each view separately by appropriately selecting neural networks that — for each sample view $\boldsymbol{x}_i^{(v)}$ — create an output vector $\boldsymbol{z}_i^{(v)}$ of a predetermined common shape for all views $v = 1, ..., V$ with V total views. This is, for each view v, given by the equation

$$\boldsymbol{z}_i^{(v)} = f^{(v)}(\boldsymbol{x}_i^{(v)}). \tag{4.7}$$

5. Views may contain knowledge not present in other views.

Then the unified representation is given by a weighted average

$$\boldsymbol{z}_i = \sum_{v=1}^{V} w_v \boldsymbol{z}_i^{(v)}, \tag{4.8}$$

where w_v is the fusion weight for view $v \in [1, V]$. The weights must sum to one, i.e. $\sum_{v=1}^{V} w_v = 1$, which is achieved by utilising the softmax activation function.

Our clustering module of choice, the DDC module, is utilised on this fused representation $\boldsymbol{z}_i$. Having our soft cluster assignments α_i, we recall the three loss components from DDC 3:

$$\mathcal{L}_1 = \binom{k}{2}^{-1} \sum_{i=1}^{k-1} \sum_{j>i} \frac{\boldsymbol{\alpha}_i^T \boldsymbol{K} \boldsymbol{\alpha}_j}{\sqrt{\boldsymbol{\alpha}_i^T \boldsymbol{K} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j^T \boldsymbol{K} \boldsymbol{\alpha}_j}} \tag{4.9}$$

$$\mathcal{L}_2 = \binom{n}{2}^{-1} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_j \tag{4.10}$$

$$\mathcal{L}_3 = \binom{k}{2}^{-1} \sum_{i=1}^{k-1} \sum_{j>i} \frac{\boldsymbol{m}_i^T \boldsymbol{K}_{hid} \boldsymbol{m}_j}{\sqrt{\boldsymbol{m}_i^T \boldsymbol{K} \boldsymbol{m}_i \boldsymbol{m}_j^T \boldsymbol{K} \boldsymbol{m}_j}} \tag{4.11}$$

having k is the number of clusters, elements of kernel $\boldsymbol{K}$ being $\kappa_{ij} = \exp\{-||\boldsymbol{h}_i - \boldsymbol{h}_j||^2/(2\sigma^2)\}$ where $\boldsymbol{h}_i$ are embedded data points, σ is a hyperparameter, and elements of $\boldsymbol{m}_i$: $m_{ij} = \exp\{-||\boldsymbol{\alpha}_i - \hat{\boldsymbol{e}}_j||^2\}$ in which $\hat{\boldsymbol{e}}_j$ is the unit vector of corner j of the $\mathbb{R}^k$ standard simplex.

Finally, the loss function minimised by SiMVC is

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3. \tag{4.12}$$

4.2.2 Contrastive Multi-view Clustering

For the contrastive variant of the model, a self-supervised contrastive loss term is added. This term will attempt to pull points from the same sample towards each other across views, while pushing points from separate samples apart. These are referred to as *positive* and *negative pairs*, respectively. The contrastive module does not depend on the clustering assignments like the loss terms used in SiMVC, but will be added on top of the hidden representation, see fig. 4.3. The contrastive module itself is based on that of Chen et al. [12].

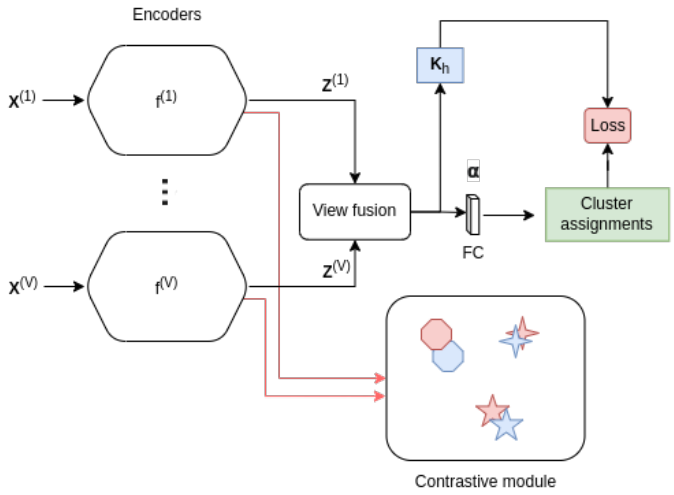


Figure 4.3: The CoMVC model [104].

The similarity between samples i and j in views v and u, respectively, will be measured by the cosine similarity function as such [12; 104]:

$$s_{ij}^{(vu)} = \frac{\left(z_i^{(v)}\right)^T z_j^{(u)}}{||z_i^{(v)}|| \cdot ||z_j^{(u)}||}. \tag{4.13}$$

The loss function is given as a generalised version of NT-Xent loss proposed in [12], and is given as

$$\mathcal{L}_{contrast} = \frac{1}{nV(V-1)} \sum_{i=1}^{n} \sum_{v=1}^{V} \sum_{u=1}^{V} \mathbb{1}_{\{u \neq v\}} l_i^{(uv)}, \tag{4.14}$$

where $\mathbb{1}_{\{u \neq v\}} = \begin{cases} 1 & u \neq v \\ 0 & else \end{cases}$ and

$$l_i^{(uv)} = -\log \frac{e^{s_{ii}^{(uv)}/\tau}}{\sum_{s' \in \text{Neg}\left(z_i^{(v)}, z_i^{(u)}\right)} e^{s'/\tau}}, \tag{4.15}$$

where τ is a hyperparameter[6] and $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ is the set of negative pairs corresponding to the positive pair counterpart $z_i^{(v)}$ and $z_i^{(u)}$. The hyperparameter τ decides the polarisation of the contrastive loss term, i.e. how strong the contrastive force should be on close points versus distant points. The term in

6. τ is set to 0.1 for all experiments.

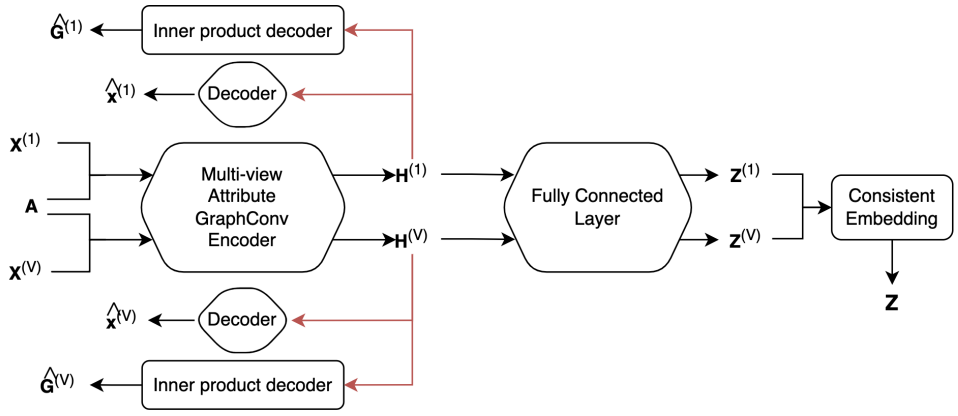


Figure 4.4: The MAGCN model architecture [13].

eq. 4.15 will have the undesirable effect of separating the group structures of which we are after. Thus the set of negative samples $\mathcal{N}_i$ is contructed in such a way that only samples from different cluster predictions than that of i are used in calculating the contrastive loss:

$$\mathcal{N}_i = \left\{ s_{ij}^{uv} : j = 1, ..., N, j \neq i, u, v = 1, ..., V, \arg\max \alpha_i \neq \arg\max \alpha_j \right\}. \tag{4.16}$$

Given a weighting for the contrastive loss, δ, as well as the fusion weights, $w_1, ..., w_V$, from eq. 4.8, we create the differentiating term for CoMVC by scaling the weighted contrastive loss with the minimum fusion weight. This is done to ensure that its importance is equal to that of the least important view. Finally, we obtain the final loss function for CoMVC given by

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \delta \cdot \min\{w_1, ..., w_V\} \mathcal{L}_{contrast}. \tag{4.17}$$

This model is set up to tackle the mixing and weighting of data from various domains, while attempting to avoid the disadvantages of opposing alignments by different views. Due to these properties the model is in this thesis hypothesised to be suitable to cluster medical data, where information of underlying characteristics such as diagnoses or risk analysis are important to assess based on multiple domains considered in context.

4.2.3 Attribute Graph Convolution Clustering

Multi-view Attribute Graph Convolution Networks for Clustering (MAGCN) [13] is a graph-based approach to MVC, combining information from both feature data and some common predetermined affinity matrix. By employing

an attention-based graph embedding to each view, the aim is to reduce both noise and redundancy from views, making it easier to recognise the common information subspace in the *consistent embedding encoders* [13]. The architecture consists of a complete autoencoder — with both graph and view reconstruction, then feeding the encoded graph embeddings into some consistent embedding encoder, yielding our resulting clustering space. Shown in figure 4.4. The model utilises an attention mechanism with parameters shared among nodes, thus allowing for learned attention that does not require a priori knowledge of the dataset size and yielding a model that is better suited for inference and generalisation.

4.2.3.1 Multi-view Attribute Graph Convolution Autoencoder

Let $\boldsymbol{A}$ be our affinity matrix from our graph. The graph-attention mechanism is utilised in the relevance matrix $\boldsymbol{S}$, containing two learnable parameter vectors $\boldsymbol{t}_s^{(r)} \in \mathbb{R}^{d_r}$ and $\boldsymbol{t}_r^{(r)} \in \mathbb{R}^{d_r}$ that weighs nodes and their neighbours in the graph. As mentioned, parameters are shared among nodes such that each base node corresponds to a singular attention value $t_s^r(i)$. Similarly, each receiving node has an attention $t_s^r(j)$. The base node and receiving node attentions correspond to the rows and columns of the affinity matrix, respectively. The relevance matrix, inspired by that of GATE [91], is found by summing row and column embeddings[7], respectively. The two embeddings are found by masking the affinity matrix with the attentions in an element-wise multiplication in the vertical and horizontal directions, respectively. Explicitly stated as

$$\boldsymbol{S}_v^{(r)} = \varphi\left(\boldsymbol{A} \odot \left(\boldsymbol{H}_v^{(r-1)}\boldsymbol{W}^{(r)}\boldsymbol{t}_s^{(r)}\right)^T + \boldsymbol{A} \odot \left(\boldsymbol{H}_v^{(r-1)}\boldsymbol{W}^{(r)}\boldsymbol{t}_r^{(r)}\right)\right). \tag{4.18}$$

The non-linearity φ is set as the sigmoid activation function (eq. 3.4). To get the final relevance coefficient matrix $\boldsymbol{A}_v^{(r)}$, the relevance matrix is normalised in a softmax-like fashion as such

$$A_v^{(r)}(i,j) = \frac{\exp\left(S_v^{(r)}(i,j)\right)}{\sum_{k \in N_i} \exp\left(S_v^{(r)}(i,k)\right)}, \tag{4.19}$$

where $A_v^{(r)}(i,j)$ and $S_v^{(r)}(i,j)$ denotes the (i,j) index of matrices $\boldsymbol{A}_v^{(r)}$ and $\boldsymbol{S}_v^{(r)}$, respectively, and N_i is the set of all nodes in node i's neighbourhood. This will assert that attentions are pushed to either its full or minimum weights relative to the sum. This forces the model to attempt to learn the most relevant data

7. Equation 4.18 is a correction to the original paper, where the equation provided is found to contradict both what is done in their code [5], as well as lacking logical meaning (summing two different parameter vectors in an identical context) nor being solvable with its matrix dimensions not corresponding.

points for all nodes by restricting the total attention.

Letting $\boldsymbol{H}_v^{(0)} \equiv \boldsymbol{X}_v$. The multi-view attribute graph convolution encoder is — for each view $v = 1, ..., V$ — given as the familiar graph convolution used in the GCN with common weights across views, i.e.

$$\boldsymbol{H}_v^{(r)} = \sigma\Big(\Big(\tilde{\boldsymbol{D}}_v^{(r)}\Big)^{-1/2} \tilde{\boldsymbol{A}}_v^{(r)} \Big(\tilde{\boldsymbol{D}}_v^{(r)}\Big)^{-1/2} \boldsymbol{H}_v^{(r-1)} \boldsymbol{W}^{(r)} \Big). \tag{4.20}$$

$\tilde{\boldsymbol{A}}_v^{(r)}$ is our *relevance coefficient matrix* with added self-connections, using the renormalisation trick from the GCN (eq. 3.52) where $\tilde{\boldsymbol{D}}_v^{(r)}$ is its associated degree matrix. The non-linearity σ is set to be the ReLU activation function (eq. 3.6).

Following L encoder layers yielding $\boldsymbol{H}_v \equiv \boldsymbol{H}_v^L$, $\boldsymbol{X}_v$ is reconstructed by using L decoder layers, acting as the inverse of the preceding encoding process, and using tied weights for feature reconstruction.

$$\hat{\boldsymbol{H}}_v^{(r-1)} = \sigma\Big(\Big(\tilde{\hat{\boldsymbol{D}}}_v^{(r)}\Big)^{-1/2} \tilde{\hat{\boldsymbol{A}}}_v^{(r)} \Big(\tilde{\hat{\boldsymbol{D}}}_v^{(r)}\Big)^{-1/2} \hat{\boldsymbol{H}}_v^{(r)} \hat{\boldsymbol{W}}^{(r)} \Big), \tag{4.21}$$

yielding feature reconstructions $\hat{\boldsymbol{X}}_v \equiv \hat{\boldsymbol{H}}_v^{(0)}$.

The reconstructed graphs $\hat{\boldsymbol{A}}_v$ are computed from the inner products, denoted $\langle \cdot, \cdot \rangle$, of row vectors in embedding $\boldsymbol{H}_v$:

$$\hat{A}_v^{ij} = \big\langle -\boldsymbol{h}_v^i, \boldsymbol{h}_v^j \big\rangle. \tag{4.22}$$

Completing the autoencoder section, a reconstruction loss is calculated as such:

$$\mathcal{L}_{re} = \sum_{i=1}^{V} ||\boldsymbol{X}_i - \hat{\boldsymbol{X}}_i||_F^2 + \lambda_1 \sum_{i=1}^{V} ||\boldsymbol{A} - \hat{\boldsymbol{A}}_i||_F^2, \tag{4.23}$$

where $|| \cdot ||_F^2$ denotes the squared Frobenius norm (eq. 2.7).

4.2.3.2 Consistent Embedding Encoder

Having graph embeddings for views $v = 1, ..., V$, $\boldsymbol{H}_v$, the goal becomes to find a unified clustering space $\boldsymbol{Z}$. To solve this task, consistent embedding encoders are employed. Each embedding is transformed to a lower-dimensional clustering space by a MLP layer g having parameters $\boldsymbol{\theta}$

$$g_v(\boldsymbol{H}_v; \boldsymbol{\theta}) \rightarrow \boldsymbol{Z}_v. \tag{4.24}$$

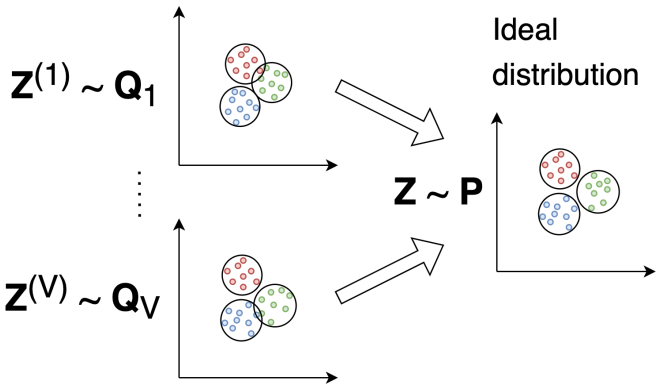


Figure 4.5: MAGCN clustering embedding [13].

The consistent embedding encoder takes two loss functions, one concerns the *geometric relationships*, $\mathcal{L}_{geo}$, and one the *probabilistic relationships*, $\mathcal{L}_{prob}$. The former aims to minimise the Frobenius norm of all embedding differences

$$\mathcal{L}_{geo} = \sum_{i \neq j}^{V} ||Z_i - Z_j||_F^2. \tag{4.25}$$

The embeddings Z_v are combined into Z through *adaptive fusion*

$$Z = \sum_{v=1}^{V} \beta_v Z_v. \tag{4.26}$$

A Student's t-distribution is fit for each data point, i, to measure the similarity to centroids, j, which are randomly initialised in a k-means clustering fashion. With α being the degree of freedom in the t-distribution, the probability of assigning node i to cluster j is given by

$$q_{ij} = \frac{(1 + ||\boldsymbol{z}_i - \boldsymbol{\mu}_j||^2/\alpha)^{-(\alpha+1)/2}}{\sum_{j'}(1 + ||\boldsymbol{z}_i - \boldsymbol{\mu}_{j'}||^2/\alpha)^{-(\alpha+1)/2}}. \tag{4.27}$$

An ideal probability distribution $\boldsymbol{P}$ of $\boldsymbol{Z}$ is calculated by squaring these possibilities and normalising by the per-class frequency, i.e.[8]

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \tag{4.28}$$

8. Correcting the equation in the original paper where f_i is used in the numerator, rather than f_j.

where $f_j = \sum_i q_{ij}$ are the fuzzy cluster frequencies. In effect, eq. 4.28 will help encourage separated cluster distributions. Finally, by again using the Frobenius norm the third loss function is given as

$$\mathcal{L}_{prob} = \sum_{i=1}^{V} \rho_i ||\boldsymbol{Q}_m - \boldsymbol{P}||_F^2. \tag{4.29}$$

ρ_v is trade-off parameters to alter the loss impact of certain view distributions.

Finalising the model, MAGCN is trained by minimising the sum of terms 4.23, 4.25, and 4.29:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{re} + \lambda_2 \mathcal{L}_{geo} + \lambda_3 \mathcal{L}_{prob} && (4.30) \\ =& \sum_{i=1}^{V} ||\boldsymbol{X}_i - \hat{\boldsymbol{X}}_i||_F^2 + \lambda_1 \sum_{i=1}^{V} ||\boldsymbol{A} - \hat{\boldsymbol{A}}_i||_F^2 && (4.31) \\ &+ \lambda_2 \sum_{i \neq j}^{V} ||\boldsymbol{Z}_i - \boldsymbol{Z}_j||_F^2 + \lambda_3 \sum_{i=1}^{V} \rho_i ||\boldsymbol{Q}_m - \boldsymbol{P}||_F^2, \end{aligned}$$

and the final, hard cluster label of node i is simply defined as

$$y_i = \arg\max_k \left(p_{ik}\right). \tag{4.32}$$

4.2.3.3 Discussion

The geometric and probabilistic loss terms in eqs. 4.25 & 4.29 are losses that align the averages of data points across views and the corresponding distributions in embedded space. These do not consider distance-based similarities in hidden space, likely making intra-cluster positioning of nodes to be largely independent from each other. This is due to the closeness measures being purely based on statistical similarity measures to clusters, disregarding other data points entirely. These loss terms will undoubtedly allow for much variation of the within-cluster structure and not utilising the assumed natural cluster structure that may originate from considering similarities with surrounding nodes. Another potential area of improvement with the MAGCN model is related to the issue of over-smoothing in deep GCNs [76]. Having the graph attention considered in eq. 4.19 only consider one neighbour, thus, potentially considering reaching L degrees of neighbours in a L layer GCN. Further-reaching attention per layer could potentially compress the model potential by being more selective in each graph convolution layer.

Part II

Method & Data

5

Proposed Method

5.1 Motivation

Having covered background theory of clustering, deep learning, and multi-view learning, we have built a foundation to present our proposed deep graph-based multi-view clustering model. This thesis presents a novel approach that aims to unify the advantages of representation-alignment as documented by Trosten et al. [104], with a new interpretation of the graph embedding layer utilised in graph-based multi-view clustering methods [13; 95; 40]. The method aims to combine the per-sample linkage of views — borne of the contrastive module in CoMVC — with a focused attention-based graph embedding encoder that acts both as a smoothing operator and a redundancy filter [78]. The novelty of our graph attention approach is that of a new approach to graph attention weighting, using Markov chain probabilities to improve the selective process when applying attention, reducing the attention space from the full graph, while making better use of further-reaching attention neighbourhoods. We also hypothesise that the prior may reduce the smoothing effect frequently observed in GCNs as the number of layers increases [76] when combined with a suitable sampling operator. With our solution, we reason that each layer may attend to distant nodes and avoid the penalty of excessive over-smoothing [76], which will be explored in chapter 7.

Given an ability properly able to unify views on a sample-to-sample basis we hypothesis that our model may provide more informative embedding spaces

and transform raw, unlabeled data into more stable subspaces compared to the existing SOTA, which attends to a much more general cluster structure [13]. Looking at the medical domain in particular, unsupervised methods are in many cases plagued by noisy data as well as significant black-box problems in inference. Recognising the success of the *contrastive module* in CoMVC [104] for the unification of multi-view data to recognise commonality, we reason that the method may still lack the relational information required to properly filter noise and redundancy and enhance the underlying group structures. Assuming that this is especially problematic for complex and imperfect data, we assess that the incorporation of group structures observed in relational data, with the already existing feature-based clusters — will yield more robust results.

5.2 Graph Attention

We firstly introduce our interpretation of the graph attention layer. Unlike methods such as MCGC [78] using a fully learned graph, we reason that the attention solution utilised in e.g. MAGCN [13] and GAT [107], using parameters invariant to dataset size improves its flexibility immensely — allowing for e.g. batch sampling as well as inference on unseen data using a pre-trained model. Unlike the MAGCN, we develop a novel approach based on a Markov chain prior used in the graph attention module.

5.2.1 The Markov Prior

In order to create a neighbourhood mask of our attention, we employ a random walk Markov model [88] and treat each node as each own state. Using this, we assume that a given state u may be associated with another state v by the edge weight connecting the two nodes. Having adjacency matrix $\boldsymbol{A}$, the edges of every node may be represented as a probability by normalising each row using graph normalisation

$$\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}.$$

This function assumes that the transition probabilities are given by each edge weight normalised by all outgoing edge weights from node u. The first assumption made is that the random walk probabilities are proportional to the weights of edges. We assess this as a reasonable assumption for similarity-based graphs. Now, by assuming that a state v_j at time $k+1$ is affected only by the previous state we may define the stochastic process as a Markov chain. The transition probabilities are then stated as the following:

$$P_{ij} = P\{v_{k+1} = j | v_k = i\}. \tag{5.1}$$

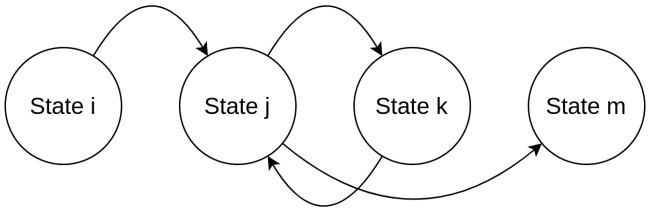


Figure 5.1: A Markov process from node i to node m: $i \rightarrow j \rightarrow k \rightarrow j \rightarrow m$.

Extending this, we may define the n-step transition probability from node i to node j,

$$P_{ij}^{n} = P\{v_{k+n} = j | v_k = i\},\ n \geq 0.$$

By applying the Chapman-Kolmogorov equations we may prove the relationship between nodes after $n + m$ transitions may be stated as such [88]

$$\begin{aligned} P_{ij}^{n+m} &= P\{v_{n+m} = j | v_0 = i\},\ \ n, m \geq 0 \\ &= \sum_{k=0}^{\infty} P\{v_{n+m} = j, v_n = k | v_0 = i\} \\ &= \sum_{k=0}^{\infty} P\{v_{n+m} = j | v_n = k, v_0 = i\} P\{v_n = k | v_0 = i\} \\ &= \sum_{k=0}^{\infty} P_{ik}^{n} P_{kj}^{m}. \end{aligned} \tag{5.2}$$

Letting P_{ij}^{t} be the transition weights w_{ij} associated with nodes in our adjacency matrix $\boldsymbol{A}$ after t time steps, and with the full matrix representation denoted as $\boldsymbol{P}^{(t)}$ (with $\boldsymbol{P}^{(0)} \equiv \hat{\boldsymbol{A}}$), we use eq. 5.2 to define

$$\boldsymbol{P}^{(n)} = \boldsymbol{P}^{(n-1)} \hat{\boldsymbol{A}} = \hat{\boldsymbol{A}} \cdots \hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}^{n}. \tag{5.3}$$

Having found probabilities for transitioning from node i to node j in exactly n steps given by $\boldsymbol{P}^{(n)}$, we reason that the similarity measure defining the n-step neighbours is insufficient in described similarity between nodes i and j. We reason that a superior measure of familiarity is the number of times node i has visited j after n steps. This will include, not only the start node and end node, but rather the full probability describing every encounter along the way. The neighbourhood matrix used for masking is then proportional to the sum of the Chapman-Kolmogorov transition probabilities.

$$\boldsymbol{M}^{(n)} = \sum_{k=1}^{n} \boldsymbol{P}^{(k)} = \sum_{k=1}^{n} \hat{\boldsymbol{A}}^{k}. \tag{5.4}$$

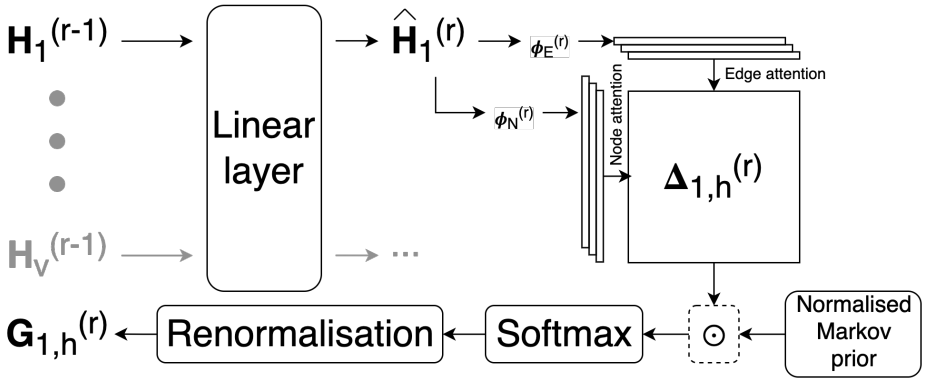


Figure 5.2: Finding graph attention $G_{v,h}^{(r)}$ in the graph attention layer.

Assuming a rapid growth of the number of non-zero edges in $\boldsymbol{P}^{(k)}$, we reason that its probabilities will naturally decline as k increases, given its restriction as a probability matrix having to sum to N in its entirety. Looking at figure 5.1, this would correspond to the weight from node i to node j being commonly higher weighted (due to the recurring early transitions to node j) than the weight of transitioning from i to m in 4 steps. This will create a natural focus on closer nodes, as well as continually strengthening nodes that are regularly re-visited. In order to assess $\boldsymbol{M}^{(n)}$ as a probability, normalisation must be applied to all nodes through normalisation along the horizontal axis as such

$$\hat{\boldsymbol{M}}^{(n)} = \boldsymbol{D}^{-1}\boldsymbol{M}^{(n)}, \tag{5.5}$$

being a matrix of probabilities of having visited any given node (given by column index j), starting at any node (given by row index i) after n random steps.

5.2.2 Graph Attention Convolutions

In order to properly embed graph information into our data, we define the graph convolution process we employ in DRAGMVC. To encourage the extraction of commonality across views, all parameters of the operation are to be shared across views, as inspired by MAGCN [13]. This results in the operation learning to extract features while strengthening the requirement for similar view representations. The operation will propagate the contrastive loss used in representation alignment in addition to the clustering loss, thus also optimising the operation for maximising feature extraction of shared features.

We introduce the operation by looking to the standard GCN equation (eq. 3.53),

splitting the process into two sub-operations: a linear transformation and a graph convolution. Thus, we initially transform input $\boldsymbol{H}_v^{(r-1)}$ as follows

$$\hat{\boldsymbol{H}}_v^{(r)} = \boldsymbol{H}_v^{(r-1)}\boldsymbol{W}^{(r)}, \tag{5.6}$$

using shared weights for all $v = 1, ..., V$.

The graph attention is computed given a set of weight parameters. With $d^{(r)}$ features $\boldsymbol{\phi}_N^{(r)} \in \mathbb{R}^{d^{(r)}}$ and $\boldsymbol{\phi}_E^{(r)} \in \mathbb{R}^{d^{(r)}}$ define the weights for vertices and edges, respectively. Linear transformations of node and edge weights are performed and thereafter activated using activation functions φ_N and φ_E, as such

$$\boldsymbol{\alpha}_{N,v}^{(r)} = \varphi_N\big(\hat{\boldsymbol{H}}_v^{(r)}\boldsymbol{\phi}_N^{(r)}\big) \tag{5.7}$$

$$\boldsymbol{\alpha}_{E,v}^{(r)} = \varphi_E\big(\hat{\boldsymbol{H}}_v^{(r)}\boldsymbol{\phi}_E^{(r)}\big), \tag{5.8}$$

as displayed in figure 5.2. In our experiments $\varphi_N = LeakyRELU = \varphi_E$ with negative slope 0.2, as in [107]. The use of LeakyReLU activation allows for a gradient leak for inactive attentions as well as slight negative attention proving useful in cases where node attentions and edge attentions contradict. It will also allow the model to "disable" certain attentions by setting negative values.

We compute the corresponding attention matrix $\boldsymbol{\Delta}_v^{(r)} = \text{ReLU}\left(a\left(\boldsymbol{\alpha}_{N,v}^{(r)}, \left(\boldsymbol{\alpha}_{E,v}^{(r)}\right)^T\right)\right)$ as done in the graph attention auto-encoders (GATE) model [91]:

$$\boldsymbol{\Delta}_v^{(r)} = \text{ReLU}\left(\begin{bmatrix} \alpha_{N,v}^{(r)}(0) + \alpha_{E,v}^{(r)}(0) & \cdots & \alpha_{N,v}^{(r)}(0) + \alpha_{E,v}^{(r)}(N) \\ \alpha_{N,v}^{(r)}(1) + \alpha_{E,v}^{(r)}(0) & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \alpha_{N,v}^{(r)}(N) + \alpha_{E,v}^{(r)}(0) & \cdots & \alpha_{N,v}^{(r)}(N) + \alpha_{E,v}^{(r)}(N) \end{bmatrix}\right) \tag{5.9}$$

where a denotes element-wise summation of edges and nodes along the horizontal and diagonal axes. The resulting attention matrix is a $N \times N$ matrix of attentions, activated by the ReLU activation function to allow for apt "disabling" of values $\Delta_v^{(r)}(i, j) \leq 0$.

The matrices $\boldsymbol{\Delta}_v^{(r)}$ are weighted by the prior $\hat{\boldsymbol{M}}^{(n)}$ from section 5.2.1, eq. 5.5, creating attention graphs

$$\tilde{\boldsymbol{G}}_v^{(r)} = \text{SparseSoftmax}\big(\boldsymbol{\Delta}_v^{(r)} \odot \hat{\boldsymbol{M}}^{(n)}\big). \tag{5.10}$$

We define SparseSoftmax as Softmax using $\exp(x_i) = 0$ for $x_i = 0$, which corresponds to a Softmax with $x_i = 0 \rightarrow x_i = -\infty$. This will avoid unnecessary attention to every edge in $\tilde{\boldsymbol{G}}_v^{(r)}$ (yielding attentions > 0 for all values in

$\tilde{G}_v^{(r)}$) and thus reduce unnecessary smoothing by the GCN. The advantage of using a Markov prior in eq. 5.10 is two-fold: it provides a more informative initialisation for graph attention; it allows for further-reaching attention than the traditional graph masking which only attends to graph neighbours and more specialised attention than the unmasked attention struggling with over-smoothing as datasets increase in size $N \longrightarrow \infty$. The added benefit of possibly disabling attentions by utilising the LeakyReLU activation in eqs. 5.7 & 5.8 will further contribute to a narrower attention space, reasoned to further optimise the ratio of information gain to over-smoothing.

Finally, the output of the embedding layer is given by

$$\boldsymbol{H}_v^{(r)} = \sigma\Big(\boldsymbol{G}_v^{(r)} \hat{\boldsymbol{H}}_v^{(r)}\Big), \tag{5.11}$$

using the renormalisation trick from section 3.3.1:

$$\boldsymbol{G}_v^{(r)} = \Big(\tilde{\boldsymbol{D}}_v^{(r)}\Big)^{-1/2}\Big(\tilde{\boldsymbol{G}}_v^{(r)} + \boldsymbol{I}\Big)\Big(\tilde{\boldsymbol{D}}_v^{(r)}\Big)^{-1/2}.$$

$\sigma(\cdot)$ denotes the activation function set to ReLU.

5.3 Model Architecture

Putting together DRAGMVC, we assert that view encoders should be feature reduced representations of the raw input data creating similar representations across views, thus opting to use independent and fitting view encoders for the view at hand. The views are reduced to their encoded representations of a predetermined common dimensionality. Following the process described in section 5.2.2, we pass the views through a graph attention encoder with shared parameters for each layer. From this, we aim to strengthen and extract commonality across the view embedding space — and in the process remove irrelevant noise. Having the graph-embedded encoding of our data, we perform weighted fusion using view weights $w_1, ..., w_V$ as in [104]. The final clustering is performed by the DDC clustering module [50] on a graph attention-embedded fused representation. The aim of the shared-weight graph attention embedding for the views was to extract commonality, while its application in the DDC module is to attend to fully fused representations without any constraints from other views. The effects of graph attention in the DDC module will be quantitatively explored and analysed in section 7.4.3.

For the loss function we use the generalised NT-Xent [12] as utilised in [104] for the CoMVC model. The contrastive module is applied after the graph attention

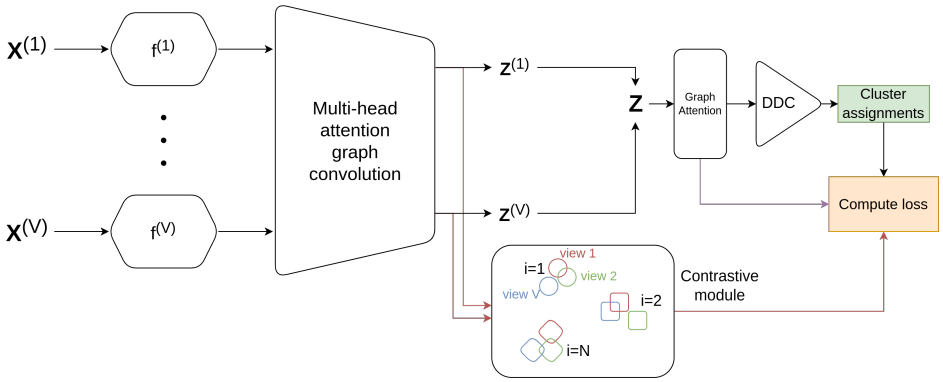


Figure 5.3: The proposed model.

layers. As a consequence, we get the desired effect of view alignment in the view encoders as well as in the graph embedding, thus guiding both the view feature extraction modules and the graph attention module. We do not apply any reconstruction loss to our model, as is common in many encoder structures (e.g. MAGCN [13], GATE [91]). This is due to the observation that reconstruction loss strongly mitigates contrastive loss, reducing the ability to properly align representations (see appendix A.3.1).

6

Proposed Dataset

The experiments on our MIMIC dataset are performed in an unsupervised and semi-supervised manner, thus — due to the limitations of the methods used — having a thorough pipeline which covers data extraction, coupling data from different databases, creating pseudo-labels, extracting and processing static data, as well as processing of image data is critical. This subsection will cover the structure of the database, the dataset creation process, as well as the processing that was done. We will initially cover a brief introduction to two databases for context, but will not delve deep into the details of the creation of the database itself, as this topic is beyond the scope of this thesis[1]. The majority of this chapter will focus on how tables in the databases were linked, pre-processed, and the results of the data cleaning process. In addition, we will detail the process of creating an associated affinity graph for the dataset.

The databases used were the Medical Information Mart for Intensive Care IV (MIMIC-IV) and its chest X-ray counterpart: MIMIC-CXR [47; 49; 24]. These are publicly available, de-identified databases from the Beth Israel Deaconess Medical Center (BIDMC) Emergency Department in the years 2008 till 2019, and 2011 till 2016, respectively. MIMIC-IV presents critical care data for more than 40 000 patients, while the MIMIC-CXR provides imaging studies for 65 379 patients, yielding 377 110 separate images. The author of the paper [47; 49]

1. For more information about the creation of the databases, please refer to the publications [24; 47; 49].

has confirmed that the two datasets are linked by subject identifiers [48] which we will use to create a multi-modal dataset such that each X-ray image corresponds to data from MIMIC-IV in a given time frame around the time and day at which the X-ray was captured. Each data point will include properties of the static data and one X-ray image.

6.1 Databases

6.1.1 MIMIC-IV

MIMIC-IV is a database of critical care data. This consists of measurements done throughout their hospital stay such as vital signs, lab tests, chart data and observations made, drugs the patients have been prescribed, diagnoses, transfers, procedures, and more. The database has gone through a number of steps prior to the publication. Namely: acquisition, preparation, and de-identification [47]. The data was extracted from the BIDMC emergency department and intensive care units (ICU). From this point, the data was organized in a way such that it was suited for data analysis, which include de-normalisation and reorganisation into fewer tables [47]. Data cleaning was not performed such that the data would not lose its likeness to real-world data. Finally, the data was de-identified appropriately. Dates and times were randomly shifted. However, the shift was applied on a subject-to-subject basis and thus, we may — for a single subject — compare information by relative time differences, e.g. image acquisition time relative to hospital admission.

6.1.2 MIMIC-CXR

MIMIC-CXR is a chest X-ray database that contains two items, namely chest X-rays in DICOM data format and free-text radiology reports [49]. Similarly to MIMIC-IV, MIMIC-CXR is de-identified by removal of protected health information (PHI). This include — among the alterations mentioned in the previous subsection — adding black boxes to X-ray images for removal of sensitive information and eliminating PHI from the free-text reports.

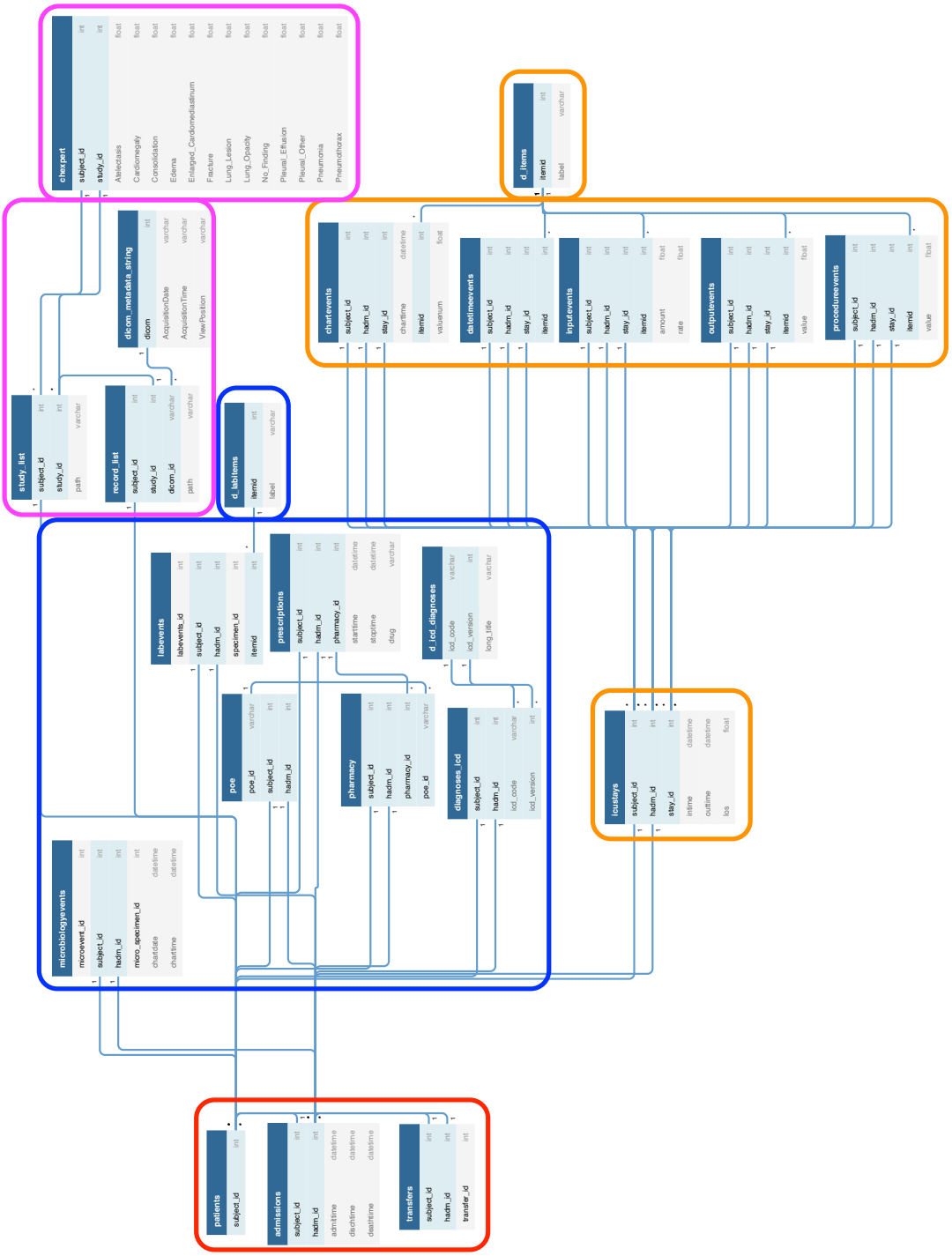


Figure 6.1: Database structure. "*mimic-core*" in red, "*mimic-hosp*" in blue, "*mimic-icu*" in orange, "*mimic-cxr*" in pink.

6.2 Extraction

6.2.1 Database Diagram

A database map of MIMIC is illustrated in figure 6.1. A range of tables and fields were not included in this map due to the sheer total number. It would only serve to clutter the figure above while not providing any information relevant to this thesis. For the sake of order, the complete list of tables included in the datasets is provided in section A.4.2.

6.2.2 Data Extraction & Filtering

The main goal of our cleaned dataset is to unite ICU data with X-ray data. ICU data (*mimic-icu*) was selected over the more general hospital data (*mimic-hosp*) by the reasoning that ICU patients are significantly more closely monitored and thus will provide more data during the the time frame in which X-rays were captured. Thus, the initial step of our extraction process is to find overlapping *subject_id*s and *hadm_id*s that appear in both datasets at the same period. This can be done by simply joining the tables on subject identifier and ensuring that the acquisition time of the X-ray is during the patient's stay at the ICU.

The next step is to select the distributions from which to draw our samples, e.g. k lung diseases. It is important to note that — as this is in preparation for an unsupervised machine learning model — we do not require this grouping to be the grounds for the output clusters, it is rather included to assure that there *exist* some underlying group structure in the data. For this, we join tables *diagnoses_icd* and *d_icd_diagnoses* on ICD code and version. Then, we wish to find the population that covers e.g. disease 1, while excluding the patients with any combination of diseases $2, ..., k$. From these k populations we draw $N_{min} = \min\left(N_1, N_2, ..., N_k\right)$ samples from each population without replacement, ending up with a balanced dataset of $N = k \cdot N_{min}$ samples.

Having the patient samples, we may now generate the image input data by linking it to the MIMIC-CXR database. The images are downloaded and presented for local data processing. The static data tables selected are *vital signs* and *labevents*. Vital signs contain information on: heart rate, blood pressure (and its variations), respiratory rate, temperature, oxygen saturation, and glucose levels. The second table, lab events, was selected to increase the range of measurements done. However, by using multiple tables we may define different time slots in which to gather data depending on the table in question. For example, lab tests could be less frequent than vital signs and thus should be

collected with a bigger time frame in mind.

These tables makes for a multi-view dataset on their own, as it is generated from a separate set of observations. The data is then — similar to the image data extraction — extracted within a given time frame, with fields *dicom_id*, *chart_time*, *valuenum*, and *label*. Lab events not accounted for in less than 40% of the patients are removed. The value of 40% is based on the amount of data needing to undergo imputation in the processing stage, as $> 40\%$ missing data is considered substantial [45]. Another reason was the intention to exclude rare tests, as such tests could be a strong indicator of a given disease and thus create groupings that heavily depend on rare data. Following this filtering, the table is pivoted such that each label gets its separate column given by value *valuenum* at index *dicom_id*. As each index may corresponds to many measurements, we generate six statistical parameters to represent the data for any given patient, *first*, *last*, *minimum*, *maximum*, *median*, and *mean*. Assuming that the data is non-categorical in nature, these parameters should provide an accurate picture of the dataset in both a statistical and temporal sense. Standard deviation was excluded due to the inconsistency of measurements taken. Seeing as the period and seriousness of illness may vary significantly, patients will not be measured an equal number of times nor retain critical values for equally long. We reason that this will therefore likely disrupt features more than assist in unsupervised applications. Finally, the maximum / minimum values describe potential spikes / dips in the values.

6.3 Data Processing

The image processing consists of two main steps generally used in medical image analysis: contrast enhancement and resizing. In this subsection we will cover the decisions made and describe the processing steps for the chest X-rays, as well as for the static data. For the image data, we performed all image processing techniques prior to resizing as to not lose information when processing, e.g. in histogram transformations.

6.3.1 Processing the Static Data

We apply our imputation method of choice. Imputation in medical data is often a necessity as there is a large number of missing variables (e.g. from certain patients not having taken a test — willfully or not) as removal of the missing data will often be far worse than having any arbitrary imputation method [39]. There are a number of approaches to imputation. However, for many

cases is often preferable to make no assumptions regarding whether the data is Missing Completely At Random[2] (MCAR), Missing At Random[3] (MAR), or Missing Not At Random[4] (MNAR), and use the zero imputation (i.e. setting `NaN` values to 0, before data normalisation) [39]. This will often yield values far outside of the expacted range, seemingly lacking intuitive sense. However, as this imputation is not of any clinically meaningful value [39] we reason that a machine learning model will learn to segregate abnormal values on its own, potentially explaining its often unexpected high performance [39].

After setting `Nan`'s to 0 (zero imputation) we perform data normalisation, as is common for deep learning approaches [41]. In our case, this was done by Z-score normalisation given by

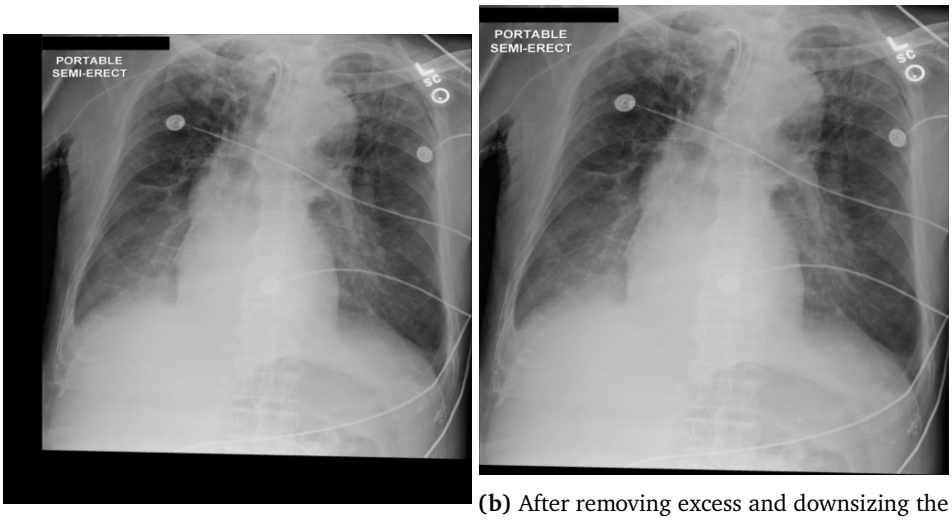
$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

for sample $i = 0, 1, ..., N$ and feature $j = 1, ..., p$. This removes magnitude differences from having data in a range of different units, and allows the network to weigh input nodes on an equal basis.

6.3.2 Interpolation

For interpolation, cubic spline (B-spline) [111] was selected, as it is well-suited for medical image tasks such as ours [60]. Before down-sizing the image, we rotated the image to a number of different angles $(-45\,\text{deg}, 45\,\text{deg})$ and counted the number of black columns and rows. Angles with the maximum number of black rows and columns is selected and the black bars removed (see fig. 6.2), in order to maximise the image content. This was a necessary pre-processing step as a number of images was randomly rotated by a significant margin. The image size selected was 256×256 as this is regularly used in deep learning implementations for chest X-rays (as in [102; 32], or the comparable resolution 224×224 as in [75; 82]). The small size does not provide any very significant disadvantage compared to the larger size alternatives 512×512 and 1024×1024 [102; 32], while being far less memory demanding and commonly requiring fewer training parameters.

2. The missing data has no correlation with other features [39]. E.g. a dropped vial containing a lab measurement.
3. Not missing at random. Indirectly depend on other observed variables in the dataset [39]. E.g. if men are more likely to drop out of a clinical trial.
4. The missing data is directly correlated with other features [39]. E.g. a missed drug test.



(a) Image pre-rotation.

(b) After removing excess and downsizing the image.

Figure 6.2: An X-ray image before and after removing excess.

6.3.3 CLAHE

Contrast Limited Adaptive Histogram Equalisation (CLAHE) [80] has been a successful processing step in various chest X-ray deep learning classification approaches [90; 119]. Its benefit is to improve contrast while limits the amplification of noise [119].

Rather than equalising the histogram of intensities over the entire images, CLAHE operates on small tiles of the image. The boundaries between tiles are then merged by interpolation. The result is that both lighter and darker areas receive much more contrast relative to its immediate surroundings. Given the previously referenced improvements that has been observed with CLAHE in chest X-rays, this was deemed a favourable step in pre-processing. The resulting processed image is shown in figure 6.3 (as compared to fig. 6.2b).

6.4 Graph Construction

In order to evaluate the MIMIC dataset on our graph-based MVC method, we needed to construct a relevant graph, as this is not included in the MIMIC database. Thus, we constructed four informative graphs for three different methods. One graph is created with full access to label data, one graph is created without label data, and two graphs are created with respectively $N_1 = 100$

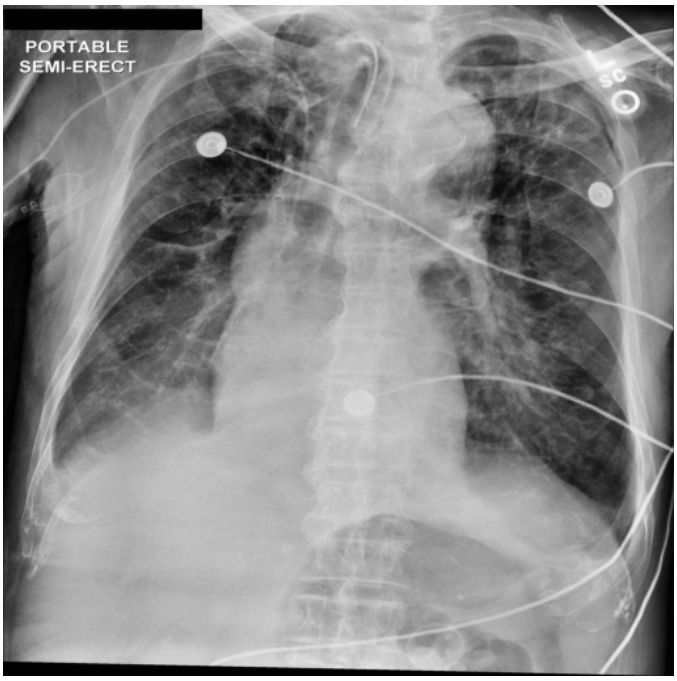


Figure 6.3: A fully processed image from input 6.2a.

and $N_2 = 500$ balanced labels as a semi-supervised approach.

The source of the graph is a difficult choice, having many possibilities as well as a lacking precedence in the literature for medical applications. Parisot et. al [79] and Cao et. al [10] built medical graphs by having vertices correspond to fMRI image features, and edges correspond to a similarity of image features and non-image features. However, fMRI images are much more clear-cut in activation than X-ray images. Similarities between X-ray images are hard to define in an unsupervised manner. Our solution is to utilise the associated radiology reports of X-rays. The radiology reports are to be valued based on a weighted average of their words. The word weights are given as a simple difference in expected word frequency, n_k between groups. We define two groups C_1 and C_2 denoting the labels for our supervised evaluation graph. P_1 and P_2 denote predicted classes by classifiers for both the semi-supervised and unsupervised cases. The weight for word k is given by:

$$\begin{aligned} \omega_k &= \mathbb{E}\big[n_k \big| k \in C_1\big] - \mathbb{E}\big[n_k \big| k \in C_2\big] \\ \omega_k &= \mathbb{E}\big[n_k \big| k \in P_1\big] - \mathbb{E}\big[n_k \big| k \in P_2\big]. \end{aligned} \tag{6.1}$$

The weight vector $\boldsymbol{\omega}$ is constructed by concatenating the weights for all words. The process is illustrated in fig. 6.4. The classifier used for unsupervised classi-

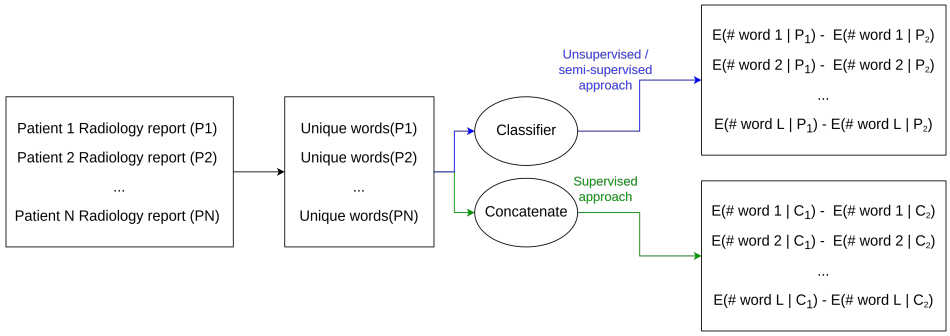


Figure 6.4: Our supervised and semi-supervised/unsupervised approaches to extract word weights.

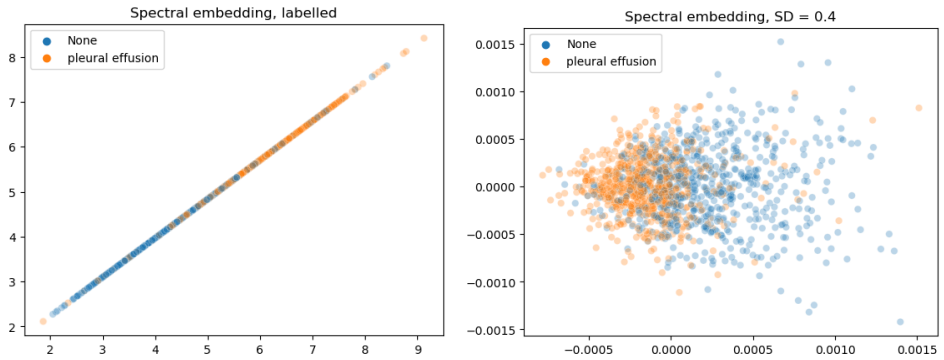
fication is set to K-means, and the semi-supervised approach is a self-training classifier, using a linear SVM as a base estimator. In the semi-supervised case, the linear approach is selected as to avoid over-fitting by limiting the decision boundary, thus making it comparable to the K-means approach in separability. We reason that a simplistic prediction algorithm is superior to a complex one given the graph's role as a very general indication of data affinity. The radiology reports are given as *bag-of-words* representations: one report corresponds to a vector of elements with the number of occurrences of a given word. The shape of the data is $N \times W$ where W is the total number of unique words in radiology reports. No filtering is applied as this should occur naturally due to the low word weights yielded by eq. 6.1 for words that are common throughout the dataset. The models are run on the full $N \times W$ data matrix and yields two prediction classes: P_1 and P_2. These are naively used as ground truth labels in an approach similar to self-training.

Having weight vector $\boldsymbol{\omega}$, all radiology reports are weighted, yielding the report values $\boldsymbol{t} = [t_1, ..., t_N]^T$, where

$$t_i = \langle \text{word counts}(i), \boldsymbol{\omega} \rangle$$

defines the report value by an inner product operator. A symmetric affinity matrix $\boldsymbol{A}$ — having $\hat{\mu}$ and $\hat{\sigma}$ be the first moment and second central moment of the elements in $\boldsymbol{t}\boldsymbol{t}^T$ — is constructed as such:

$$\begin{aligned} \tilde{\boldsymbol{A}} &= \left(\boldsymbol{t}\boldsymbol{t}^T - \hat{\mu}\boldsymbol{J}\right) \odot \hat{\sigma}^{-1}\boldsymbol{J} + \boldsymbol{E} \\ \tilde{\boldsymbol{A}}_+ &= \tilde{\boldsymbol{A}} - \min\left(\tilde{\boldsymbol{A}}\right) \\ \boldsymbol{A} &= \frac{\tilde{\boldsymbol{A}}_+ + \tilde{\boldsymbol{A}}_+^T}{2}, \end{aligned} \tag{6.2}$$



(a) Affinity graph without noise. **(b)** Affinity graph with noise.

Figure 6.5: A Laplacian eigenmap of affinity graph $\boldsymbol{A}$ (from fully labeled data) with and without added noise.

where $\boldsymbol{J}$ is defined as a $\mathbb{R}^{N\times N}$ matrix of ones and

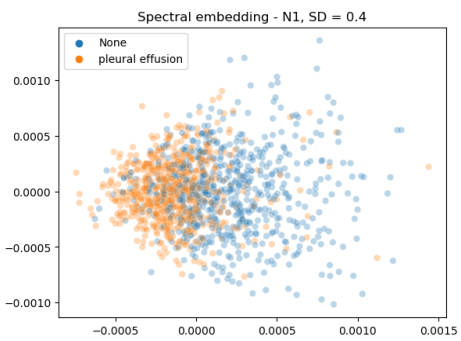
$$\boldsymbol{E} \equiv \begin{bmatrix} \varepsilon(0,s) & \cdots & \varepsilon(0,s) \\ \vdots & \ddots & \vdots \\ \varepsilon(0,s) & \cdots & \varepsilon(0,s) \end{bmatrix} \in \mathbb{R}^{N\times N}$$

is a matrix of random noise sampled in $\mathcal{N}(0,s)$, defining $\varepsilon(0,s=0)=0$. Noise is added to avoid issues arising with low data dimensionality as there is but a single input feature dimension. The issue at hand, visualised in fig. 6.5a, is that of overly linear spectral relations. Adding noise, will make between-node relations less linear. This is illustrated in figure 6.5 displaying a Laplacian eigenmap of matrix $\boldsymbol{A}$ (fully labeled) with and without added noise. The graphs for unsupervised and semi-supervised with N_1 and N_2 are subsequently given in figure 6.6.

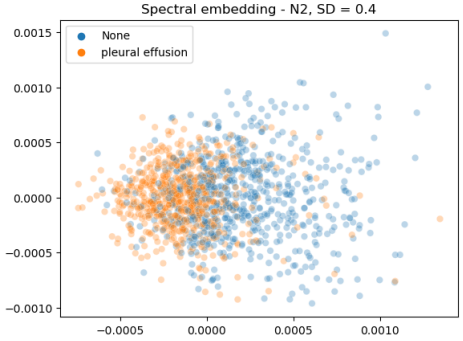
To assert the informativeness of the graphs in an unsupervised manner, we employ K-means clustering[5] with K-means++ initialisation (see section A.2.1.1). Firstly, we perform clustering on eigenmaps from randomly generated matrices $\boldsymbol{A}$, having $s = 0, 0.1, 0, 2, ..., 0.9$. The results can be seen in the histogram in fig. 6.7. Aiming to minimise s to retain the input information while aiming to maximise clustering performance, we set $s = 0.4$.

Now, we perform clustering on randomly generated eigenmaps such as the ones in figs. 6.5b, 6.6a, 6.6b, & 6.6c. With $N = 30$ runs, we recognise no significant accuracy related advantage using a supervised-generated graph over

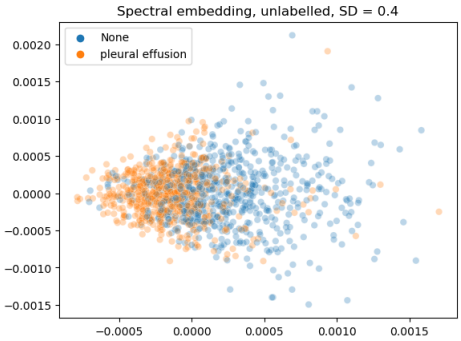
5. Every K-means model is determined by minimum inertia of 10 initiations.



(a) Semi-supervised (N_1) affinity graph.



(b) Semi-supervised (N_2) affinity graph.



(c) Unsupervised affinity graph.

Figure 6.6: Laplacian eigenmaps of affinity graphs $\boldsymbol{A}$ of the unsupervised and semi-supervised approaches.

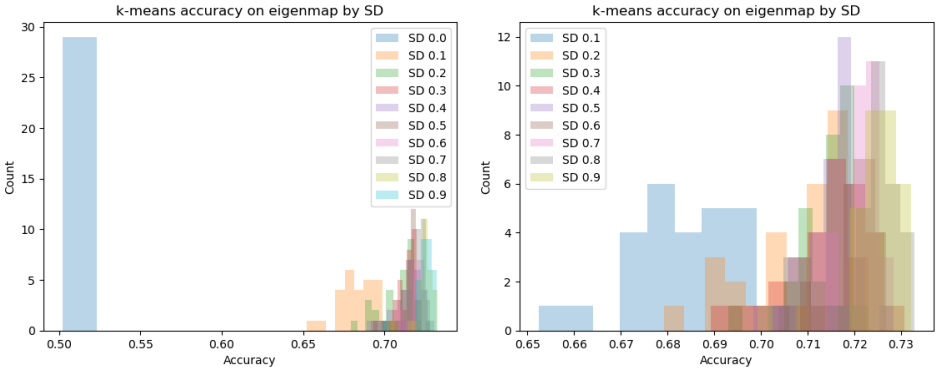
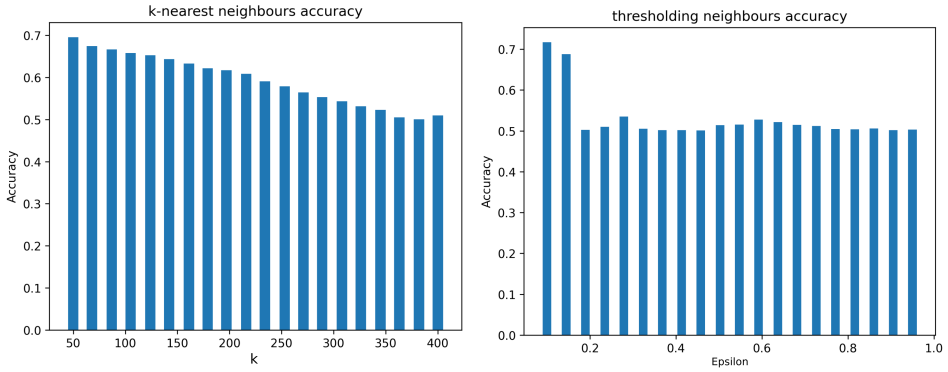


Figure 6.7: Histogram of K-means clustering accuracy on eigenmaps (fully labeled) by noise, in SD. Peaking at 73.3% clustering accuracy.

Table 6.1: Mean results of 30 K-means clustering models on Laplacian eigenmaps generated from affinity matrices with noise $s = 0.4$.

Graph	Accuracy	Standard deviation
Unsupervised	0.71532	0.00662
Semi-supervised (N_1)	0.70883	0.00622
Semi-supervised (N_2)	0.71197	0.00807
Supervised	0.71512	0.00602



(a) k-NN transform on the supervised affinity graph.

(b) ε thresholding on the supervised affinity graph normalised to $[0, 1]$.

Figure 6.8: Accuracy of 2-dimensional eigenmap K-means for a range of k-nearest neighbour and ε neighbourhood parameters.

unsupervised / semi-supervised based graphs. See table 6.1.

Lastly, considering the exponential growth of $N \times N$ affinity matrices as N gets large, we select appropriate neighbourhoods of edges. Both k nearest neighbour and ε neighbourhood are considered (see fig. 6.8). The optimal selection is one combining a high level of sparsity, for practical reasons, with high performance. Consider that the graph's unsupervised counterpart will likely be subject to higher uncertainty in its edges (thus excluding low-neighbour / low-threshold alternatives). Taking the above into consideration, a reasonable middle-ground is found to be a k nearest neighbourhood with $k = 140$ neighbours. Evaluation of the k-NN graph is detailed later in section 7.4.6.2. Now, due to the high separability recognised in all eigenmaps (figs. 6.5b & 6.6), the unsupervised k-NN (with $k = 140$) graph is decided to be used in our MIMIC multi-view graph dataset.

Part III

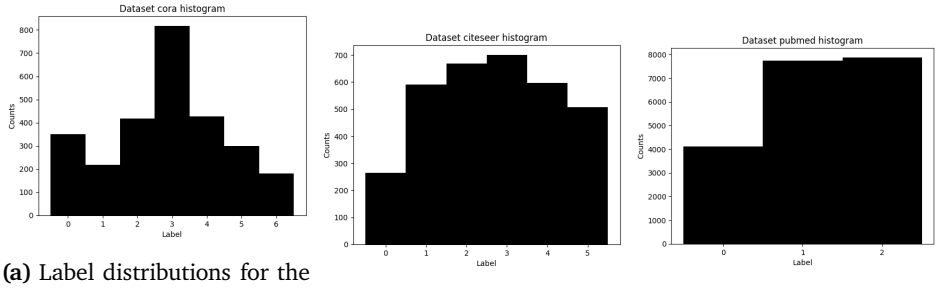
Experiments & Conclusion

7

Experiments

Prior to presenting our experiment results, we will briefly cover the datasets used, the tools used in analysis and their added level of insight, as well as the setup of our model. In our experiments we aim to shed light on ambiguities regarding our model and its use cases. Firstly, we will assess model performance on familiar datasets containing two complementary views. Here we wish to compare our model[1] to the state-of-the-art in a less complex clustering setting. After establishing the basic clustering potential of our model, we wish to determine parameter significance, the effects of dropout in the graph embedding layers, as well as the effects of replacing a fully-connected layer with a graph embedding convolution in the DDC clustering module. Following the experiments on model architecture and parameter values, we progress to experimentation on our real-world dataset. For our MIMIC experiments, we wish to determine the limitations of contrastive learning for proper feature extraction and the abilities and limitations of applying our model to highly complex data. We will assess the importance of views and the effects of view count in unsupervised contrastive learning by analysing representation plots, and finally, we will consider the semi-supervised approach to clustering and its effect on fusion space and the contrastive loss. Following our experiments, we will discuss our findings and limitations.

1. The source code of DRAGMVC can be found on GitHub [28].



(a) Label distributions for the Cora dataset. Coefficient of variation: $\frac{SD_{cora}(n_{C_i})}{\mathbb{E}_{cora}[n_{C_i}]} = 0.548$.

(b) Label distributions for the CiteSeer dataset. $\frac{SD_{cs}(n_{C_i})}{\mathbb{E}_{cs}[n_{C_i}]} = 0.284$.

(c) Label distributions for the PubMed dataset. $\frac{SD_{pm}(n_{C_i})}{\mathbb{E}_{pm}[n_{C_i}]} = 0.326$.

Figure 7.1: Label distributions for secondary datasets.

Table 7.1: Specifications for our selection of datasets.

	Nodes	Classes	$d_{(1)}$	$d_{(2)}$	$\min_{C_i} n_{C_i}$	$\max_{C_i} n_{C_i}$
Cora	$N_{cora} = 2\,708$	7	1 433	2 708	180	818
CiteSeer	$N_{cs} = 3\,327$	6	3 703	3 327	264	701
PubMed	$N_{pm} = 19\,717$	3	500	19 717	4 103	7 875

7.1 Datasets

We describe the process of creating the MIMIC dataset creation and its preprocessing in section 6. The views used are images and vectorial data. The tables from which the linear views are gathered are those containing vital signs gathered from *mimic_derived*, as well as lab results from *mimic_hosp*. The images are chest X-rays from MIMIC-CXR which are bound to vectorial data by subject ID and registered in a 1-day time frame. The final dataset consists of 3 520 samples containing three views and two classes: *pleural effusion/not pleural effusion*. Each sample correspond to one X-ray image and its associated static data. The three views have, respectively, shapes $(1, 256, 256)$, $(66,)$, and $(210,)$. A graph is computed in an unsupervised manner by a similarity measure based on the radiology reports. This is described in detail in section 6.4. The graph is further simplified by applying the k nearest neighbour transform — the effects of which are covered in section 7.4.6.2.

For our secondary datasets, we use the Cora and CiteSeer [93] and PubMed [73] datasets. The datasets are real-world bibliographic datasets of various papers being divided into a number of different categories [93]. Cora and

CiteSeer were pre-processed by stemming[2], removing stop words[3] and words with fewer than 10 occurrences. The Cora dataset was composed of 2 708 machine learning papers with 1 433 distinct words and 5 429 links (graph edges); the CiteSeer dataset was composed of 3 327 papers with 3 703 distinct words and 4 732 links. The PubMed dataset was composed of 19 717 PubMed papers using words corresponding to the 500 highest TF/IDF scores[4] as binary attribute features. The links making up the graph in PubMed has been set as undirected and make up 44 438 edges in total [73]. A summary of the three secondary datasets can be found in table 7.1 and label distributions in fig. 7.1.

Similar to [13], a second view is constructed by using the Cartesian product[5] $\boldsymbol{X}_2 = \boldsymbol{X}_1\boldsymbol{X}_1^T$, each sample having N attribute features. The datasets are binary valued as well as having unweighted graphs unlike those of our constructed MIMIC dataset. All graphs are assumed to be undirected, static, and homogeneous in that multiple views are associated with every node, but graph topography and edge relations are identical in all views.

7.2 Analysis tools

In our MIMIC dataset, experimental results originate from multi-view data with an unknown number of underlying natural clusters (other than *pleural effusion/not pleural effusion*), and for patients with long lists of diagnoses. Thus, the results can be hard to analyse in a clear-cut way. In this part we will go through the analysis tools we will use, as well as define their aid in interpretation.

7.2.1 Dimensionality Reduction

Firstly, presenting a dimensionality-reduced scatter plot of the embedded representations before and after fusion will help us better understand the level of mixing that happens. This will especially help highlight the effect of contrastive

2. Reducing words to their word stem.
3. Commonly used words that are removed due to lacking information, e.g. ”the”.
4. Term frequency/inverse document frequency (TF/IDF) is given as the product of the relative term frequency in a given document $tf_{t,d} = f_{t,d}/\sum_{t' \in d} f_{t',d}$, with the inverse document frequency — used as a measure of how informative a given word is by comparing across all documents D: $idf_{t,D} = \log(N/df_t)$ where df_t is the number of documents that a given word t appears in [68].
5. As found to be superior with the MAGCN [13] to the Fourier, Gabor, and Euler transforms by Cheng et al. [13].

loss as we wish to observe sample-level mixing from the contrastive loss term. The dimensionality reduction techniques used is t-SNE (see appendix A.1.2). The reasoning behind our selection of data visualisation was with the goal of enhancing the natural groupings in the hidden space. Visualising the similarity between views in pre-fusion space will help us evaluate the mixing abilities of the method. In other words, t-SNE will more accurately display the probability of features from different views being located in the same estimated distribution than using non-statistical measures such as PCA.

7.2.2 Plot of X-ray Images

Another analysis method is to sample predictions and analyse the X-ray images in each cluster. This gives an intuitive understanding of the differences picked up by the model. Although we lack the proper expertise to evaluate these images, we may yet detect whether clusters detect differences within the lung region or simply by arbitrary differences in colour and/or contrast.

7.2.3 p-values

In order to assess whether natural groupings exists in clusters, p-values are calculated for patient diagnoses by the difference between clusters. This means that the proportional distributions for each diagnosis is used in a two- or multi-sample proportion test from which the p-values are calculated. In the case of $k = 2$, this reduces to a simple *two-sample T-test for proportions* with $H_0 : p_1 - p_2 = 0$ given by [108]

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \tag{7.1}$$

where $\hat{p}_1$ and $\hat{p}_2$ are the diagnosis sample proportions in the two clusters, respectively, and $\hat{p}$ is the diagnosis sample proportion for the data in its entirety. From using double the corresponding probability value for $|z|$ (which will correspond to a one-sided hypothesis) we get a final p-value for the two-sided alternate hypothesis $H_1 : p_1 - p_2 \neq 0$.

Having p-values will aid by highlighting which diagnoses are significantly different between the prediction clusters. However, as the number of diagnoses is as large as it is, p-value correction is in order to control the number of randomly occuring significant values. For this, we selected the conservative

Bonferroni correction[6] [7] and find our adjusted p-values by:

$$p_{adj} = mp_i \tag{7.2}$$

where p_i are our standard p-values and m is the number of features.

7.2.3.1 p-value Histograms

Using the (pre-correction and post-correction) p-values calculated for each diagnosis in the dataset, we plot a histogram of all p-values to assess the p-value distribution. From this we desire a tendency towards significant p-values, i.e. a left-shifted histogram. This suggests that significant values are not due to random deviations in the populations, but in fact discovered by the model. Displaying the post-corrected p-value histogram will provide the reader with a relative display of the number of significant diagnoses. Now, given that p-values are shown to be significant: a large left-shift post-correction will imply a high number of correlated diagnoses, while a lesser left-shift will indicate that clusters are correlated to fewer diagnoses.

7.2.4 Diagnosis Word Cloud

The final analysis tool is — similar to the sampled X-rays — a visualisation tool. We create a word cloud of diagnoses[7] for each prediction cluster to hopefully ease the process of distinguishing clusters from each other. This process requires more customisation and therefore likely to induce more bias than the other analysis tools, as simply inserting all diagnoses into a word cloud will clutter the image with too common diagnoses and filler words such as ”and”, ”the”, ”with”, and so on. To prevent these issues we subtract all the diagnosis counts by the population in the least populous cluster, e.g. if a split with three clusters yields $(100, 200, 50)$ positive cases the altered split will be $(50, 150, 0)$. In addition to this, we remove common words[8].

7.3 Setup

For all hidden layers in view encoders as well as the graph embedding encoder we use ReLU activation after applying batch normalisation. We initialise all

6. Stating that the adjusted significant value is found by $\alpha_{adj} = \alpha/m$ where α is the standard significant value and m is the number of p-values.
7. The diagnosis list is drawn from the corresponding ICD diagnosis titles from each person's hospital admission ID.
8. A complete list of the words removed from diagnoses are given in table A.2 in the appendix.

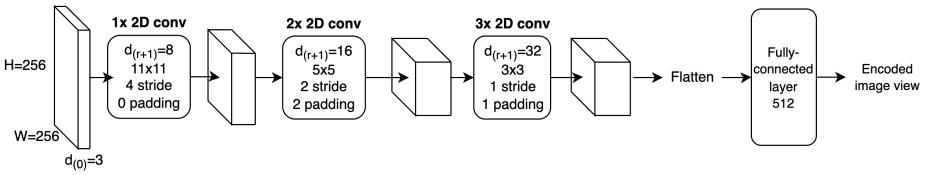


Figure 7.2: Model image encoder.

model weights and graph attention parameters using He normal initialisation [36] aiming to control the feature distribution of a deep neural networks having the ReLU non-linearity [36]. The initial biases are set to constant 0 to avoid shifting the output distribution from the standard normal. For optimisation, we utilise the Adam optimiser [56] with learning rate 10^{-3} for all datasets. We implement dropout in the graph embedding layers following the linear transformation (eq. 5.6) as well as for the node and edge weights (eqs. 5.7 & 5.8) similarly to the dropout applied in GAT [107]. We evaluate a range of dropout probabilities in the experiment in section 7.4.2. The metrics selected for evaluation are *ACC*, geometric *NMI*, *ARI*, and macro-$F1$ (see appendix A.2.2), and all confidence intervals in our metrics assume a normal sample distribution using significance level $\alpha = 0.05$, i.e. 95% confidence intervals.

7.3.1 MIMIC Dataset

For the MIMIC dataset we determine minimalistic encoders to control parameter count in each view encoder with the intention of minimising imbalanced learning rate between view encoders. The image encoder is inspired by the AlexNet network (shown in fig. 3.5), but further simplified in consideration of limiting parameter count for unsupervised application. The CNN (shown in fig. 7.2) consists of 6 convolutional layers with batch normalisation and a non-linearity, and a fully-connected layer. The blocks $1 \times$ *2D conv*, $2 \times$ *2D conv*, and $3 \times$ *2D conv* in fig. 7.2 are followed by maximum pooling operations. The batch size used for GraphSAINT sampling for the MIMIC dataset is 512 unless otherwise stated. This is due to the large size of the dataset and the number of features.

7.3.1.1 Performance Potential

To assess performance potential, we run supervised classification, exchanging the graph attention embedding and DDC module for a fully connected output layer. Using ResNet-18 [35] (pre-trained on the ImageNet dataset [19])

we achieve a peak 73.0% validation accuracy with supervised classification proving existing — though limited — performance potential in this view. Our CNN (fig. 7.2) achieve 66.8% peak validation accuracy. For the vectorial data, we select *two* fully-connected layers $[1024, 512]$ as to allow for non-linear decision boundaries. Similarly, we perform supervised learning to assess learning potential by adding a prediction head. For the *vital signs* view, we achieve 73.2% accuracy, while the *lab tests* dataset achieve 86.2% supervised validation accuracy.

7.3.2 Secondary Datasets

For the Cora datasets, we set the graph embedding layers to $[512, 512]$ with single-layer view encoders $f_i : \mathbb{R}^{d_{(i)}} \rightarrow \mathbb{R}^{512}$ for views $i = 1, ..., V$. For CiteSeer, we use $[2000, 128]$ as our graph embedding layers and 2000 dimensions from our view encoders. For PubMed, we use graph embedding layers $[128, 64]$, and the dimensionality of our encoded representations are 256. All architectures correspond to undercomplete encoders, thus requiring the model to find efficient data selection solutions to retain important features. For Cora and CiteSeer, the batch size used in experimentation is equal to the size of the dataset unless otherwise stated. For PubMed, the batch size used is 512 due to the large dataset size.

7.3.3 Models & Parameters

For all our unsupervised experiments we use the full dataset for training, hyperparameter tuning, and evaluation. For our semi-supervised experiments, all metrics are computed using the labels $\{\boldsymbol{y}\} \setminus \{\boldsymbol{y}_{ss}\}$ ($\{\boldsymbol{y}\}$ and $\{\boldsymbol{y}_{ss}\}$ being the total set of labels and the set of semi-supervised labels, respectively) to assess generalisation capabilities of our model and avoid evaluation of a potentially over-fitted model. The set $\boldsymbol{y}_{ss}$ is drawn randomly at the beginning of each run and is used for batch loss if $\boldsymbol{y}_{ss} \neq \emptyset$, else $\mathcal{L}_{ss} = 0$.

7.3.3.1 Parameters

There are a large number of parameters to consider for the DRAGMVC model. For the sampling, we define the number of random walk steps by GraphSAINT as k_{gs}. This is set to equal the number of Graph attention layers as suggested by Zeng et al. [122]. The CoMVC model uses parameters δ and τ, being, respectively, the strength of the contrastive loss and a temperature parameter for the cosine similarity used in finding positive contrastive pairs [104]. The latter is set to be 0.1 in all our experiments. The DDC module uses parameter σ to determine the

standard deviation of the kernel function used to generate kernel $\boldsymbol{K}$ (see eq. 3.60) and is set to be $\sigma = 0.15$. The number of neurons for the hidden space in the DDC module is set to be 100 in all experiments. In addition, our model uses the following parameters: dropout probability $p_{dropout}$; the number of steps for our Markov chain prior k_{mc}. For experiments we set $k_{mc} = 2$ by the same reasoning as Zeng et al. [122]: in a random walk, the probability of a random walk starting from u having visited v after k_{mc} random steps is given by $M_{uv}^{(k_{mc})}$ (see section 5.2.1). Thus, with GraphSAINT sampling having walk length equal to k_{mc}, we remain within the full sample space for every graph attention layer. Finally, the two parameters found to be most likely to affect the results are δ and $p_{dropout}$, which will be covered in sections 7.4.2 and 7.4.4, respectively.

7.3.3.2 Models

We compare our model to MAGCN [13] (see section 4.2.3), being the only deep learning graph-based multi-view clustering for one view-invariant graph to the best of our knowledge. Models CoMVC & SiMVC [104] are utilised to evaluate the convenience of adding graph data. For all runs using CoMVC and SiMVC, we set up the view encoders identically to those of DRAGMVC. In addition, we will compare performance to the traditional clustering approaches k-means[9] and spectral clustering[10].

7.3.3.2.1 Recreated MAGCN In our results, we will refer to the recreated MAGCN results as MAGCN_{ours}. Given major deviations from the original paper, we alter the MAGCN source code [5] to comply with the description in [13] to the best of our understanding. We find a number of inaccuracies in the code provided. Firstly, the graph reconstrucion and reconstruction loss correspond to those of GATE [91] rather than the one described in [13] (in eqs. 4.22 & 4.23). Using sigmoid activation for graph convolution layers, the $-logsigmoid(\cdot) = -log\big(sigmoid(\cdot)\big)$ was implemented as an activation for the graph reconstruction. Secondly, the graph embedding layers do not share weights across views, nor have preceding view-specific encoders. In addition, we found that the distribution alignment was incorrectly cancelling out the second view embedding, essentially only aligning one view's distribution. In our MIMIC run we correct these deviations as well as extending the model's use to the multi-view case with $V > 2$ with the image encoder described in

9. Using the minimum inertia model from 10 runs using k-means++ initialisation. See section A.2.1.1.
10. Using k number of eigenvectors for spectral embedding and selecting the minimum inertia model from 10 runs.

7.3.1. Using this code we were able to closely recreate the results from the paper (see comparison in table 7.2).

7.3.4 Selection Criteria for Evaluation

We use various sets of patterns to determine the optimal experiments. This is due to two main factors: the metrics used by the competition and the sampling method used. For every model, we neglect the initial metrics (epoch 0) to assert that the performance is properly learned. All of our metrics (that are not global maximum) are taken at a single epoch that satisfies the conditions to be stated. In regards to comparative performance to other models we use the absolute maximum observed for all performance metrics as this is what has been done by our competition [13]. In table 7.8 comparing peak performance as well as stable performance, we determine both of which by the performance from each metric's peak mean. This is due to the difference in sampling method not allowing for a fair comparison of losses between the two models. The evaluation loss by GraphSAINT's random walk sampling will be subject to the potentially penalising effect of evaluation on different and unseen data (in early training as much of the randomly sampled data is not yet encountered), while MAGCN performs evaluation on the full dataset, yielding losses equal to those with which it is trained on. This results in differing evaluation losses that misrepresent the state of either model. With the exceptions of the previously mentioned results (tables 7.2 & 7.8), all measurements is measured at a single epoch yielding the lowest mean clustering loss. For DRAGMVC, this corresponds to the loss $\mathcal{L}_{ddc}$ as described in 4.2.2. The following describes which optima is used for all results:

- *Absolute maximum performance metric for each metric separately*: Table 7.2.
- *Maximum mean performance metric for each metric separately*: Table 7.8.
- *Minimum mean clustering loss for all metrics*: Tables 7.3, 7.4, 7.5, 7.6, 7.7, 7.9, 7.12, & 7.13.

7.4 Results

Using the above specifications we will present the results of our model on the graph multi-view datasets detailed in section 7.3. We will assess our proposed model in light of similar clustering models to assess its general capacity for deep graph-based multi-view clustering on well-documented data. In addition,

the section will cover the evaluation and tuning of the most impactful hyper-parameters, their effects, as well as model design choices made. In tackling the medical dataset, we will evaluate the clustering performance in a comparative setting, the effect of our k-NN graph, analysis of the prediction clusters, results of our model on pairs of views in the MIMIC dataset, and lastly, the semi-supervised approach using our model and its effect on prediction clusters and loss terms.

The following will be evaluated and analysed with regards to our proposed model:

1. Model performance on well-evaluated datasets.
2. Model performance on complex medical data, compared to established SOTA.
3. Model performance on complex medical data, compared to non-graph counterpart.
4. View mixing & cluster separability on complex data.
5. Effects of applying semi-supervised loss to performance.
6. Effects of applying semi-supervised loss to view mixing and clustering ability.
7. Effects on graph network over-smoothing.
8. Effects of using dropout.
9. Effects of the DDC encoder.

And the following will be analysed with regards to our constructed dataset:

1. Evaluating the dataset in light of the multi-view principles from chapter 4.
2. Diagnostic significance, separability, and observed effects from clustering.
3. Evaluating pairs of views with regards to performance and contrastive clustering ability.
4. Effects of using k-NN graph as opposed to the fully connected affinity

Table 7.2: Comparing performance metrics of deep learning graph-based multi-view clustering models on our secondary datasets.
* Results reported in [13].

Dataset		Cora				CiteSeer				PubMed			
Model	*Data*	ACC	NMI	ARI	$F1$	ACC	NMI	ARI	$F1$	ACC	NMI	ARI	$F1$
K-means	$\boldsymbol{X}_1$	0.448	0.264	0.165	0.346	0.573	0.306	0.278	0.528	0.597	0.322	0.284	0.582
Spectral clustering	$\boldsymbol{G}$	0.318	0.126	0.000	0.120	0.227	0.026	0.009	0.117	0.590	0.192	0.133	0.435
SiMVC	$\boldsymbol{X}_1$ & $\boldsymbol{X}_2$	0.386	0.170	0.130	0.271	0.470	0.224	0.210	0.310	0.659	0.295	0.256	0.670
CoMVC	$\boldsymbol{X}_1$ & $\boldsymbol{X}_2$	0.499	0.278	0.227	0.302	0.600	0.336	0.320	0.407	0.663	0.304	0.268	0.669
MAGCN$_{ours}$	$\boldsymbol{G}$, $\boldsymbol{X}_1$, & $\boldsymbol{X}_2$	0.704	0.497	0.434	0.470	0.689	0.407	0.430	0.403	0.716	0.347	0.342	0.696
MAGCN*	$\boldsymbol{G}$, $\boldsymbol{X}_1$, & $\boldsymbol{X}_2$	0.751	**0.598**	0.532	—	0.711	0.458	0.462	—	0.691	0.331	0.321	—
Ours	$\boldsymbol{G}$, $\boldsymbol{X}_1$, & $\boldsymbol{X}_2$	**0.762**	0.586	**0.568**	**0.640**	**0.724**	**0.478**	**0.490**	**0.568**	**0.750**	**0.349**	**0.390**	**0.720**

graph.

7.4.1 Comparative Results

To begin, we present the comparative results of our secondary datasets. For context we include two basic clustering mthods: K-means for the primary feature matrix $\boldsymbol{X}_1$, and spectral clustering for the graph $\boldsymbol{G}$. For the multi-view (without graph data) approach, we evaluate the datasets using the SiMVC and CoMVC models. For the graph-based multi-view clustering model, we employ the MAGCN model (being the only comparable model to ours given the conditions defined in the introduction, section 1.2) by the results presented in their paper, referred to as MAGCN*, as well as our recreated results (as described in section 7.3.3.2), referred to as MAGCN$_{ours}$.

Table 7.2 displays the comparative results of our method, showing the best results in bold. DRAGMVC outperforms other comparable methods on every secondary dataset. We suggest that the added information of direct sample-to-sample representation aligning yields a more stable and nuanced embedding space: allowing graph embedding layers to extract more data than the more common distribution-wise contrastive alignment seen in MAGCN. We will further analyse this notion in tackling the complex MIMIC dataset and display and analyse its hidden spaces.

7.4.2 Using Dropout

As a measure to improve the robustness of our model, a dropout layer is added to the graph attention layers, adding regularisation and lessens over-dependencies on certain neurons. In addition, it has been speculated that the regularisation effect borne of dropout induces natural clustering in neural networks [63]. With this phenomena in mind, we analyse the effects of applying dropout with probability p in the model's graph embedding layers. The dropout

Table 7.3: Comparing our model by dropout (taking n random samples for each setup). Batch size N_{cora}.

		ACC			NMI			ARI			$F1$		
Dropout probability p		Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.
Cora dataset	$n = 150$: $p = 0.0$	0.618	0.058	(0.609, 0.627)	0.482	0.032	(0.477, 0.487)	0.404	0.052	(0.396, 0.412)	0.205	0.160	(0.179, 0.231)
	$n = 150$: $p = 0.2$	0.624	0.061	(0.614, 0.634)	0.490	0.033	(0.485, 0.495)	0.411	0.049	(0.403, 0.419)	0.205	0.146	(0.182, 0.228)
	$n = 150$: $p = 0.5$	**0.647**	0.050	(0.639, 0.655)	**0.503**	0.027	(0.499, 0.507)	**0.431**	0.044	(0.424, 0.438)	**0.207**	0.135	(0.185, 0.229)
MIMIC dataset	$n = 123$: $p = 0.0$	0.609	0.038	(0.602, 0.616)	0.042	0.023	(0.038, 0.046)	0.052	0.030	(0.047, 0.057)	0.606	0.039	(0.599, 0.613)
	$n = 150$: $p = 0.2$	0.616	0.042	(0.609, 0.623)	0.054	0.028	(0.050, 0.058)	0.059	0.035	(0.053, 0.065)	0.609	0.052	(0.601, 0.617)
	$n = 150$: $p = 0.5$	**0.624**	0.034	(0.619, 0.629)	**0.061**	0.026	(0.057, 0.065)	**0.065**	0.031	(0.060, 0.070)	**0.619**	0.041	(0.612, 0.626)

Table 7.4: Comparing DDC module encoder with respective 95% confidence intervals.

Cora Dataset	ACC			NMI			ARI			$F1$		
DDC Encoder	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.
$n = 150$: Linear	0.620	0.056	(0.611, 0.629)	0.493	0.031	(0.488, 0.498)	0.412	0.048	(0.404, 0.420)	**0.227**	0.144	(0.204, 0.250)
$n = 150$: GraphAttn.	**0.647**	0.050	(0.639, 0.655)	**0.503**	0.027	(0.499, 0.507)	**0.431**	0.044	(0.424, 0.438)	0.207	0.135	(0.185, 0.229)

will be employed following the linear layer (eq. 5.6) as well as on the generated graph attentions $\tilde{G}_h^{(r)}$ (eq. 5.10), similarly to the dropout utilised in the GAT model [107].

Seeing table 7.3, increasing dropout does appear to significantly improve every metric (except for $F1$ score) for the Cora and MIMIC datasets. Namely, $p = 0.50$ significantly increases the mean performance in ACC, NMI, and ARI in both datasets, suggesting that dropout does improve model performance. We may assume that, unless otherwise stated, dropout is implemented using probability $p = 0.50$ in all following experiments.

7.4.3 DDC Module — Graph Convolution Encoder

As previously described, the encoder used for the DDC module is set to a graph attention convolution layer, exchanging the original fully connected layer. The final output layer of the DDC is unchanged. This switch could help add relational information into the fused representations, avoiding the restrictions in preceding graph embedding layers of having to extract commonality across views (due to the weight-sharing across views). We compare our model to a similar model using a linear DDC layer on the Cora dataset.

We find that altering the DDC encoder provide a significant advantage to the clustering performance (see table 7.4) in the ACC, NMI, and ARI metrics. It does not appear to be significantly affecting $F1$-score. Considering the significant performance improvement we employ the graph attention convolution in all following experiments.

7.4.4 Effects of the Contrast Parameter δ

When tuning the contrastive loss strength, we consider values $\delta = 0.1, 1, 3, 6, 10$. We evaluate the contrastive strength significance on both the MIMIC dataset and the Cora dataset. This is reasoned by the intuition that information extraction may be sourced from different loss terms depending on dataset complexity. For highly complex datasets such as our *pleural effusion/not pleural effusion* dataset we expect that the model will depend highly on contrastive loss to learn proper feature extraction. To evaluate the intervals of $\mathcal{L}_{contrast}$ equally, we divide the intervals by δ. This should be unproblematic as all values $\mathcal{L}_{contrast}$ are simple products of δ, linearly reducing as we know that $\mathbb{E}[X/c] = (1/c)\mathbb{E}[X]$ and the contrastive loss is scaled similarly (see section 4.2.2).

Table 7.5 shows the results of the various values of δ. We note that runs with $\delta > 0.1$ for the both datasets tends to yield higher overall performance in ACC, NMI, and ARI (and $F1$ for the MIMIC dataset). Purely by considering the number of significantly superior performance metrics we observe that $\delta \in \{3.0, 6.0, 10.0\}$ generally outperform $\delta = \{0.1, 1.0\}$ in most metrics.

We consider the null hypothesis that δ does not affect mean performance in any metrics. The alternate hypothesis is that at least a single mean performance metric is significantly different from the other means. We set up an one-way ANOVA test with $\alpha = 0.05$ for the Cora dataset yielding p-values < 0.0001, < 0.0001, < 0.0001, and 0.8453, for ACC, NMI, ARI, and $F1$, respectively. Thus, we conclude that the strength of the contrastive loss is significantly different for $\delta \in \{0.1, 1, 3, 6, 10\}$ for the metrics ACC, NMI, and ARI. Now, for the MIMIC dataset, using the same alternate hypothesis, we get p-values < 0.0001, < 0.0001, < 0.0001, and < 0.0001. Now in defining values of p we set up a test to explore whether our optimal set $\delta \in \{3, 6, 10\}$ is significantly different. Similarly, one-way ANOVA yields 0.7590, 0.9599, 0.7830, and 0.7495 for Cora, suggesting that there is no significant difference between the three contrastive strengths. For MIMIC, a similar test yields 0.2391, 0.0664, 0.1902, and 0.1164, also proving insignificant. Thus, we employ the value $\delta = 6.0$ for the secondary dataset, and $\delta = 10.0$ for the MIMIC dataset.

The results of table 7.5 contradict our hypothesis as both Cora and MIMIC proved to benefit from increased contrastive weight to a similar digree. For Cora all sets of runs had minimum mean clustering losses at epoch 40. For the MIMIC dataset, however, the optimal epochs were at 35, 10, 15, 25, and 30 (from top to bottom in the table). This makes contrastive loss comparisons for the latter dataset more difficult, as contrast may optimise at a different rate

Table 7.5: Comparing the effects of δ on two datasets of differing levels of complexity.

		ACC			NMI			ARI			$F1$			Loss			
Finding δ		Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean $\mathcal{L}_{contrast}$	$\mathcal{L}_{contrast}/\delta$, C.I.	Mean $\mathcal{L}_{ddc}$	$\mathcal{L}_{ddc}$, C.I.
Cora dataset	$n = 75$: $\delta = 0.1$	0.611	0.052	(0.599, 0.623)	0.484	0.024	(0.479, 0.489)	0.399	0.041	(0.390, 0.408)	**0.220**	0.132	(0.190, 0.250)	0.069 ($\mathcal{L}_c/\delta = 0.686$)	(0.682, 0.690)	0.762	(0.751, 0.773)
	$n = 75$: $\delta = 1.0$	0.622	0.050	(0.611, 0.633)	0.497	0.025	(0.491, 0.503)	0.416	0.041	(0.407, 0.425)	0.208	0.151	(0.174, 0.242)	0.343	(0.322, 0.364)	0.782	(0.769, 0.795)
	$n = 75$: $\delta = 3.0$	0.642	0.050	(0.631, 0.653)	**0.507**	0.025	(0.501, 0.513)	0.426	0.039	(0.417, 0.435)	0.209	0.147	(0.176, 0.242)	0.935 ($\mathcal{L}_c/\delta = 0.312$)	(0.287, 0.336)	0.823	(0.809, 0.837)
	$n = 75$: $\delta = 6.0$	0.640	0.049	(0.629, 0.651)	0.506	0.023	(0.501, 0.511)	0.430	0.038	(0.421, 0.439)	0.207	0.143	(0.175, 0.239)	2.053 ($\mathcal{L}_c/\delta = 0.342$)	(0.318, 0.367)	0.862	(0.847, 0.877)
	$n = 75$: $\delta = 10.0$	**0.646**	0.052	(0.634, 0.658)	0.506	0.026	(0.500, 0.512)	**0.430**	0.044	(0.420, 0.440)	0.193	0.131	(0.163, 0.223)	3.303 ($\mathcal{L}_c/\delta = 0.330$)	(0.310, 0.351)	0.874	(0.862, 0.886)
MIMIC dataset	$n = 75$: $\delta = 0.1$	0.599	0.042	(0.589, 0.609)	0.040	0.024	(0.035, 0.045)	0.045	0.030	(0.038, 0.052)	0.594	0.044	(0.584, 0.604)	0.266 ($\mathcal{L}_c/\delta = 2.656$)	(2.568, 2.744)	1.063	(1.025, 1.101)
	$n = 75$: $\delta = 1.0$	0.620	0.028	(0.614, 0.626)	0.049	0.021	(0.044, 0.054)	0.060	0.025	(0.054, 0.066)	0.614	0.038	(0.605, 0.623)	3.412	(3.367, 3.457)	0.938	(0.914, 0.962)
	$n = 75$: $\delta = 3.0$	**0.630**	0.030	(0.623, 0.637)	**0.061**	0.023	(0.056, 0.066)	**0.071**	0.029	(0.064, 0.078)	**0.628**	0.032	(0.621, 0.635)	8.964 ($\mathcal{L}_c/\delta = 2.988$)	(2.859, 3.117)	0.985	(0.959, 1.011)
	$n = 75$: $\delta = 6.0$	0.627	0.034	(0.619, 0.635)	0.059	0.026	(0.053, 0.065)	0.068	0.032	(0.061, 0.075)	0.622	0.041	(0.613, 0.631)	16.865 ($\mathcal{L}_c/\delta = 2.811$)	(2.624, 2.997)	1.013	(0.977, 1.049)
	$n = 75$: $\delta = 10.0$	0.621	0.035	(0.613, 0.629)	0.052	0.025	(0.046, 0.058)	0.062	0.031	(0.055, 0.069)	0.615	0.041	(0.606, 0.624)	27.028 ($\mathcal{L}_c/\delta = 2.703$)	(2.510, 2.896)	1.010	(0.971, 1.049)

Table 7.6: Results of replacing the number of layers in our graph encoder while increasing the number of steps in our Markov prior. We use batch size 512 to get the desired sample space using GraphSAINT with $k_{gs} = 4$.

Cora Dataset	ACC			NMI			ARI			$F1$		
	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.
$n = 150$: $L = 1$ & $k_{mc} = 4$	**0.649**	0.053	(0.641, 0.657)	**0.524**	0.025	(0.520, 0.528)	**0.452**	0.041	(0.445, 0.459)	**0.236**	0.163	(0.210, 0.262)
$n = 150$: $L = 2$ & $k_{mc} = 2$	0.630	0.057	(0.621, 0.639)	0.505	0.026	(0.501, 0.509)	0.422	0.043	(0.415, 0.429)	0.211	0.138	(0.189, 0.233)
$n = 150$: $L = 4$ & $k_{mc} = 1$	0.612	0.051	(0.604, 0.620)	0.474	0.025	(0.470, 0.478)	0.395	0.039	(0.389, 0.401)	0.199	0.136	(0.177, 0.221)

than the clustering loss. Analysing the losses of Cora, however, we assess that only $\delta = 0.1$ is significantly higher than others. We reason that this suggests that the full contrastive potential is not utilised. The clustering loss does seem to be affected by the increase in δ, observing significantly higher clustering losses for higher-valued δ. Considering the insignificant difference between corresponding setups in normalised contrastive loss, we suggest that this may be due to varying convergence times: that clustering optimisation adapts to the contrastive optimisation rather than the opposite, thus limiting the loss landscape of the DDC.

7.4.5 Reducing Over-smoothing

Having made the hypothesis that the Markov chain attention may lessen the effect for deep GCNs causing over-smoothing [76], we will perform experiments to strenghten this hypothesis. Considering our GraphSAINT sampling approach (as described in section 3.3.2), we know that our batch space is within a set number of steps k_{gs} from randomly sampled nodes. We have previously set this value to equal L — the number of graph attention embedding layers in our network — by the reasoning that L layers will be able to include every sampled point in batch space. However, using our Markov chain prior, we may alter the number of layers required to span k_{gs} neighbours[11] through the parameter k_{mc}, thus potentially reduce the significance of over-smoothing as is common in deep GCNs. The experiment is therefore set using $k_{gs} = 4$, requiring $L = 4$ layers to cover for the common GCN. We run our model using $k_{mc} = 4$ and $L = 1$, $k_{mc} = 2$ and $L = 2$, and $k_{mc} = 1$ and $L = 4$ — all corresponding to a potential reach of 4 steps from any random node. If our hypothesis is correct, we expect there to be a significant performance difference between the three setups, favouring shallower models.

Table 7.6 shows that fewer layers with higher markov chain connections yields significantly better results for the Cora dataset. The table may be interpreted in a number of ways. As covered, the sample space and the receptive field of the graph encoder are equal in all runs, leading to one assumption that our Markov

11. The number of random walk steps taken by our sampling method.

Table 7.7: Comparing the our graph-induced model with its CoMVC and SiMVC counterparts using the full MIMIC dataset with batch size 512 and no dropout.

MIMIC Dataset	ACC			NMI			ARI			$F1$		
Graph	Mean	SD	C.I	Mean	SD	C.I	Mean	SD	C.I	Mean	SD	C.I
n = 316: SiMVC (no graph)	0.545	0.028	(0.542, 0.548)	0.008	0.008	(0.007, 0.009)	0.011	0.011	(0.010, 0.012)	0.502	0.053	(0.496, 0.508)
n = 150: CoMVC (no graph)	0.586	0.028	(0.582, 0.590)	0.024	0.015	(0.022, 0.026)	0.032	0.020	(0.029, 0.035)	0.492	0.091	(0.477, 0.507)
n = 123: Our model	**0.609**	0.038	(0.602, 0.616)	**0.042**	0.023	(0.038, 0.046)	**0.052**	0.030	(0.047, 0.057)	**0.606**	0.039	(0.599, 0.613)

prior reduces the need for deeper networks for the sake of increasing the graph receptive field, while yielding superior clustering results. It is hard to uncover the direct causal effect for this, but we speculate that the over-smoothing caused by repeated graph convolutions [76] may be lessened as attention is applied over the full receptive field, rather than the set of immediate neighbours. It is also worth noting that for each graph convolution layer, every node is given as a fusion of all its attentions. Thus, the receptive field will lose strength as L increases[12], while our model allows for stronger connections to more distant neighbours.

7.4.6 Clustering on Real-world Medical Data

7.4.6.1 Comparative Results

We will now cover the results from the constructed medical dataset (detailed in section 6). This section aims to answer questions regarding our DRAGMVC's ability to deal with complex, multi-modal data. We will perform this evaluation by comparing our proposed method with previous methods, graph-based and not, and analyse the resulting clusters. In addition, we aim to explore the contents of the MIMIC dataset by evaluating pairs of views, and finally, we wish to explore the effects and potential of applying a semi-supervised loss term to our model on our dataset. To begin, we evaluate the effects on performance by utilising graph attention convolutions in a representation-aligned multi-view clustering model. We run the MIMIC dataset on the SiMVC, CoMVC, and DRAGMVC purely to establish the difference in clustering accuracy. Subsequently, we will evaluate the results in comparison to another graph-based MVC model to establish its prediction performance compared to a similar approach.

Table 7.7 display a clear improvement by DRAGMVC in peak performance for all metrics. For mean values, DRAGMVC increases or performs similarly to Si-/CoMVC in all metrics except for macro-$F1$. We do note that better metrics may not directly reflect a superior clustering performance as the clustering may be high in quality, yet unrelated to our predefined labels. We will explore

12. Each attention layer will attend to a set of neighbours by averaging its feature vectors weighted by its attentions. Thus, the expanding receptive field will extend through a fraction of the fused feature vector from the previous layer.

Table 7.8: Comparing MAGCN and our model on the MIMIC sample dataset

Mini-MIMIC	ACC				NMI				ARI				*F*1			
Batch size 1747	Max.	Mean	SD	C.I.	Max.	Mean	SD	C.I.	Max.	Mean	SD	C.I.	Max.	Mean	SD	C.I.
n = 54: MAGCN$_{ours}$	**0.731**	0.553	0.032	(0.544, 0.562)	**0.244**	0.011	0.047	(-0.002, 0.024)	**0.212**	0.007	0.034	(-0.002, 0.016)	**0.708**	0.348	0.066	(0.330, 0.366)
n = 100: Our model	0.649	**0.576**	0.036	(0.569, 0.583)	0.067	**0.021**	0.015	(0.018, 0.024)	0.088	**0.028**	0.021	(0.024, 0.032)	0.649	**0.574**	0.038	(0.567, 0.581)

this motion more closely in section 7.4.6.3 by dissecting the clusterings made by both DRAGMVC and CoMVC.

Due to memory issues with MAGCN from to a lack of mini-batch sampling, the comparison with MAGCN must be performed on a smaller dataset. We use GraphSAINT to create a dataset of 1 747 samples from the full MIMIC dataset. The dataset will be referred to as *Mini-MIMIC* to separate it from the full dataset. We predict that there are a number of dissimilarities between our method and previous comparable methods. Sample-to-sample representation alignment could improve stability in fusion space and thus affect predictions as compared to solely using distribution-wide view alignment as in MAGCN. The reason for this assumption is that sample-to-sample alignment could push representations to be placed in relation to other samples based on their similarities, i.e. a representation in cluster space could contain different information towards one end of the cluster as compared to the other end. We reason that the positions of samples in clusters by distribution-wide alignment will be more random, allowing for a larger degree of variation within the cluster space. This intuition is at the core when reasoning what effect contrast parameter δ has on complex datasets (see section 7.4.4). Also, provided that sample-to-sample representation alignment could allow for improved encoder training, we may observe differences in terms of general accuracy.

Table 7.8 shows interesting results in that MAGCN yields higher peaks than our model, yet DRAGMVC shows significantly better overall performance throughout. We interpret the results as sample-to-sample alignment providing more stable grounds for learning as compared to the distribution-wise alignment. However, we observe that the MAGCN model is much more sensitive to deviations. This strengthens our hypothesis that within-cluster variation may be higher for the MAGCN model, as random variations could allow for more optimal decision boundaries by purely random deviation. We observe during training that the clusters of the MAGCN model regularly collapses, explaining the exceedingly poor mean performance in table 7.8. We reason that this is another consequence of the instability hypothesised by their alignment. The reasoning being that their lacking contrastive loss do not allow for sufficient feature extraction for highly complex data.

Table 7.9: Comparing the full graph with the k-NN graph on the MIMIC dataset using our model. No dropout is applied to allow for full graph utilisation.

MIMIC Dataset	ACC			NMI			ARI			$F1$		
Our model (dropout $p = 0.5$)	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.
$n = 150$: Full graph	0.600	0.032	(0.595, 0.605)	0.034	0.018	(0.031, 0.037)	0.044	0.025	(0.040, 0.048)	0.594	0.042	(0.587, 0.601)
$n = 150$: k-NN graph	**0.624**	0.034	(0.619, 0.629)	**0.061**	0.026	(0.057, 0.065)	**0.065**	0.031	(0.060, 0.070)	**0.619**	0.041	(0.612, 0.626)

Table 7.10: Optimal DRAGMVC runs on MIMIC by DDC (runs 1, 2, & 3) and contrastive (runs 3, 5, & 4) loss.

MIMIC dataset	Loss		ACC	NMI	ARI	$F1$
	$\mathcal{L}_{ddc}$	$\mathcal{L}_{contrast}$				
Run 1	**0.775**	7.578	0.528	0.002	0.003	0.526
Run 2	0.778	12.705	0.644	0.062	0.083	0.642
Run 3	0.778	**5.790**	0.653	0.072	0.094	0.652
Run 4	0.782	7.072	**0.670**	**0.088**	**0.115**	**0.668**
Run 5	0.786	6.314	0.572	0.015	0.020	0.569

7.4.6.2 Effects of k-NN graph

Given that both graph attention and sampling are based on non-zero values of our graph, having a sparse graph, rather than the full graph computed in section 6.4, may allow for easier recognition of significant neighbours. A $k = 140$ nearest neighbour graph will significantly reduce the number of possible attention subjects down from $N = 3\,520$.

From table 7.9, we see that the k-NN graph described in section 6.4 significantly outperforms the full graph in every metric. Thus, we implement the model using the k-NN graph dataset in every following experiment.

7.4.6.3 Analysing the Clusters

In order to detail the clustering performance on our proposed MIMIC dataset, we should display its capabilities in terms of view mixing, separability, and diagnostics. We will perform analysis on a small selections of runs, allowing for deeper analysis of the qualities mentioned. In our analysis, we will employ the tools described in section 7.2. We select the 3 runs that minimises overall DDC loss

$$\mathcal{L}_{ddc} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3,$$

referred to as runs 1, 2, & 3; as well as 3 runs that minimises contrastive loss $\mathcal{L}_{contrast}$, being runs 3, 5, & 4, respectively.

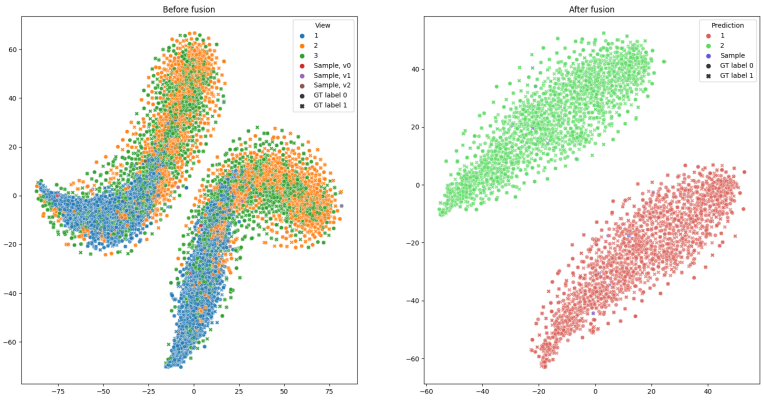
Details of the runs are shown in table 7.10. We will analyse and compare these five runs with two runs for CoMVC from table 7.2 minimising the DDC and contrastive loss, having $\mathcal{L}_{ddc} = 0.833$ and $\mathcal{L}_{contrast} = 6.394$, respectively. For both loss terms we observe that our model yields far lower minimum loss values than CoMVC — possibly suggesting that our graph may improve on both the natural cluster structure as well as the resulting view mixing. The CoMVC runs will be referred to as *CoMVC DDC* and *CoMVC Contrast*, respectively.

7.4.6.3.1 Fusion Plots The runs in fig. 7.3 all display a high level of mixing of views — particularly of views *vital signs* (view 1) and *lab tests* (view 2). In addition, the pre-fusion space display an indication of natural groupings in having full separation in run 1, and a weak connection in runs 2 and 3. For the fusion space, we observe well-separated clusters in all runs. Ideally we would like to see full separation in pre-fusion space for all runs as this would indicate that there are certain underlying trends of groupings that are fully recognisable in all views.

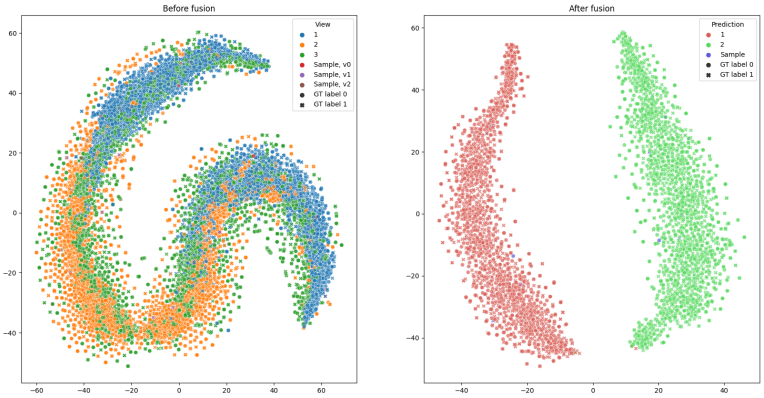
For the optimum runs in terms of contrastive loss (fig. 7.4) we observe a slight increase in the image view (view 0) spread within pre-fusion clusters. Ideally, we would prefer all views to cover the entirety of the space defining the pre-fusion clusters — thus implying that the views are properly able to recognise some level of commonality sample-to-sample. We will later go over our reasoning as to why this is not the case. Firstly, for the sake of comparison, we look at the optimum runs' fusion plots using CoMVC (see fig. 7.5).

The difference between figs. 7.3 & 7.4 and 7.5 is evident. The fusion plots by CoMVC show far less separability in fused representation and no separability in pre-fusion space at all. Suggesting that the model struggle to identify and learn any commonality for views. In addition, we observe far inferior view mixing for all CoMVC runs' views — even for the most optimal contrastive loss (fig. 7.5b). Thus, we state that the graph information appears to add a very significant amount of contextual information to the data pool, allowing for a far superior ability to extract commonality and separate clusters.

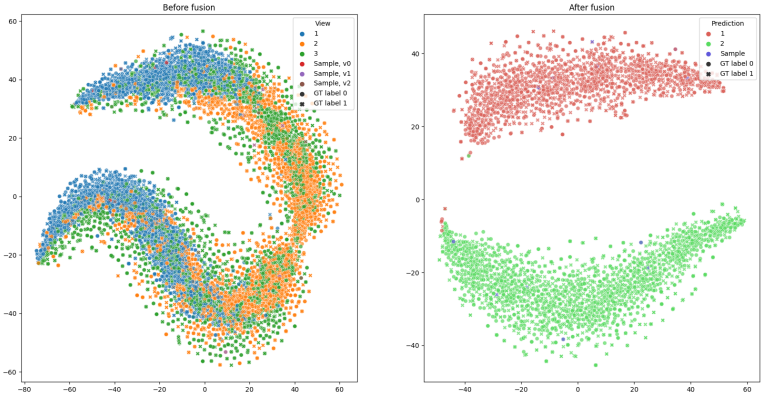
7.4.6.3.2 X-ray Samples Figure 7.6 displays samples drawn from their respective pre-fusion plots 7.3a & 7.4a (where they are illustrated by red data points). Lacking the subject knowledge required to properly analyse X-rays, we restrict the analysis to strictly surface-level observations. Due to the high variation of quality in real-world data, images appear to differ greatly by patients' physical differences in addition to the image angle and contrast. We reason



(a) Run 1 fusion plot.

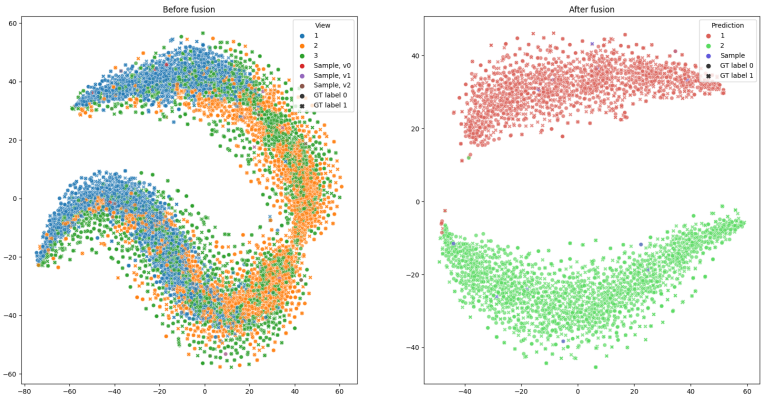


(b) Run 2 fusion plot.

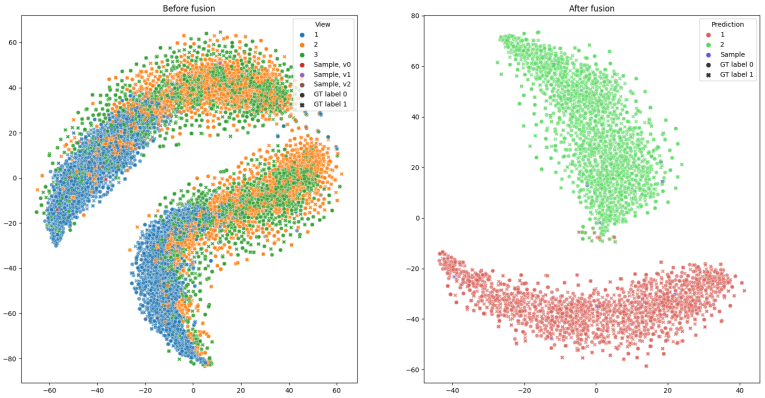


(c) Run 3 fusion plot.

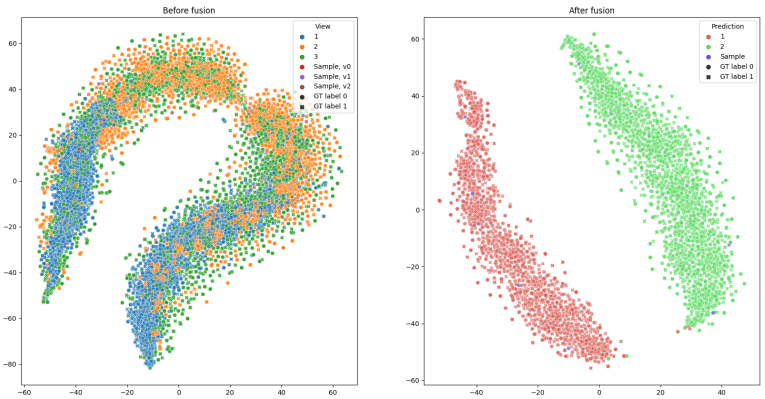
Figure 7.3: Runs by DDC loss fusion plots (dimensionality reduced by t-SNE) from best (top) to worst (bottom).



(a) Run 3 fusion plot.

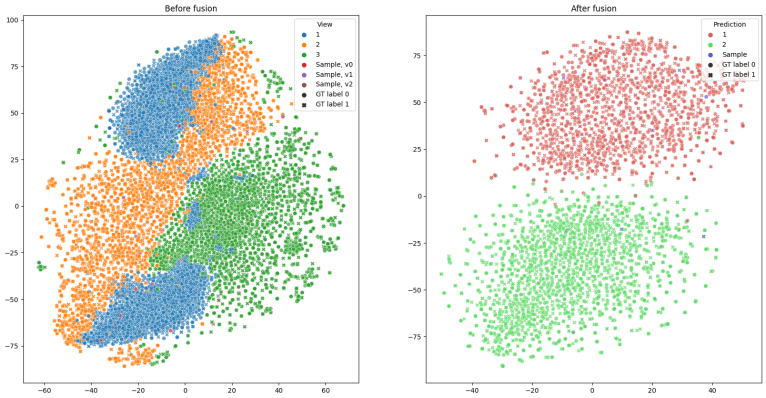


(b) Run 5 fusion plot.



(c) Run 4 fusion plot.

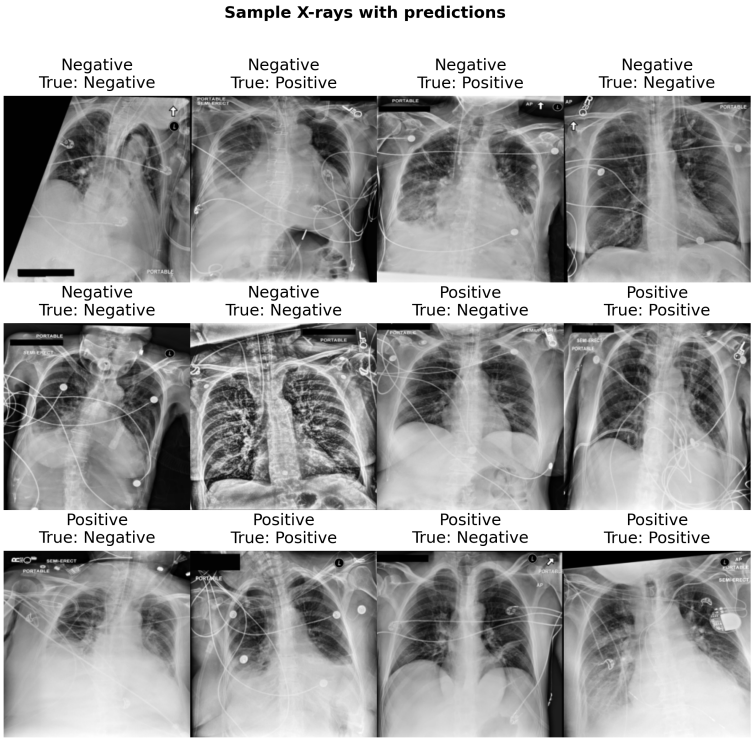
Figure 7.4: Runs by contrastive loss fusion plots (dimensionality reduced by t-SNE) from best (top) to worst (bottom).



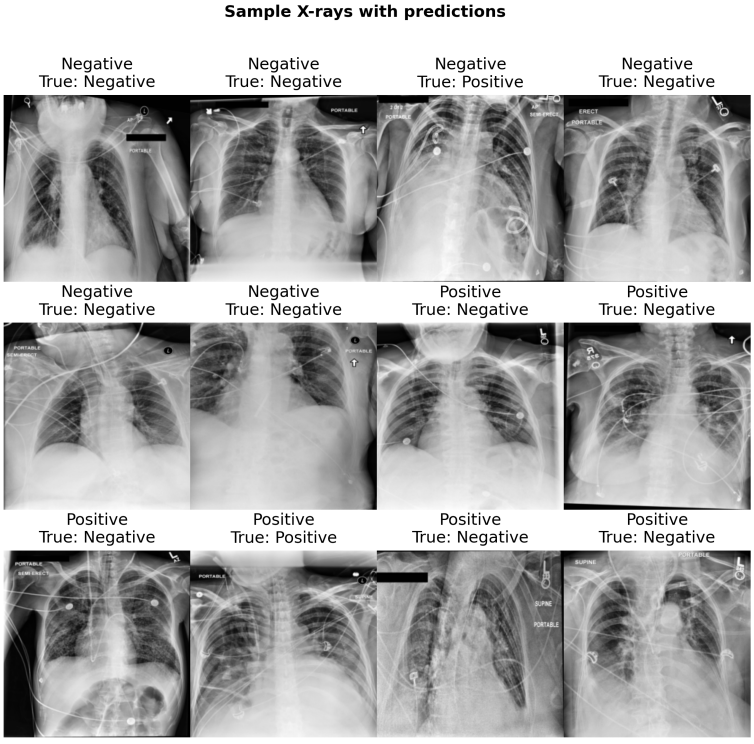
(a) *CoMVC DDC* fusion plot.

(b) *CoMVC Contrast* fusion plot.

Figure 7.5: Best CoMVC runs fusion plots (dimensionality reduced by t-SNE).

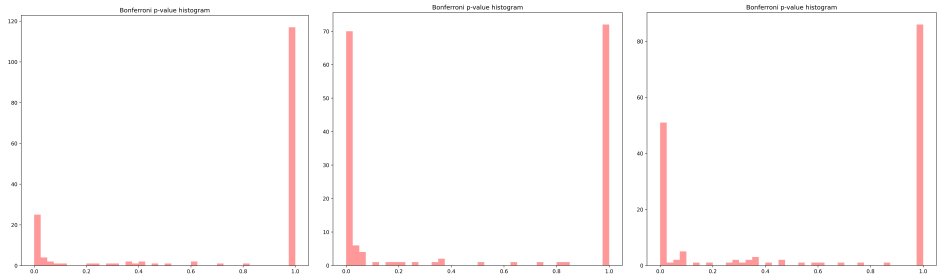


(a) Run 1 samples.

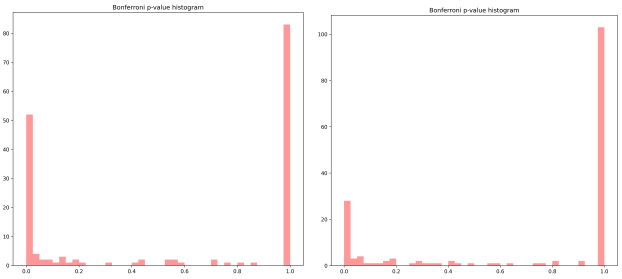


(b) Run 3 samples.

Figure 7.6: X-rays from the two optimal runs from table 7.10.



(a) Run 1 Bonferroni-adjusted p-value histogram. **(b)** Run 2 Bonferroni-adjusted p-value histogram. **(c)** Run 3 Bonferroni-adjusted p-value histogram.



(d) Run 4 Bonferroni-adjusted p-value histogram. **(e)** Run 5 Bonferroni-adjusted p-value histogram.

Figure 7.7: Bonferroni-adjusted p-value histogram.

Table 7.11: Proportion of significant diagnoses post-Bonferroni correction.

Run	$\mathbb{E}_{run}\left[P(\text{sign. diag. diff.}; \alpha = 0.05)\right]$
Run 1	0.176
Run 2	0.461
Run 3	0.315
Run 4	0.339
Run 5	0.188

that views lacking a sufficient number of informative samples will struggle with representation alignment, yielding bundles of a single view within a cluster, as may be the cause in the contrastive run fusion plots (fig 7.5) and to a lesser degree in the DRAGMVC fusion plots (figs. 7.3 & 7.4). One interpretation of the main challenge of view mixing is therefore that of insufficient images in terms of quantity and/or quality. We will explore this further in section 7.4.6.4.

7.4.6.3.3 Diagnostics We may assert that the clusters *are* related to a set of diagnoses. This is apparent by the left-shift as compared to the expected flat p-value distribution of a randomly generated and fully uncorrelated set (standard p-value histograms are displayed in the appendix, fig. A.2). From figure

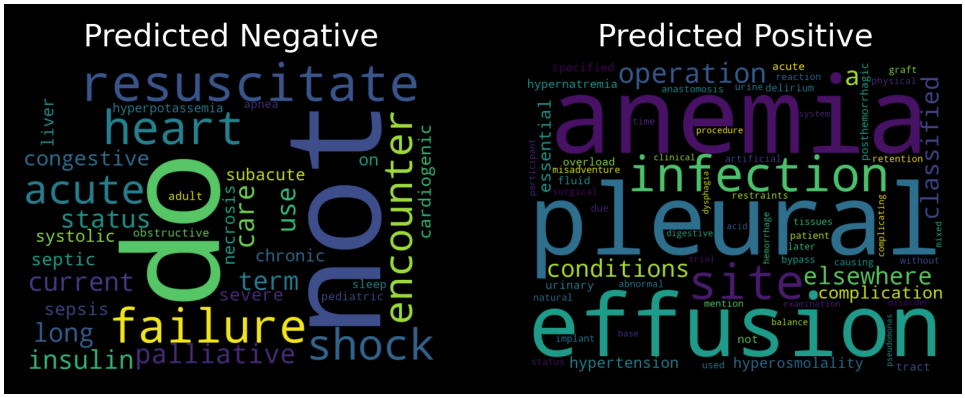
7.7 we observe that there is a significant degree of separation in the number of (post-correction) significant diagnoses for each run, which is strengthened by the significant proportions shown in table 7.11. This suggests that the model recognises data patterns that are related to a set of diagnoses. We should note, however, that it should *not* be interpreted as the model recognising any of these diagnoses specifically. What is recognised by the model could be diagnosis-related correlations and not exact features of any statistically significant diagnosis. Run 1 has the lowest number significant diagnoses with 17.6% of all diagnoses in the sample being significantly different in the two clusters (where $\alpha = 0.05$). Run 1 also has the highest degree of separability as seen in fig. 7.3a. This may suggest a high degree of independence in the variables recognised across views. The same is observed for run 5 (with the second lowest proportion of significant diagnoses) having the second most pronounced pre-fusion separability (fig. 7.4b). Clusters with correlated diagnoses will likely yield representations that appear more diffuse due to their many interpretations and unclear separation boundaries. Examples of this are: model 2 from the DDC selection (figs. 7.3b & 7.7b) and run 4 from the contrast selection (figs. 7.4c & 7.7d).

Figure 7.8 displays word clouds for all DRAGMVC sample runs. There are similarities in clusters based on recurring words. Words *hemorrhage* and *cerebral* appear in all negative clusters except for run 1. This imply that there may be a strong natural grouping for these diagnoses is our dataset. Similarly, *acute kidney failure* appear to be words with strong natural groups, seeing as they appear to be common words for the positive prediction clusters for runs 2, 3, and 4. Finding these natural groupings indicate that the model likely possesses an ability to recognise strong underlying patterns related to patient diagnoses.

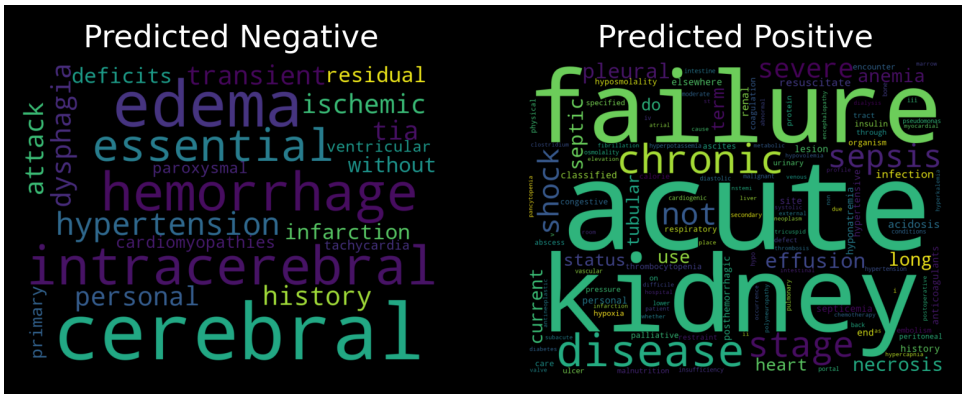
7.4.6.4 Exploring Dataset Partitions

In order to evaluate the contribution of each view to the total model performance, we compare results from every pair of views in the MIMIC dataset, and analyse how they compare to the complete set. By comparing subsets, we aim to discover which views fulfil and maximise the two multi-view principles: the consensus principle and the complementary principle, as covered in chapter 4. If any view were to contain redundant and/or non-infomative data, we expect to see worse results for the complete set of views as compared to the pair not containing the subpar view. Another potential drawback of using 3 views as compared to 2 is the weakened contrastive strength. The force of any single view embedding on its contrastive counterparts will weaken as any added views will provide an additional force on the points. In addition to having less

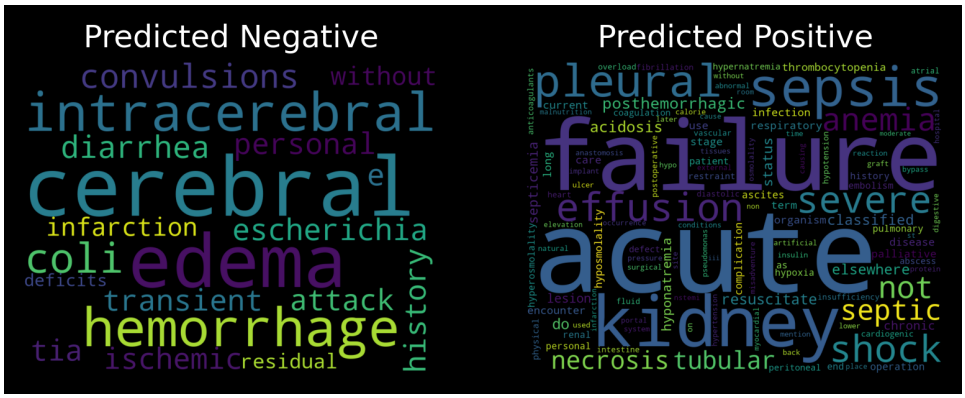
geometric significance, contrastive loss is scaled by the minimum fusion weight, thus further reducing the loss significance for any given view for high view counts. In combination with the multi-view principles, this will result in added



(a) Word cloud of run 1 diagnoses.

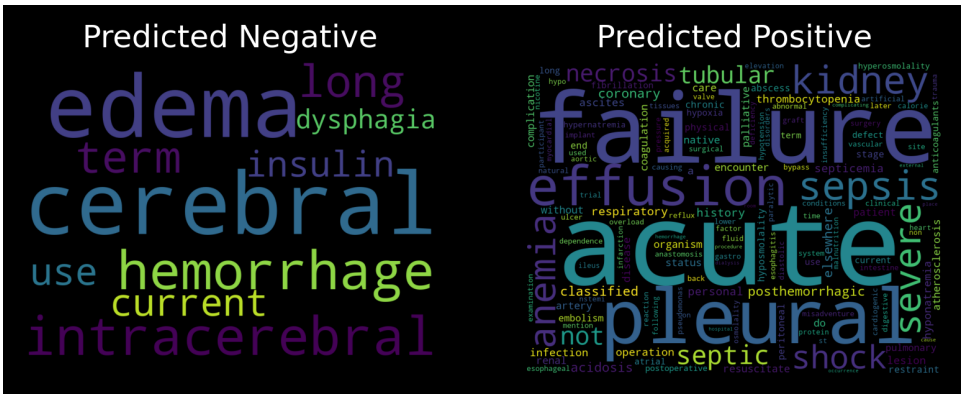


(b) Word cloud of Run 2 diagnoses.

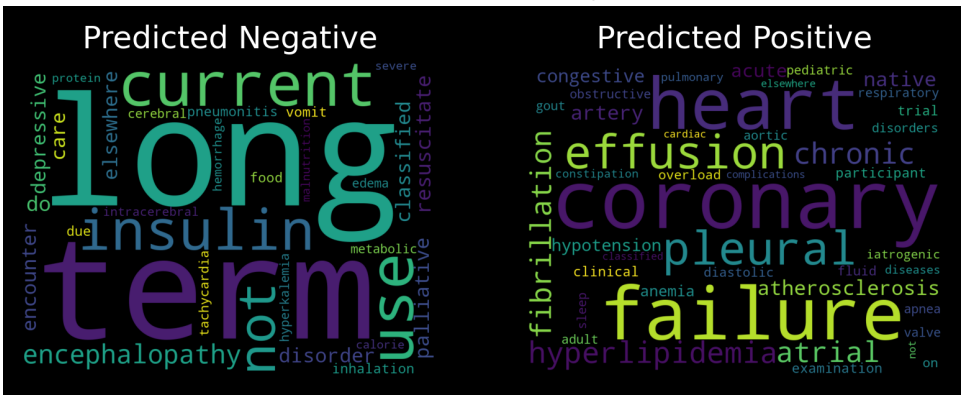


(c) Word cloud of Run 3 diagnoses.

Figure 7.8: Word cloud of the most common diagnoses in the output clusters. Figure continues on the next page.



(d) Word cloud of Run 4 diagnoses.



(e) Word cloud of Run 5 diagnoses.

Figure 7.8: (Cont.) Word cloud of the most common diagnoses in the output clusters.

noise to the contrastive module — making clean clustering increasingly hard as contrative strength lessen and noise increase.

From table 7.12 and fig. 7.9, we may suggest two interpretations: (i) the *vital signs* view does not fulfil the multi-view principles and is solely adding redundant noise to the data; (ii) added information in *vital signs* does not out-weigh the loss of contrastive force between the two remaining, stronger views. Finally, we reason that the encoders required for various views may differ greatly in complexity. Therefore — with 3 views — the optimisation time for view encoders may be unbalanced, allowing the two easier views to optimise by each other, while the most complex view lags behind limited by the abilities of its encoder. We hypothesise that the resulting effect of the uneven convergence times is that the image view may develop sufficiently to provide representations that provide some clusters, however, being insufficiently developed to mix on a sample-to-sample level. In this case, most multi-view clustering runs with

Table 7.12: Comparing pairs of views with the full MIMIC dataset.

MIMIC Dataset	ACC			NMI			ARI			$F1$			$\mathcal{L}_{contrast}/\delta$		Mean fusion weight		
View pairs	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	C.I.	CXR	Vital signs	Lab tests
n = 123: All views	0.609	0.038	(0.602, 0.616)	0.042	0.023	(0.038, 0.046)	0.052	0.030	(0.047, 0.057)	0.606	0.039	(0.599, 0.613)	3.394	(3.241, 3.548)	0.357 (s = 0.010)	0.323 (s = 0.008)	0.320 (s = 0.008)
n = 59: CXR & vital signs	0.561	0.033	(0.553, 0.569)	0.015	0.015	(0.011, 0.019)	0.018	0.018	(0.013, 0.023)	0.555	0.035	(0.546, 0.564)	5.231	(4.764, 5.698)	0.540 (s = 0.023)	0.460 (s = 0.023)	—
n = 82: CXR & lab tests	**0.627**	0.039	(0.619, 0.635)	**0.053**	0.028	(0.047, 0.059)	**0.070**	0.035	(0.062, 0.078)	**0.622**	0.043	(0.613, 0.631)	5.108	(4.640, 5.574)	0.550 (s = 0.021)	—	0.450 (s = 0.021)
n = 215: Vital signs & lab tests	0.594	0.027	(0.590, 0.598)	0.028	0.013	(0.026, 0.030)	0.038	0.018	(0.036, 0.040)	0.591	0.027	(0.587, 0.595)	1.590	(1.372, 1.809)	—	0.463 (s = 0.012)	0.537 (s = 0.012)

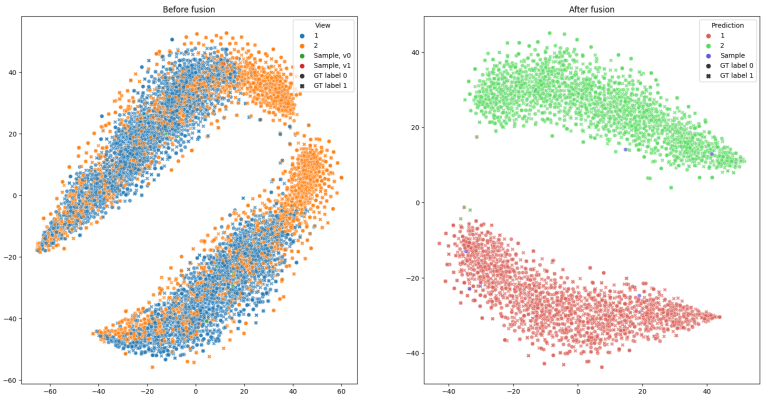
$V > 2$ will suffer for this issue of uneven convergence.

We observe that the view mixing of pairs of MIMIC data surpass the view mixing observed for all views (figs. 7.3 & 7.4). Yet, we observe that the contrastive loss surpasses that of the full dataset in both *CXR & vital signs* and *CXR & lab tests*. This may be due to the limiting factor of having the contrastive loss be scaled by $\min\{w_1, ..., w_V\}$ and/or the previously mentioned geometric limiting contrastive force. In addition, the weighted averaging over views that occurs in the contrastive loss function (eq. 4.14) results in the possibility that high-quality representation alignment among two views lessens the significance of bad mixing for the third view. The result of this is that the model may allow bad mixing for outlier views as long as the remaining views mixes well.

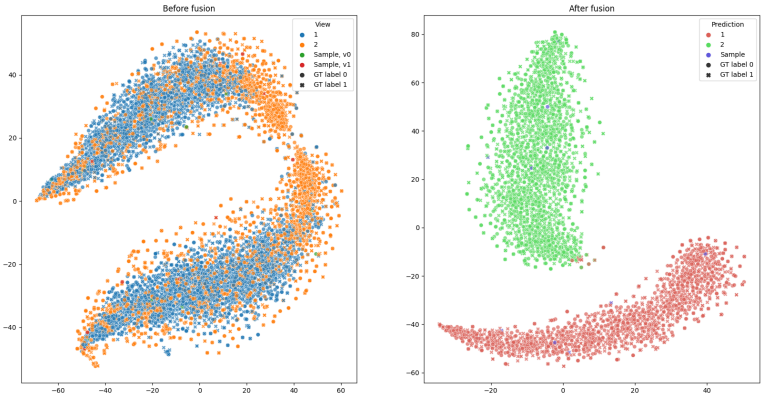
7.4.6.5 The Deep Semi-supervised Approach

For our semi supervised approach, we wish to combine the advantages of clustering with those of semi-supervised learning. As such, we add a semi-supervised loss term to the preexisting clustering loss with the intention of generating a more significant representation space that may be utilised in further data analysis. We will compare these to various other loss term combinations and compare fusion spaces. We implement the semi-supervised loss by determining a number of labeled samples, N_{ss}, determined when initiating the model. The loss function will be applied to these samples for all epochs in the run. The loss function selected is the binary cross-entropy (eq. 3.11).

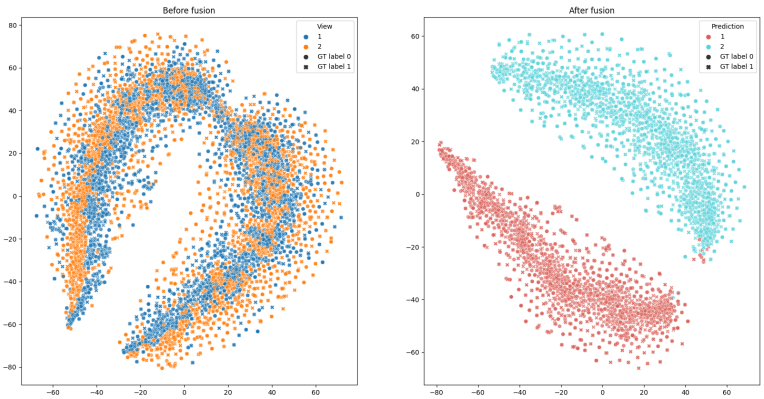
Table 7.13 displays results for DRAGMVC with added semi-supervised loss. As expected, performance metrics are positively affected by the increasing number sampled labels, peaking at 70.0% mean validation accuracy at our minimum mean DDC loss. We observe that the contrastive loss provides a significant positive impact over the non-contrastive runs. This suggests that representation-alignment does not negatively interfere with the added information of having access to labels. This notion is further strengthened by the fact that the contrastive loss is unaffected by increasing N_{ss}. In the case of interference between the losses, we would instead expect a tradeoff between contrastive loss, DDC loss, and semi-supervised loss. Considering the non-contrastive runs, we note that DDC loss is superior without contrast and not significantly affected by the number of semi-supervised labels. This is, however, not the case for the contrastive runs, where clustering loss significantly worsens for higher N_{ss}, suggesting that the clustering may be replaced by the available supervision. On the other hand, $\mathcal{L}_{contrast}$ is significantly improving with added semi-supervised labels. This may suggest that the available labels



(a) CXR & vital signs fusion plot.



(b) CXR & lab tests fusion plot.

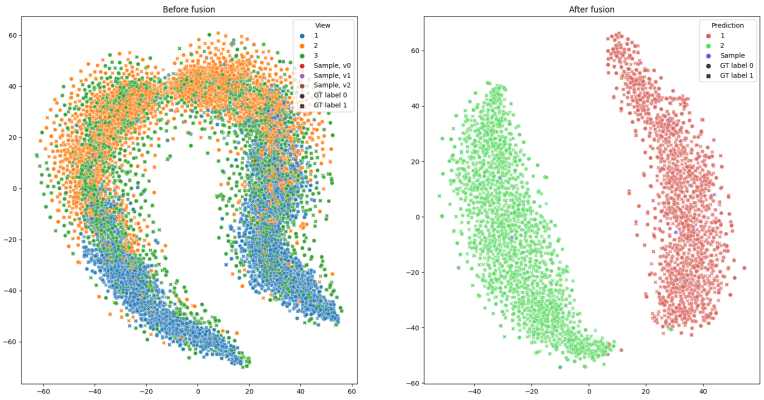


(c) Vital signs & lab tests fusion plot.

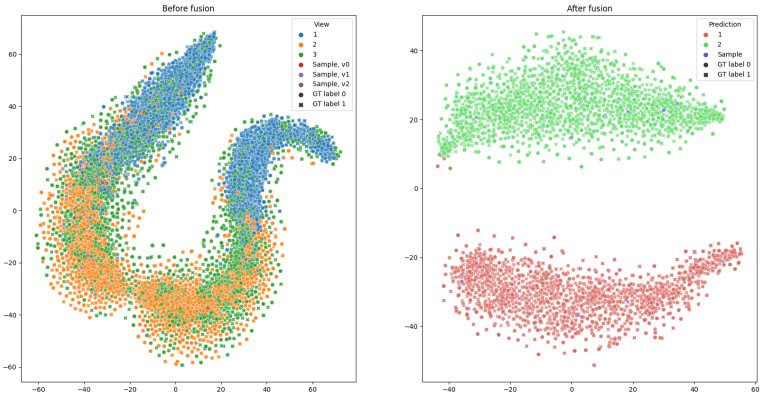
Figure 7.9: Runs by contrastive loss fusion plots (dimensionality reduced by t-SNE) from best (top) to worst (bottom).

Table 7.13: Results using BCE semi-supervised loss function on N_{semi} random samples. With and without contrastive loss.

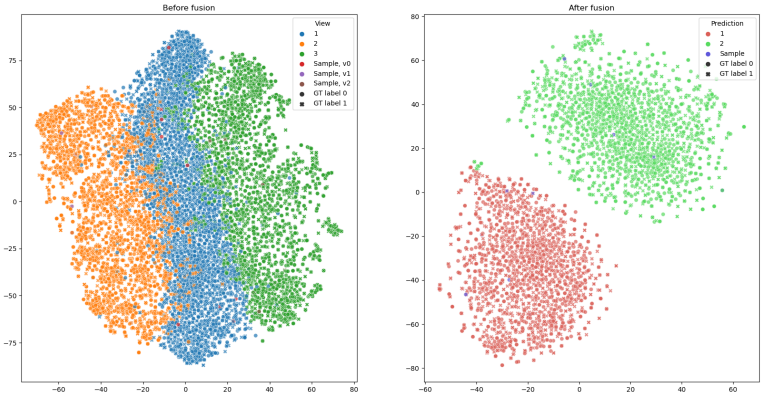
MIMIC Dataset	Metrics	ACC			NMI			ARI			$F1$			$\mathcal{L}_{ddc}$		$\mathcal{L}_{contrast}$	
Losses	Number of labels, N_{semi}	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	SD	C.I.	Mean	C.I.	Mean	C.I.
$\mathcal{L}_{ddc} + \mathcal{L}_{contrast} + \mathcal{L}_{semi-sup.}$	n = 73: N_{semi} = 50	0.641	0.047	(0.630, 0.652)	0.068	0.030	(0.061, 0.075)	0.087	0.043	(0.077, 0.097)	0.634	0.059	(0.620, 0.648)	0.989	(0.925, 1.053)	22.106	(20.872, 23.34)
	n = 110: N_{semi} = 100	0.667	0.032	(0.661, 0.673)	0.091	0.026	(0.086, 0.096)	0.116	0.039	(0.109, 0.123)	0.663	0.038	(0.656, 0.67)	0.981	(0.935, 1.027)	21.392	(20.372, 22.412)
	n = 72: N_{semi} = 250	0.682	0.023	(0.677, 0.687)	0.107	0.017	(0.103, 0.111)	0.133	0.030	(0.126, 0.140)	0.675	0.029	(0.668, 0.682)	1.076	(1.049, 1.103)	19.678	(19.187, 20.169)
	n = 108: N_{semi} = 500	**0.700**	0.027	(0.695, 0.705)	**0.127**	0.022	(0.123, 0.131)	**0.161**	0.036	(0.154, 0.168)	**0.691**	0.034	(0.685, 0.697)	1.118	(1.094, 1.142)	19.191	(18.832, 19.550)
$\mathcal{L}_{ddc} + \mathcal{L}_{semi-sup.}$	n = 64: N_{semi} = 50	0.618	0.035	(0.609, 0.627)	0.048	0.024	(0.042, 0.054)	0.060	0.030	(0.053, 0.067)	0.614	0.037	(0.605, 0.623)	0.972	(0.955, 0.989)	—	—
	n = 98: N_{semi} = 100	0.640	0.026	(0.635, 0.645)	0.066	0.018	(0.062, 0.070)	0.080	0.027	(0.075, 0.085)	0.636	0.027	(0.631, 0.641)	0.966	(0.952, 0.980)	—	—
	n = 67: N_{semi} = 250	0.660	0.023	(0.654, 0.666)	0.087	0.016	(0.083, 0.091)	0.104	0.028	(0.097, 0.111)	0.658	0.023	(0.652, 0.664)	0.946	(0.934, 0.958)	—	—
	n = 95: N_{semi} = 500	**0.676**	0.028	(0.670, 0.682)	**0.104**	0.018	(0.100, 0.108)	**0.126**	0.033	(0.119, 0.133)	**0.673**	0.031	(0.667, 0.679)	0.956	(0.937, 0.975)	—	—



(a) Fusion plot of the minimum $\mathcal{L}_{ddc}$ semi-supervised run with $N_{labeled} = 500$.



(b) Fusion plot of the minimum $\mathcal{L}_{contrast}$ semi-supervised run with $N_{labeled} = 500$.



(c) No contrastive loss. Fusion plot of the minimum $\mathcal{L}_{ddc}$ semi-supervised run with $N_{labeled} = 500$ and.

Figure 7.10: t-SNE reduced fusion plots by the semi-supervised DRAGMVC models.

help in extracting relevant features from views, contributing to improved view mixing. Figure 7.10 displays the fusion plots of minimum loss runs. We observe that the level of mixing resemble that of earlier fusion plots (figs. 7.3 & 7.4). This reinforces earlier analyses of the clustering setting: view counts $V > 2$ will yield sub-optimal mixing or views may not adhere to the multi-view principles. The no contrast run (fig. 7.10c) displays poor pre-fusion separability, which is to be expected. This could also result in a less informative subspace as the positions of representations will lose their similarities to surrounding representations.

7.5 Final Discussion & Future Work

7.5.1 Key Findings

In light of the tasks listed in section 7.4, we may list our key model findings as such:

1. *Performance on well-evaluated datasets*: We have displayed that our model surpasses MAGCN [13], the established state-of-the-art, in performance on three well-evaluated bibliographic graph datasets.

2. *Performance on complex medical data, compared to other SOTA*: We have displayed that our model yields significantly better mean performance metrics when compared to the current SOTA within fully unsupervised graph-based multi-view clustering.

3. *Performance on complex medical data, compared to non-graph counterpart*: We have compared our model to its non-graph counterparts to determine significant advantages of utilising graph data to strengthen underlying *pleural effusion*-associated clusters in terms of separability, view mixing, and classification.

4. *View mixing & cluster separability on complex data*: Through visualisation of the pre-fusion and fusion spaces in figs. 7.3, 7.4, & 7.10 as compared to the model's non-graph counterpart (fig. 7.5), we observe a very clear benefit of the utilisation of graph data to improve (i) view mixing through the use of contrastive loss; (ii) separability in both pre-fusion and fusion space.

5. *Semi-supervised performance*: We managed to achieve 70.0% mean validation accuracy in the semi-supervised setting, using $N_{ss} = 500$ labels. The performance was achieved having a significant positive effect on

contrastive loss, suggesting that supervision strengthen existing representation alignment.

6. *Semi-supervised effect on view mixing and clustering ability*: Separability of the semi-supervised approach appear to be negatively affected by the added semi-supervised loss term. Correspondingly, we observe that the added term significantly increased clustering loss. We interpret the weakened separability as the DDC loss and the semi-supervised loss limiting each other in regards to their respective objectives.

7. *Effects on GCN over-smoothing*: The hypothesis that our Markov prior assist in reducing over-smoothing is strengthened by the significant performance improvement seen by reducing the number of graph embedding layers in exchange for increasing the number of steps in our Markov prior (see table 7.6).

8. *Effects of using dropout*: We observe that by using dropout in both linear transformations and attentions in our graph attention embedding layers, we observe a significant improvement in performance both for the Cora and MIMIC datasets.

9. *Effects of the DDC encoder*: Using the graph attention layer in our DDC module has been shown to significantly improve performance over the original linear layer used in [50].

With regards to our constructed dataset, our key findings have been:

1. *Multi-view principles*: The results in table 7.12 may suggest that certain views do not add non-redundant and informative information to the dataset. However, we may not be certain of the cause for this result, as the contrastive effect for view counts $V > 2$ was argued to possibly be suboptimal due to geometric reasons and reasons based on the contrastive loss used (see section 7.4.6.4).

2. *Diagnostic significance & separability*: The dataset showed clear separability when employing graph data (see figs. 7.3, 7.4, & 7.10) while being limited in the non-graph multi-view approach (see fig. 7.5).

3. *Pairs of views*: We identify that not all pairs of views appear to contain strong natural clusters in regards to the underlying *pleural effusion* distribution. The weakest pair of views was *CXR* & *vital signs*, yielding the highest contrastive loss of all pairs, possibly implying that these may contain little common information related to the underlying distribution. The strongest view, *CXR* & *lab tests*, outperformed the full dataset by a

significant margin.

4. *k-NN graph*: From comparing the k-NN graph with the full MIMIC graph (from section 6.4), we found significant improvements in using the former.

7.5.2 Discussion

Seeing as the view mixing did not sufficiently incorporate all views in sample-to-sample alignment for view counts $V > 2$ (see figs. 7.3, 7.4, & 7.10) we aim to discuss possible causes for this shortcoming: (i) the data is lacking in informativeness or non-redundant data; (ii) concurrent training of encoders leading to instability; (iii) a suboptimal contrastive loss solution, not properly accounting for outlier views. The first of which is hard to determine the likeliness of with our dataset being new and untested prior to this thesis. Seeing the results from pairs of views (table 7.12) we observe that vital signs appear to be a limiting factor. However we cannot assert this without considering the limitations of the model (to be discussed). Considering the performance potential assessed in section 7.3.1.1, we observe a limitation of 73.2% accuracy. Higher than that of chest X-rays (66.8%). The best performance potential was observed in lab tests at 86.2%. Taking this into consideration along with the superior performance of chest X-rays & lab tests, we raise the suggestion that hard-to-cluster views (such as the CXRs) may learn from easy-to-cluster views (being lab tests). This allows for one fast converging view and one slow converging view that is learning from the former. This hypothesis explains the limitations in the other pairs of views, as vital signs & lab tests contain two equally easy-to-cluster views, both assumed (by their performance potential) to be superior in contained information to the CXRs. Extending the situation to $V > 2$, we experience multiple views attempting to concurrently learn from the other views. Our intuitive understanding is that two arbitrary views will be the first to discover some existing commonality, which will thereafter limit other views as the two views with commonality will seek to strengthen the identified similarities. This will force the additional views to attempt to learn traits corresponding to a predetermined common trait in the two fastest converging views — rather than concurrently strengthen innate traits to find common similarities to other views. We reason that this could be optimised by exploring an alternating optimisation scheme for views in an attempt to control convergence times. We will revisit this concept in section 8.1 covering potential future work.

Considering our model design choices, shortcomings may occur in our view encoders, graph embedding layers, or the clustering module employed. Firstly, our of our view encoders the image view seems to suffer the most with feature extraction (in figs. 7.3, 7.4, & 7.10). A possible weakness of our model is the

lack of a deeper, pretrained model for the image view. This would assist in the representation alignment for this view. In addition, this could aid in the problem with uneven view encoder learning as previously discussed, seeing as it would improve image convergence time. Secondly, considering our graph embedding layers, we establish its success by the improved cluster structure, as observed in figs. 7.3 & 7.4 as opposed to its non-graph counterpart in fig. 7.5. Yet, one apparent shortcoming of the graph convolution layers implementation is its lack of multi-head attention. The exclusion of multi-head attention was deliberate as it (using mean fusion and a comparable integration to [91]) was found to be insignificant in terms of clustering performance. Yet, other multi-head attention approaches may be employed, e.g. using concatenated attentions. Lastly, while the DDC clustering module is performing well, we reason that applications within graph clustering may be worth exploring. Particularly regarding the possibility of unifying feature-based and graph-based clustering. This could be utilised in the semi-supervised loss as e.g. label propagation for deep semi-supervised models [43].

For our medical data, the MIMIC dataset contained 3 520 nodes, and the Mini-MIMIC dataset contained 1 747. When comparing the performance metrics registered by DRAGMVC in table 7.7 as compared to table 7.8, it is a significant increase in performance with the increase in dataset size. Having graph data, we suggest that larger datasets could prove especially beneficial in the complex clustering case. Expanding on this, performance limitations in our evaluation of the MIMIC dataset may be further restricted by the size of our dataset, as was the case with the Mini-MIMIC dataset. We have also discussed the possibility of dataset size being a possible contributing factor in the poor spread of the CXR view in fusion plots.

We have shown that certain clusters were well-confined to a set of diagnoses by having a high degree of separation, as well as high separability in significant p-values. We believe to have displayed that applications using expert-evaluated data, informative and non-redundant views could show potential even in a fully unsupervised manner given its performance consistency due to the sample-level contrastive alignment. While showing promising results, employing DRAGMVC on our custom MIMIC dataset did not yield sufficient alignment to be able to consider model interpretability and between-sample relations for medical analysis. However, in the case of proper view alignment and high-level predictions, we hypothesise that the analysing the similarities between sets of view-aligned samples to nearby view-aligned samples could yield promising results for a range of possible applications, e.g.: (i) risk stratification; (ii) the recognition of sub-groups within diagnosis clusters; (iii) the progression of a diagnosis over time. Nevertheless, for our constructed MIMIC dataset, fully unsupervised

solutions has been shown to be unrealistic due to the lack of proper view alignment for view counts $V > 2$ — and, based on predictive performance in table 7.7 & 7.13, assumed to be insufficient dataset information in the case of $V = 2$.

8

Conclusion

In summary, this thesis reviews the field of graph-based multi-view clustering to lay the groundwork for a novel dataset and a state-of-the-art machine learning model within the field of deep multi-view clustering on one view-invariant graph. We achieved 76.2% maximum accuracy, 58.6% maximum geometric NMI, 56.8% maximum ARI, and 64.0% maximum macro-$F1$ on the Cora dataset; 72.4% maximum accuracy, 47.8% maximum NMI, 49.0% maximum ARI, and 56.8% maximum $F1$ on CiteSeer; 75.0% maximum accuracy, 34.9% maximum NMI, 39.0% maximum ARI, and 72.0% maximum $F1$ on PubMed, thus surpassing the established state-of-the-art within graph-based multi-view clustering on a single view-invariant graph. Additionally, we achieved 63.0% mean unsupervised accuracy, 6.1% mean unsupervised NMI, 7.1% mean unsupervised ARI, and 62.8% mean unsupervised $F1$ on our constructed *pleural effusion/not pleural effusion* graph-based multi-modal medical dataset. On the same dataset we achieved 70.0% mean accuracy using 500 labels in a semi-supervised approach. Experiments performed and their interpretations has been detailed, as well as a final discussion which covered possible limitations and areas of improvement. Being highly specialised and wide-reaching in subject matter, graph-based multi-view clustering require thorough background theory to properly cover. Our method is built upon the existing CoMVC [104] model by implementing a new interpretation of the graph attention convolution as seen in GATE [91] and, extended to the MVC case: MAGCN [13]. Our graph-embedding layer used shared weights, and utilises an attention graph weighted by a *Markov prior* with weights as probabilites of having visited a node in random walk process over a set number of steps. The

Markov prior approach to graph attention convolutions has been unexplored within the realm of graph-based multi-view clustering to the best of the author's knowledge. The advantage of using a Markov prior is two-fold: (i) it provides a more informative initialisation for graph attention; (ii) it allows for farther-reaching attention than the traditional graph masking which only attends to graph neighbours, as well as more specialised attention than the unmasked attention struggling with over-smoothing as datasets increase with number of graph convolution layers L [76]. This hypothesis was strengthened by experiments replacing graph attention convolution layers with additional Markov prior steps, showing significant improvements for the additional Markov prior steps with a shallower model.

Another significant contribution was the creation of the *pleural effusion/not pleural effusion* MIMIC dataset described in section 6. We extracted, combined, and pre-processed data from the MIMIC-IV and MIMIC-CXR databases [47; 49; 24]. Extending the dataset to graph applications, we computed a k-NN affinity matrix (section 6.4) based on radiology reports in a fully unsupervised manner. Other datasets gathered for evaluation were Cora, CiteSeer, and PubMed [93; 73]: relational datasets of papers containing attributes by words used and a graph computed from their respective references. Using our MIMIC dataset, we explored the clustering capabilities of our model, displaying massive improvements over the base model in separability, view alignment, clustering optimisation, in addition to performance. Thus suggesting that our contribution successfully embed graph information in a clustering context. From thorough analysis of the MIMIC dataset, we uncover of weaknesses and areas of potential improvement of our model and the dataset used. Most prominent was the issue of uneven view alignment. We argue that the issue stems from an unstable concurrent encoder optimalisation and a contrastive loss that is too relaxed in regards to outlier views (for view counts $V > 2$). Yet, aligning representations on a sample-to-sample level was shown to yield superior results when compared to the distribution-wide alignment in MAGCN [13]. For the semi-supervised approach we have discovered that semi-supervised loss does not interfere with our main clustering objective, allowing the model to be used in a semi-supervised settings as well. The semi-supervised models yielded equally view-mixed subspaces as well as superior performance in comparison to the fully unsupervised DRAGMVC. The semi-supervised setting did also benefit from representation alignment, significantly improving in both clustering loss as well as accuracy for the contrastive model, while positively impacting the contrastive loss, suggesting that the two loss objectives align.

Due to the nature of medical data in diagnostics, correlation is a strong presence in our MIMIC dataset. DRAGMVC displayed an ability to — in certain

cases — recognise the more isolated diagnosis clusters (see run 1 in fig. 7.3a): a difficult task considering the many challenges when employing imperfect health data. From running supervised performance potential tests on evaluation sets of each view (chest X-rays: 66.8%; vital signs: 73.2%; and lab tests: 86.2%), we discovered the limitations of each view separately. Then, employing DRAGMVC on pairs of views shown that the chest X-rays and lab tests may contain the most non-redundant, complementary information for clustering — outperforming the full dataset with all three views. This does not necessarily contradict the multi-view principles, as we have discussed with regards to the limitations of DRAGMVC for complex views and view counts $V > 2$.

With regards to the lists describing our hypothesised model and dataset in the introduction (see section 1.2), we attend to every item, while having thoroughly discussed the weaknesses, doubts, and ambiguities associated with: (i) the extraction of diagnostically relevant information for all views separately; (ii) the nuance between data points in representation space; (iii) the dataset's adherence to the multi-view principles. Finalising the thesis we summarise the contributions undertaken in key words: (i) the examination of relevant literature; (ii) the hypothesising and developing of a novel machine learning model; (iii) construction and processing of a real-life medical dataset containing chest X-rays; (iv) the analysis, evaluation, and discussion of the proposed dataset and the proposed model.

8.1 Future Work

Other approaches that builds on DRAGMVC include the extension into similar representation learning approaches, e.g. inspired by the work of Hassani & Khasahmadi [33]. Another potentially interesting extension in the medical sense is to extend the network to account for multiple graphs [64; 78] or other heterogeneous graph approaches [94]. Considering the dataset, the image view is thoroughly limited and noise-affected by the details surrounding the lungs. Far lacking in sufficient medical knowledge[1], the task of segmenting lungs in the X-ray images was — in our case — deemed to difficult to accomplish in an unsupervised manner while lacking expertise for manual evaluation. One potential development directed towards interpretability is to apply e.g. Grad-CAM [92] in order to recognise any determining properties of the image view. Furthermore, successful masking of the image data could improve on its informativeness by removing a likely significant source of noise. This could

1. Recognising the borders of lungs in chest X-rays prove challenging for pleural effusion data given the diffuse edges.

potentially be accomplished unsupervised by e.g. utilisation of SLICO superpixels [118]. For our vectorial medical data, we selected the zero imputation method for missing data. The effectiveness of various imputation methods is worth exploring more closely, along with any potential differences between these methods in multi-view versus single view learning approaches.

Bibliography

[1] Appendix C: Positive Semidefinite and Positive Definite Matrices. In *Parameter Estimation for Scientists and Engineers*, pages 259–263. John Wiley & Sons, Ltd, 2007. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470173862.app3.

[2] Shotaro Akaho. A kernel method for canonical correlation analysis. October 2006.

[3] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. Self-Training: A Survey. *arXiv:2202.12040 [cs]*, February 2022. arXiv: 2202.12040.

[4] David Arthur and Sergei Vassilvitskii. K-Means++: The Advantages of Careful Seeding. volume 8, pages 1027–1035, January 2007.

[5] Chen Bei. IMKBLE/MAGCN, March 2022. original-date: 2021-03-29T08:43:04Z.

[6] Michael Berry and Murray Browne. *Lecture notes in data mining*. January 2006.

[7] Carlo Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

[8] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101.

[9] Bokai Cao, Lifang He, Xiangnan Kong, Philip Yu, Zhifeng Hao, and Ann Ragin. Tensor-Based Multi-view Feature Selection with Applications to Brain Diseases. *Proceedings / IEEE International Conference on Data Mining. IEEE International Conference on Data Mining*, 2014:40–49, De-

cember 2014.

[10] Menglin Cao, Ming Yang, Chi Qin, Xiaofei Zhu, Yanni Chen, Jue Wang, and Tian Liu. Using DeepGCN to identify the autism spectrum disorder from multi-site resting-state data. *Biomedical Signal Processing and Control*, 70:103015, September 2021.

[11] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. *arXiv:1801.10247 [cs]*, January 2018. arXiv: 1801.10247.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations, 2020. _eprint: 2002.05709.

[13] Jiafeng Cheng, Qianqian Wang, Zhiqiang Tao, Deyan Xie, and Quanxue Gao. Multi-View Attribute Graph Convolution Networks for Clustering. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2973–2979. International Joint Conferences on Artificial Intelligence Organization, July 2020.

[14] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, July 2019. arXiv: 1905.07953.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[16] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-Scale Convolutional Neural Networks for Time Series Classification. *CoRR*, abs/1603.06995, 2016. arXiv: 1603.06995.

[17] Sanjoy Dasgupta, Michael Littman, and David Mcallester. PAC Generalization Bounds for Co-training. In *Advances in Neural Information Processing Systems*, pages 375–382, January 2001.

[18] David L. Davies and Donald W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei.

Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[20] Ivo D. Dinov. Volume and Value of Big Healthcare Data. *Journal of medical statistics and informatics*, 4:3, 2016.

[21] Monika Doerfler and Thomas Grill. Inside the Spectrogram: Convolutional Neural Networks in Audio Processing. July 2017.

[22] S. Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, January 1967. Conference Name: IEEE Transactions on Information Theory.

[23] Ana Fred and Anil Jain. Data clustering using evidence accumulation. In *Proceedings - International Conference on Pattern Recognition*, volume 16, pages 276– 280 vol.4, February 2002.

[24] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. Publisher: Am Heart Assoc.

[25] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[27] Erland Grimstad. Exploring Multi-modal Medical Data with Deep Multi-view Clustering, December 2021.

[28] Erland Grimstad. erlandg/thesis, June 2022. original-date: 2022-06-01T12:06:51Z.

[29] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved Deep Embedded Clustering with Local Structure Preservation. August 2017.

[30] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216 [cs, stat]*, September

2018. arXiv: 1706.02216.

[31] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, March 2011.

[32] Md Inzamam Ul Haque, Abhishek K. Dubey, and Jacob D. Hinkle. The Effect of Image Resolution on Automated Classification of Chest X-rays. *medRxiv*, 2021. Publisher: Cold Spring Harbor Laboratory Press _eprint: https://www.medrxiv.org/content/early/2021/08/01/2021.07.30.21261225.full.pdf.

[33] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive Multi-View Representation Learning on Graphs. *arXiv:2006.05582 [cs, stat]*, June 2020. arXiv: 2006.05582.

[34] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*, February 2015. arXiv: 1502.01852.

[37] Geoffrey Hinton and Sam Roweis. Stochastic Neighbor Embedding. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 857–864, Cambridge, MA, USA, 2002. MIT Press.

[38] Harold Hotelling. Relations Between Two Sets of Variates. *Biometrika*, 28(3/4):321–377, 1936. Publisher: [Oxford University Press, Biometrika Trust].

[39] Zhen Hu, Genevieve B. Melton, Elliot G. Arsoniadis, Yan Wang, Mary R. Kwaan, and Gyorgy J. Simon. Strategies for handling missing clinical data for automated surgical site infection detection from the electronic health record. *Journal of Biomedical Informatics*, 68:112–120, 2017.

[40] Jiaming Huang, Z. Li, V. Zheng, Wen Wen, Yifan Yang, and Yuanmi Chen. Unsupervised Multi-view Nonlinear Graph Embedding. In *UAI*, 2018.

[41] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Nor-

malization Techniques in Training DNNs: Methodology, Analysis and Application, 2020. _eprint: 2009.12836.

[42] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

[43] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label Propagation for Deep Semi-supervised Learning. Technical Report arXiv:1904.04717, arXiv, April 2019. arXiv:1904.04717 [cs] type: article.

[44] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[45] Janus Jakobsen, Christian Gluud, Jørn Wetterslev, and Per Winkel. When and how should multiple imputation be used for handling missing data in randomised clinical trials - A practical guide with flowcharts. *BMC Medical Research Methodology*, 17, December 2017.

[46] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjørn Eltoft. The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 343(6):614–629, 2006.

[47] A Johnson, L Bulgarelli, T Pollard, S Horng, LA Celi, and R Mark. MIMIC-IV (version 1.0), 2020.

[48] Alistair Johnson. Tweet: MIMIC-IV and MIMIC-CXR use the same subject identifiers, ..., 2020.

[49] Alistair E W Johnson, Tom J Pollard, Seth J Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):1–8, 2019. Publisher: Nature Publishing Group.

[50] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep divergence-based approach to clustering. *Neural Networks*, 113:91–101, May 2019. Publisher: Elsevier BV.

[51] K. Karhunen. Zur Spektraltheorie stochastischer Prozesse. *Ann. Acad. Sci. Finnicae, Ser. A*, 1:34, 1946.

[52] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul

Sukthankar, and Li Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[53] Keisuke Kawauchi, Sho Furuya, Kenji Hirata, Chietsugu Katoh, Osamu Manabe, Kentaro Kobayashi, Shiro Watanabe, and Tohru Shiga. A convolutional neural network-based system to classify patients using FDG PET/CT examinations. *BMC Cancer*, 20, March 2020.

[54] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952. Publisher: Institute of Mathematical Statistics.

[55] Benjamin King. Step-Wise Clustering Procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1967.10482890.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.

[57] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, February 2017. arXiv: 1609.02907.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[59] H. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly*, 2, May 2012.

[60] T.M. Lehmann, C. Gonner, and K. Spitzer. Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, 1999.

[61] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs Go as Deep as CNNs? *arXiv:1904.03751 [cs]*, August 2019. arXiv: 1904.03751.

[62] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive Graph Convolutional Neural Networks. *arXiv:1801.03226 [cs, stat]*, January 2018. arXiv: 1801.03226.

[63] David Kewei Lin. Clustering Phenomena in Dropout, 2019.

[64] Liang Liu, Zhao Kang, Ling Tian, Wenbo Xu, and Xixu He. Multilayer Graph Contrastive Clustering Network. *arXiv:2112.14021 [cs]*, December 2021. arXiv: 2112.14021.

[65] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[66] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[67] J. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

[68] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[69] Jérôme Mariette and Nathalie Villa-Vialaneix. Unsupervised multiple kernel learning for heterogeneous data integration. *Bioinformatics*, 34(6):1009–1015, 2017.

[70] James H. McClellan, Ronald W. Schafer, and Mark A. Yoder. *DSP First (2nd Edition)*. Prentice-Hall, Inc., USA, 2007.

[71] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, October 2019. Publisher: Proceedings of the National Academy of Sciences.

[72] I. Muslea, S. Minton, and C. A. Knoblock. Active Learning with Multiple Views. *Journal of Artificial Intelligence Research*, 27:203–233, October 2006. Publisher: AI Access Foundation.

[73] Galileo Namata, Ben London, L. Getoor, and Bert Huang. Query-driven Active Surveying for Collective Classification. *undefined*, 2012.

[74] Mariá C. V. Nascimento and André C. P. L. F. de Carvalho. Spectral methods for graph clustering – A survey. *European Journal of Operational Research*, 211(2):221–231, June 2011.

[75] Soumya Ranjan Nayak, Deepak Ranjan Nayak, Utkarsh Sinha, Vaibhav Arora, and Ram Bilas Pachori. Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study. *Biomedical Signal Processing and Control*, 64:102365, 2021.

[76] Kenta Oono and Taiji Suzuki. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. Technical Report arXiv:1905.10947, arXiv, January 2021. arXiv:1905.10947 [cs, stat] type: article.

[77] Juri Opitz and Sebastian Burst. Macro F1 and Macro F1. Technical Report arXiv:1911.03347, arXiv, February 2021. arXiv:1911.03347 [cs, stat] type: article.

[78] Erlin Pan and zhao kang. Multi-view Contrastive Graph Clustering. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[79] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, Ricardo Guerrero, Ben Glocker, and Daniel Rueckert. Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease. *Medical Image Analysis*, 48:117–130, August 2018.

[80] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

[81] Novi Quadrianto and Christoph H. Lampert. Learning Multi-View Neighborhood Preserving Projections. In *ICML*, 2011.

[82] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, 2017. _eprint: 1711.05225.

[83] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9(83):2491–2521, 2008.

[84] Ingrid Keilegavlen Rebnord, Tone Morken, Kjell Maartmann-Moe, and Steinar Hunskaar. Out-of-hours workload among Norwegian general

practitioners – an observational study. *BMC Health Services Research*, 20(1):944, October 2020.

[85] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015. _eprint: 1505.04597.

[86] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[87] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. Publisher: Institute of Mathematical Statistics.

[88] Sheldon M. Ross. 4 - Markov Chains. In Sheldon M. Ross, editor, *Introduction to Probability Models (Twelfth Edition)*, pages 193–291. Academic Press, January 2019.

[89] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000. _eprint: https://www.science.org/doi/pdf/10.1126/science.290.5500.2323.

[90] Fátima A Saiz and Iñigo Barandiaran. COVID-19 Detection in Chest X-ray Images using a Deep Learning Approach. *Int. J. Interact. Multim. Artif. Intell.*, 6(2):1–4, 2020.

[91] Amin Salehi and Hasan Davulcu. Graph Attention Auto-Encoders. Technical Report arXiv:1905.10715, arXiv, May 2019. arXiv:1905.10715 [cs, stat] type: article.

[92] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, 128(2):336–359, February 2020. arXiv:1610.02391 [cs].

[93] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3):93–93, September 2008. Number: 3.

[94] Zezhi Shao, Yongjun Xu, Wei Wei, Fei Wang, Zhao Zhang, and Feida Zhu. Heterogeneous Graph Neural Network with Multi-view Representation

Learning. *arXiv:2108.13650 [cs]*, August 2021. arXiv: 2108.13650.

[95] Dan Shi, Lei Zhu, Zhiyong Cheng, Zhihui Li, and Huaxzhang Zhang. Unsupervised Multi-View Feature Extraction with Dynamic Graph Learning. *Journal of Visual Communication and Image Representation*, 56, September 2018.

[96] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[97] Satya P. Singh, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, and Balázs Gulyás. 3D Deep Learning on Medical Images: A Review, 2020. _eprint: 2004.00218.

[98] P.H.A. Sneath, P.H.A. Sneath, R.R. Sokal, and U.R.R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. A Series of books in biology. W. H. Freeman, 1973.

[99] Jia Song, Shaohua Gao, Yunqiang Zhu, and Chenyan Ma. A survey of remote sensing image classification based on CNNs. *Big Earth Data*, 3(3):232–254, 2019. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/20964471.2019.1657720.

[100] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006.

[101] Ellen Rabben Svedahl, Kristine Pape, Marlen Toch-Marquardt, Lena Janita Skarshaug, Silje-Lill Kaspersen, Johan Håkon Bjørngaard, and Bjarne Austad. Increasing workload in Norwegian general practice – a qualitative study. *BMC Family Practice*, 20(1):68, May 2019.

[102] Yu-Xing Tang, You-Bao Tang, Yifan Peng, Ke Yan, Mohammadhadi Bagheri, Bernadette Redd, Catherine Brandon, Zhiyong lu, Mei Han, Jing Xiao, and Ronald Summers. Automated abnormality classification of chest radiographs using deep convolutional neural networks. *npj Digital Medicine*, 3, December 2020.

[103] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition, 2008.

[104] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. Reconsidering Representation Alignment for Multi-view

Clustering. *arXiv:2103.07738 [cs]*, March 2021. arXiv: 2103.07738.

[105] Hanneke van der Hoef and Matthijs J. Warrens. Understanding information theoretic measures for comparing clusterings. *Behaviormetrika*, 46(2):353–370, October 2019.

[106] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[107] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*, February 2018. arXiv: 1710.10903.

[108] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye. *Probability & Statistics for Engineers and Scientists*. Pearson Education, Upper Saddle River, 9 edition, 2017.

[109] Hao Wang, Yan Yang, and Bing Liu. GMC: Graph-based Multi-view Clustering. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, March 2019.

[110] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2005.

[111] George Wolberg. Cubic Spline Interpolation: A Review. 1988.

[112] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis, 2016. _eprint: 1511.06335.

[113] Chang Xu, Dacheng Tao, and Chao Xu. A Survey on Multi-view Learning, 2013. _eprint: 1304.5634.

[114] Pei Yang and Wei Gao. Information-Theoretic Multi-view Domain Adaptation: A Theoretical and Empirical Study. *The Journal of Artificial Intelligence Research (JAIR)*, 49, March 2014.

[115] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A Survey on Deep Semi-supervised Learning. *arXiv:2103.00550 [cs]*, August 2021. arXiv: 2103.00550.

[116] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Mining and Analytics*, 1:83–107, June 2018.

[117] Xujing Yao, Xinyue Wang, Shuihua Wang, and Yu-Dong Zhang. A comprehensive survey on convolutional neural network in medical image analysis. *Multimedia Tools and Applications*, August 2020.

[118] Bilal Yassine, Paul Taylor, and Al Story. Fully automated lung segmentation from chest radiographs using SLICO superpixels. *Analog Integrated Circuits and Signal Processing*, 95(3):423–428, June 2018.

[119] Dingding Yu, Kaijie Zhang, Lingyan Huang, Bonan Zhao, Xiaoshan Zhang, Xin Guo, Miaomiao Li, Zheng Gu, Guosheng Fu, Minchun Hu, Yan Ping, Ye Sheng, Zhenjie Liu, Xianliang Hu, and Ruiyi Zhao. Detection of peripherally inserted central catheter (PICC) in chest X-ray images: A multi-task deep learning model. *Computer Methods and Programs in Biomedicine*, 197:105674, 2020.

[120] Ming Yuan and Yi Lin. Model Selection and Estimation in Regression With Grouped Variables. *Journal of the Royal Statistical Society Series B*, 68:49–67, February 2006.

[121] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[122] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph Sampling Based Inductive Learning Method. *arXiv:1907.04931 [cs, stat]*, February 2020. arXiv: 1907.04931.

[123] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[124] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, January 2020.

[125] Y. Zhou and Rama Chellappa. Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, pages 71–78 vol.2, 1988.

[126] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *undefined*, 2002.

[127] Chenyi Zhuang and Qiang Ma. Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification. pages 499–508, April

2018.

[128] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning, 2020. _eprint: 1911.02685.

Appendix

A.1 Dimensionality Reduction

The task of building an understanding of big data is often challenged by high-dimensional feature vectors that cannot be visualised directly. Thus, having ways of reducing data dimensionality becomes an important mechanism. These methods are often referred to as *feature reduction* or *dimensionality reduction* methods. In this section, two of these methods are presented as they will be utilised to visualise otherwise difficult-to-interpret data.

A.1.1 Principal Component Analysis

The idea behind Principal Component Analysis (PCA) — otherwise known as the Karhunen-Loève transform [51] — is to spread the variance of multivariate data over its dimensions in a way such that as much information is contained in as few dimensions as possible. This is done by removing correlation between variables. An intuitive understanding of this process can be described as trying to picture a three-dimensional object from a view which maximises the size the object. Then creating a two-dimensional drawing from that perspective. Capturing a 3D object in a 2D representation from this view would therefore maximise variance in a lower-dimensional representation. When applying dimensionality reduction by PCA this is essentially what one is doing. Trying to find an angle in reduced dimensionality in which the spread of the data is maximal. However, as higher-dimensional objects cannot be visualised effectively, one must resort to mathematics.

A.1.1.1 De-correlation

Now, to decorrelate the components. Let $\boldsymbol{x} \in \mathbb{R}^p$ be a random vector with mean $\boldsymbol{\mu_x}$ and covariance matrix

$$\Sigma_{\boldsymbol{x}} = \mathbb{E}\big[(\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{x} - \boldsymbol{\mu_x})^T\big]. \tag{1}$$

Defining some transformation matrix $\boldsymbol{A}$, as such

$$\boldsymbol{y} = \boldsymbol{A}^T \boldsymbol{x} \tag{2}$$

having parameters $\boldsymbol{\mu_y}$ and $\Sigma_{\boldsymbol{y}}$ found by

$$\begin{aligned} \boldsymbol{\mu_y} &= \mathbb{E}\big[\boldsymbol{A}^T \boldsymbol{x}\big] \\ &= \boldsymbol{A}^T \mathbb{E}\big[\boldsymbol{x}\big] \\ &= \boldsymbol{A}^T \boldsymbol{\mu_x} \end{aligned} \tag{3}$$

and

$$\begin{aligned} \Sigma_{\boldsymbol{y}} &= \mathbb{E}\big[(\boldsymbol{y} - \boldsymbol{\mu_y})(\boldsymbol{y} - \boldsymbol{\mu_y})^T\big] \\ &= \mathbb{E}\big[(\boldsymbol{A}^T \boldsymbol{x} - \boldsymbol{A}^T \boldsymbol{\mu_x})(\boldsymbol{A}^T \boldsymbol{x} - \boldsymbol{A}^T \boldsymbol{\mu_x})^T\big] \\ &= \mathbb{E}\big[\boldsymbol{A}^T (\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{x} - \boldsymbol{\mu_x})^T \boldsymbol{A}\big] \\ &= \boldsymbol{A}^T \mathbb{E}\big[(\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{x} - \boldsymbol{\mu_x})^T\big] \boldsymbol{A} \\ &= \boldsymbol{A}^T \Sigma_{\boldsymbol{x}} \boldsymbol{A}. \end{aligned} \tag{4}$$

For our new, transformed vector $\boldsymbol{y}$ to be uncorrelated the covariance between variables must be zero, which requires a zero off-diagonal, i.e. $\Sigma_{\boldsymbol{y}}(i, j) = 0, \;\; i \neq j, \; i, j = 1, ..., p$. We obtain the diagonal matrix from orthogonal basis $\boldsymbol{A}$ — which is composed of the normalised *eigenvectors* of $\Sigma_{\boldsymbol{x}}$, i.e. $\boldsymbol{a}_0, \boldsymbol{a}_1, ..., \boldsymbol{a}_{N-1}$ [103]

$$\Sigma_{\boldsymbol{y}} = \boldsymbol{\Lambda} \tag{5}$$

where the diagonal values of $\boldsymbol{\Lambda}$ are the eigenvalues of $\Sigma_{\boldsymbol{x}}$, $\lambda_0, \lambda_1, ..., \lambda_{N-1}$, commonly ordered such that $\lambda_0 > \lambda_1 > ... > \lambda_{N-1}$. From most significant basis vector to the least significant. Now, with $\Sigma_{\boldsymbol{x}}$ being positive definite[2] [103], we know that all eigenvalues of $\Sigma_{\boldsymbol{x}}$, $\lambda_0, \lambda_1, ..., \lambda_{N-1}$ must be positive.

2. A symmetric matrix $\boldsymbol{A}$ where $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$ for all real and non-zero $\boldsymbol{x}$ [1].

Knowing that the basis $\boldsymbol{A}$ constructs a transformation such that

$$\begin{aligned}\sum_{i=1}^{p} \mathrm{Var}(x_i) &= \mathrm{Trace}(\Sigma_{\boldsymbol{x}}) \\ &= \mathrm{Trace}\left(\left(\boldsymbol{A}^T\right)^{-1}\boldsymbol{\Lambda}\boldsymbol{A}^{-1}\right) \\ &= \mathrm{Trace}(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{\Lambda}) \\ &= \mathrm{Trace}(\boldsymbol{\Lambda}) \\ &= \sum_{j=1}^{p} \lambda_j \\ &= \sum_{j=1}^{p} \mathrm{Var}(y_j), \end{aligned} \tag{6}$$

we may assert the proportionally contained variance in a dimensionality reduced representation by the sum of its eigenvalues. Let $l < p$ where l is the number of dimensions in the reduced space. We may construct a new basis

$$\tilde{\boldsymbol{A}}^T = \begin{bmatrix} \boldsymbol{a}_0^T \\ \vdots \\ \boldsymbol{a}_l^T \end{bmatrix} \in \mathbb{R}^{l \times p} \tag{7}$$

from which we deduce our vector $\boldsymbol{y} \in \mathbb{R}^l$ in reduced space

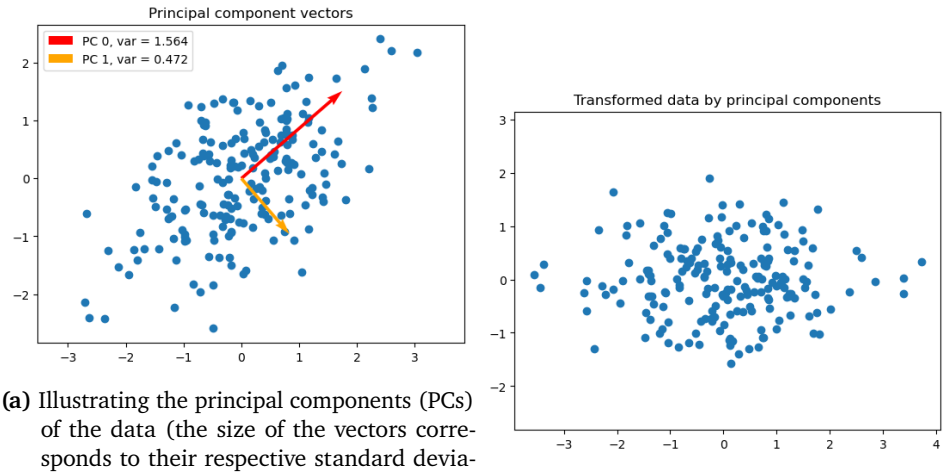
$$\tilde{y} = \tilde{\boldsymbol{A}}^T \boldsymbol{x} \tag{8}$$

which we know will retain a proportional $\sum_{i=1}^{l} \lambda_i / \mathrm{Trace}(\boldsymbol{\Lambda})$ of the variance from $\boldsymbol{x}$.

Figure A.1 illustrates the PCs of random data points. The data shown in A.1b can be reduced from $p = 2$ dimensions down to $l = 1$ dimension by applying the Karhunen-Loève transform (PCA) by projecting the data points onto the most significant axis, in this case shown as the horizontal axis.

A.1.2 *t*-SNE

t-distributed stochastic neighbour embedding (*t*-SNE) is a data visualisation tool used for visualising high-dimensional data in low dimensions [66]. It is based on the normal distributed Stochastic Neighbour Embedding (SNE) [37], but with a *t*-distributed representation in lower dimensions. We will get back to why this is the case. While PCA aims to find the angle at which variance is maximised, *t*-SNE is not a coordinate-based representation of the data but a



(a) Illustrating the principal components (PCs) of the data (the size of the vectors corresponds to their respective standard deviations).

(b) The rotated data by its most significant PCs.

Figure A.1: A randomly generated dataset from a multinormal distribution with parameters $\boldsymbol{\mu} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $\Sigma_{\boldsymbol{x}} = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$

statistical one. Therefore, the resulting representation will not be informative regarding exact relative placement of data points, but rather their statistical similarity [66]. This means that points are *not* transformed to represent their proximity to other points, but by a statistical measure. This is often helpful as many data distributions tend to be skewed along dimensions due to a correlation in its variables (such as in fig. A.1a). Using t-SNE in these cases will better help propagate the *statistical* similarity through dimensionality reduction.

Given a set of objects in higher dimensions, $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$, we may from one vector $\boldsymbol{x}_i$ find its similarity to another vector $\boldsymbol{x}_j$ through the Gaussian distribution, which may be normalised as follows [66]:

$$p_{i|j} = \frac{\exp\{-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 / 2\sigma_i^2\}}{\sum_{k \neq i} \exp\{-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2 / 2\sigma_i^2\}} \tag{9}$$

where σ_i^2 is the variance of the Gaussian distribution on $\boldsymbol{x}_i$ and letting $p_{i|i} = 0$. The variance, σ_i^2, should be varying depending on each datapoint, as density in high-dimensional space may vary — reasoning that the variance should be lower in dense regions and higher in sparse regions [66]. For this, t-SNE utilise a fixed hyperparameter *perplexity*. The process of finding σ_i^2 is then a process of selecting the distribution in which perplexity is equal to the set value. Perplexity for a given probability P_i is given as:

$$\text{Perp}(P_i) = 2^{H(P_i)}, \tag{10}$$

where $H(P_i)$ is defined as the Shannon entropy:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}. \tag{11}$$

The effect of perplexity will thus be regarding the ”pulling” force by the statistical distribution. Higher Perp values will therefore find connections between points separated by large distances, while low Perp values will look at the immediate surrounding area from each point.

L. van der Maaten and G. Hinton [66] reasons that the Student's t-distribution will be beneficial for the lower representation, rather than the original Gaussian distribution. This follows from its tails being heavier. This properties allows moderate distances in higher-dimensional space to be represented as larger distances in reduced space. This will avoid the strong attractive forces found in SNE. The distribution in reduced dimensionality can thus be defined as

$$q_{ij} = \frac{\left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}}{\sum_k \sum_{l \neq k} \left(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2\right)^{-1}}. \tag{12}$$

From eqs. 9 and 12 the dimensionality reduced data $\boldsymbol{y}_1, ..., \boldsymbol{y}_N$ are determined by minimising the Kullback–Leibler (KL) divergence of the two distributions, i.e. maximising the distributions' overlap

$$\mathrm{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{13}$$

This is accomplished by employing the optimisation tool *gradient descent*, which will be covered in detail in chapter 3.

A.2 Clustering Performance

A.2.1 Initialisation

A.2.1.1 K-means initialisation

Defining the initial k centroids $\boldsymbol{\theta}_j$, may be executed in a a variety of ways. The original approach is simply to randomly select points within input space. However, to improve the loss landscape alternate initialisation methods may be utilised. One such example is the K-means++ initialisation [4]. Having

$$D(\boldsymbol{x}_i) = \min_j d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \min_j \left\{ (\boldsymbol{x}_i - \boldsymbol{\theta}_j)^2 \right\},$$

the algorithm may be described as follows:

1. Centroid $\boldsymbol{\theta}_1$ is initialised uniformly from input space.

2. Centroids $\boldsymbol{\theta}_j, j = 2, ..., k$ are sampled from input space with probabilities given by $\frac{D(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_q} D(\boldsymbol{x}_q)}$. As new clusters are set, values $D(\boldsymbol{x}_i)$ are recalculated.

This algorithm ensures that centroids are likely to be more evenly spread than a fully randomised initialisation. This is due to the fact that the sampling probability $\frac{D(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_q} D(\boldsymbol{x}_q)}$ will be maximised in areas where $D(\boldsymbol{x}_i)$ is the highest, i.e. areas further from other centroids.

A.2.2 Metrics

As clustering is an unsupervised machine learning method, evaluation will be dependent on the circumstance of what data is available. In order to determine the quality of separation, we must first establish what information is available beforehand. The question is whether the aim is to measure known characteristics in the dataset or to measure some property entirely without external information. Recognising this difference, there exists two categories of clustering performance metrics or Cluster Validity Indices (CVIs): *external CVIs* and *internal CVIs* [103].

As our method is drawn from a dataset with ground truth information known a priori, our main focus will be directed at external CVIs. Some alternatives within the field of internal CVIs include the Calinski-Harabasz (CH) index [8]. This measure relates the Euclidean distances between cluster means with the global mean of Euclidean distances between clusters and their corresponding observations. Another internal CVI is the Davies-Bouldin (DB) index [18] which measures similarity between each cluster and its most similar ones [103].

A.2.2.1 Unsupervised Clustering Accuracy (ACC)

The first external CVI we will tackle is the unsupervised clustering accuracy (ACC). This index simply represents the best accuracy achievable when assign-

ing predictions to the set of labels.

$$\text{ACC} = \max_{m \in M} \frac{1}{N} \sum_{i=1}^{N} \delta(m(\hat{y}_i), y_i), \tag{14}$$

where M is the set of all mappings from the labels to each cluster by utilising the Hungarian algorithm [59], and $\delta(\cdot)$ is the Kronecker delta function. The Hungarian algorithm aims to maximise the weight-matching in a matrix — in our case, alter the prediction classes to maximise the number of accurate clusters. This is performed as unsupervised methods have no way of knowing the correct class number. Due to this class assignment, the minimum clustering accuracy of any balanced dataset with $|C|$ different classes will always be $1/|C|$.

A.2.2.2 Normalised Mutual Information (NMI)

The normalised mutual information (NMI) is a measure of mutual dependence between two variables. A high NMI suggests that the ground truth labels are explained by the cluster memberships which indicates a high-grade clustering with regards to recognising some underlying ground truth distribution. NMI is defined as [110]

$$\text{NMI} = 2\frac{I(\hat{\boldsymbol{y}}, \boldsymbol{y})}{H(\hat{\boldsymbol{y}}) + H(\boldsymbol{y})}, \tag{15}$$

where $I(\cdot, \cdot)$ denotes the mutual information and $H(\cdot)$ denotes the entropy of $\boldsymbol{y}$, respectively [110]

$$I(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{k} \sum_{j=1}^{k} P_{ij}^{\hat{\boldsymbol{y}}, \boldsymbol{y}} \log \frac{P_{ij}^{\hat{\boldsymbol{y}}, \boldsymbol{y}}}{P_i^{\hat{\boldsymbol{y}}} P_j^{\boldsymbol{y}}} \tag{16}$$

$$H(\boldsymbol{y}) = -\sum_{i=1}^{k} P_i^{\hat{\boldsymbol{y}}} \log P_i^{\hat{\boldsymbol{y}}}, \tag{17}$$

with P denoting the mean occurrence of indices given by

$$P_i^{\hat{\boldsymbol{y}}} = \frac{1}{N} \sum_{l=1}^{N} \delta(\hat{\boldsymbol{y}}_l, i) \tag{18}$$

$$P_{ij}^{\hat{\boldsymbol{y}}, \boldsymbol{y}} = \frac{1}{n} \sum_{i=1}^{N} \delta(\hat{\boldsymbol{y}}_l, i) \delta(\boldsymbol{y}_l, j), \tag{19}$$

where $\delta(\cdot)$ is the Kronecker delta.

The geometric variation of the NMI metric is similar to eq. 15, but puts an increased emphasis on lower values by dividing by the geometric mean rather than the arithmetic one [105], as such:

$$\mathrm{NMI}_{geo} = \frac{I(\hat{\boldsymbol{y}}, \boldsymbol{r})}{\sqrt{H(\hat{\boldsymbol{y}})H(\boldsymbol{y})}}. \tag{20}$$

A.2.2.3 Confusion Matrix & the $F1$-score

The confusion matrix is a square matrix of data points' predictions along one axis and their labels along the other, e.g.

	Positive	*Negative*
Predicted positive	#TP	#FP
Predicted negative	#FN	#TN

(21)

Where TP, TN, FP, and FN are *true positive*, *true negative*, *false positive*, and *false negative*, respectively. The *true* denotes whether the prediction agrees with the label, and the *positive / negative* determines the class predicted. Confusion matrices may be applied in a multi-class scenario as well. Many common metrics may be calculated from the confusion matrix such as [103] accuracy

$$\mathrm{ACC} = \frac{\#\mathrm{TP} + \#\mathrm{TN}}{\#\mathrm{FP} + \#\mathrm{FN}},$$

precision

$$P = \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FP}},$$

recall

$$R = \frac{\#\mathrm{TP}}{\#\mathrm{TP}\#\mathrm{FN}},$$

and lastly the $F1$ score, being a balance between the two latter

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}.$$

Similarly to previous metrics, the Hungarian algorithm [59] is utilised to best map predictions to classes.

In the multi-class case[3], variants of the $F1$-score may differ by their averaging method. We will consider the *macro* $F1$-score — due to ambiguity around the macro-$F1$ formula, it is also known as the *averaged* $F1$ [77]. It is given as the arithmetic average of $F1$-scores for all classes:

$$F1_{macro} = \frac{\sum_{C_i \in C} F1_{C_i}}{|C|}. \tag{22}$$

3. Multi-view variations of the $F1$ consider each class C_i, denoted $F1_{C_i}$, considering negatives by every class C_j where $j \neq i$.

A.2.2.4 Adjusted Rand Index (ARI)

The Rand score is a between-sample metric rather than a class-based metric such as the confusion matrix metrics.

$$\text{RI} = \frac{\#\text{SS} + \#\text{DD}}{\#\text{SS} + \#\text{DD} + \#\text{SD} + \#\text{DS}}. \tag{23}$$

Having predictions $P_{\boldsymbol{x}_i}$ and $P_{\boldsymbol{x}_j}$, and actual classes $C_{\boldsymbol{x}_i}$ and $C_{\boldsymbol{x}_j}$, a pair of data vectors $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is defined as [103]:

- *SS* if $P_{\boldsymbol{x}_i} = P_{\boldsymbol{x}_j}$ and $C_{\boldsymbol{x}_i} = C_{\boldsymbol{x}_j}$.
- *DD* if $P_{\boldsymbol{x}_i} \neq P_{\boldsymbol{x}_j}$ and $C_{\boldsymbol{x}_i} \neq C_{\boldsymbol{x}_j}$.
- *SD* if $P_{\boldsymbol{x}_i} \neq P_{\boldsymbol{x}_j}$ and $C_{\boldsymbol{x}_i} = C_{\boldsymbol{x}_j}$.
- *DS* if $P_{\boldsymbol{x}_i} = P_{\boldsymbol{x}_j}$ and $C_{\boldsymbol{x}_i} \neq C_{\boldsymbol{x}_j}$.

We defined n_{ij} as the number of common objects (predictions and classes) for pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e. $n_{ij} = 2$ for *SS*, n_{ij} for *SD* and *DS*, and $n_{ij} = 0$ for *DD*. We also have that $n_{i\cdot} = \sum_j n_{ij}$. Now, when correcting for chance, the Rand Index of eq. 23 may extend to the Adjusted Rand Index (ARI) [42]:

$$\begin{aligned}\text{ARI} &= \frac{\text{RI} - \mathbb{E}[\text{RI}]}{\text{Max. RI} - \mathbb{E}[\text{RI}]} \\ &= \frac{\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}\right] - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}/\binom{n}{2}},\end{aligned} \tag{24}$$

where the binomial coefficient $\binom{k}{2}$ is defined as 0 for $k = 0$ and 1. The metric spans from -1 to 1 and has a value of 0 for completely random predictions and 1 for perfect predictions.

A.3 Experiments

A.3.1 Reconstruction Loss Setup

To assess the usefulness of reconstruction loss in DRAGMVC, we run trial experiments using the weighted Frobernius norm for both feature and graph reconstructions:

$$\mathcal{L}_{re,st} = \frac{1}{N}\sum_{i=1}^{V} w_i ||\boldsymbol{X}_i - \hat{\boldsymbol{X}}_i||_F^2$$

$$\mathcal{L}_{re,ft} = \frac{1}{N}\sum_{i=1}^{V} w_i ||\boldsymbol{A} - \hat{\boldsymbol{A}}_i||_F^2.$$

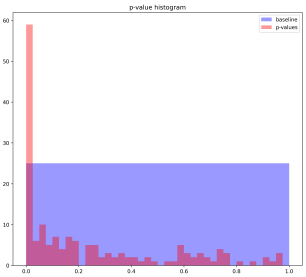
The graph decoder were constructed using tied weights (as in [13]). View encoders were created by using the direct inverse of the encoder (with separate weights), exchanging pooling with bilinear upsampling and convolution operations with transposed convolutions. Batch normalisation and activation functions were employed as in the encoder, merely altering the order such that they follow the main operation. Reconstructed graphs $\hat{\boldsymbol{A}}_i$ are computed by inner product following eq. 4.22.

Table A.1: Analysing the effects of reconstruction loss on accuracy and contrastive iteration loss for the MIMIC dataset. All runs had $\delta = 6.0$. The numbers of samples taken are (from top to bottom) 72, 65, 53, and 151.

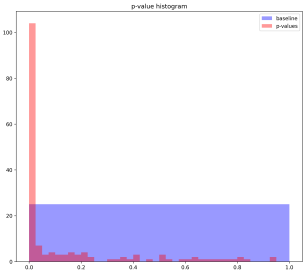
Metrics	ACC		$\mathcal{L}_{contrast}$		
Added loss terms	Max.	Mean	Min.	Mean	C.I.
$\mathcal{L}_{re,st}$	0.638	0.568	3.331	4.440	(4.109, 4.771)
$\mathcal{L}_{re,st} + \mathcal{L}_{re,ft}$	0.664	0.568	3.365	4.219	(3.955, 4.483)
$\mathcal{L}_{re,ft}$	0.686	0.568	3.317	4.433	(4.235, 4.631)
None	**0.696**	**0.589**	**1.791**	**2.911**	(2.823, 2.998)

A.3.2 p-value histograms

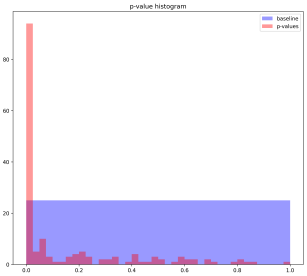
Un-adjusted p-value histograms to assess the skewness of significant values. A flat distribution is expected for true null hypotheses, as p-values naturally falls within the entire range of probabilities, i.e. $H_0 : p \in (0, 1) \sim U(0, 1)$.



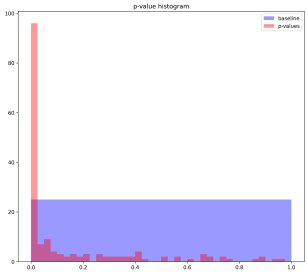
(a) Model 1 p-values.



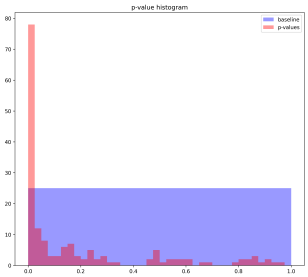
(b) Model 2 p-values.



(c) Model 3 p-values.

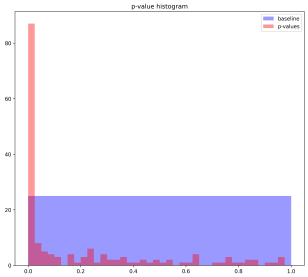


(d) Model 4 p-values.

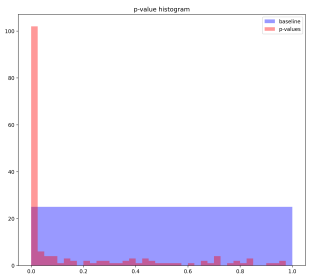


(e) Model 5 p-values.

Figure A.2: DRAGMVC p-value histograms.



(a) Model 4 p-values.



(b) Model 5 p-values.

Figure A.3: CoMVC models p-value histograms.

A.4 Dataset

A.4.1 Experiment Tables

Table A.2: Words removed from the word cloud visualisation tool in section 7.

Words removed from word cloud
unspecified
other
without mention
and
of
the
with
in
to
for
at
or

A.4.2 MIMIC Tables

The database *mimic-core* (MIMIC-IV) consists of the tables:

- *admissions*
- *patients*
- *transfers*

The database *mimic-hosp* (MIMIC-IV) consists of the tables:

- *d_hcpcs* (excluded from figure)
- *d_icd_diagnoses*
- *d_icd_procedures* (excluded from figure)
- *d_labitems*
- *diagnoses_icd*
- *drgcodes* (excluded from figure)
- *emar* (excluded from figure)
- *emar_detail* (excluded from figure)
- *hcpcsevents* (excluded from figure)
- *labevents*
- *microbiologyevents*
- *pharmacy*
- *poe*
- *poe_detail* (excluded from figure)
- *prescriptions*
- *procedures_icd* (excluded from figure)
- *services* (excluded from figure)

The database *mimic-icu* (MIMIC-IV) consists of the tables:

- *chartevents*

- *d_items*
- *datetimeevents*
- *icustays*
- *inputevents*
- *outputevents*
- *procedureevents*

The database *mimic-cxr* (MIMIC-CXR) consists of the tables:

- *chexpert*
- *dicom_metadata_string*
- *record_list*
- *study_list*

The database *mimic-derived*[4] (MIMIC-IV) consists of the tables:

- *age*
- *antibiotic*
- *apsiii*
- *bg*
- *blood_differential*
- *cardiac_marker*
- *charlson*
- *chemistry*
- *coagulation*
- *complete_blood_count*
- *creatinine_baseline*
- *crrt*
- *culture*
- *dobutamine*
- *dopamine*
- *enzyme*
- *epinephrine*
- *first_day_bg*
- *first_day_bg_art*
- *first_day_gcs*
- *first_day_height*
- *first_day_lab*
- *first_day_rrt*
- *first_day_sofa*
- *first_day_urine_output*
- *first_day_vitalsign*
- *first_day_weight*
- *gcs*
- *height*
- *heparin*
- *icp*
- *icustay_details*
- *icustay_hourly*
- *icustay_times*
- *inflammation*
- *invasive_line*
- *kdigo_creatinine*
- *kdigo_stages*
- *kdigo_uo*
- *lods*
- *meld*
- *milrinone*
- *neuroblock*
- *norepinephrine*
- *nopepinephrine_equivalent_dose*
- *oasis*
- *oxygen_delivery*
- *phenylephrine*

4. excluded from figure due to their information being drawn from other tables.

- *rhythm*
- *rrt*
- *sapsii*
- *sepsis3*
- *sirs*
- *sofa*
- *suspicion_of_infection*
- *suspicion_of_infection_v2*
- *urine_output*
- *urine_output_rate*
- *vasoactive_agent*
- *vasopressin*
- *ventilation*
- *ventilator_setting*
- *vitalsign*
- *weight_durations*