Unsupervised Representation Learning for Data Clustering

Mohammadreza Sadeghi,

Department of Electrical and Computer Engineering

McGill University, Montreal, QC, Canada

Supervisor: Professor Narges Armanfard

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

February 7, 2025

Abstract

Unsupervised representation learning is a critical component in clustering, enabling data grouping without labeled supervision. This thesis introduces innovative frameworks to address key limitations in existing deep clustering methods, such as premature cluster assignments, shared loss functions and/or latent space across clusters, and insufficient leveraging of pairwise relationships.

We introduce Deep Multi-representation Learning (DML), which leverages cluster-specific autoencoders to provide tailored latent spaces for diverse cluster characteristics. To mitigate the instability caused by crisp assignments, DML employs a novel soft assignment mechanism during training, enabling robust and adaptable clustering. Building on DML, we propose Deep Clustering with Self-Supervision (DCSS), a memory-efficient framework that integrates cluster-specific losses and pairwise similarity learning into a single autoencoder architecture. This ensures that learned representations are both compact and semantically enriched, advancing clustering effectiveness in complex data distributions. To further enhance clustering performance, we introduce Cross-instance guided Contrastive Clustering (C3), a novel framework that refines contrastive learning by leveraging cross-instance relationships. C3 increases positive pair accuracy while reducing the impact of false negatives, offering scalability and robustness for large-scale datasets. Extending clustering to streaming data, we present the novel concept of Unsupervised Continual Clustering (UCC), an innovative concept designed to cluster evolving data streams without access to labels. Within UCC, we propose a novel Forward-Backward Knowledge Distillation (FBCC) framework to mitigate catastrophic forgetting—the tendency of a neural network to lose previously learned information when trained on new data— by enabling effective knowledge retention and adaptation across tasks.

These frameworks collectively advance the state of the art in unsupervised learning and continual learning, addressing critical gaps and offering innovative solutions validated through extensive experiments on benchmark datasets.

Abrégé

L'apprentissage non supervisé de représentations constitue un pilier essentiel du clustering, puisqu'il permet de regrouper des données sans recourir à une supervision étiquetée. Cette thèse introduit des approches novatrices afin de pallier certaines limites majeures des méthodes de clustering profond existantes, notamment les affectations de clusters prématurées, l'utilisation d'une fonction de perte et/ou d'un espace latent partagés entre plusieurs clusters, ainsi que l'exploitation insuffisante des relations par paires.

Nous présentons tout d'abord le Deep Multi-representation Learning (DML), qui s'appuie sur des autoencodeurs spécifiques à chaque cluster pour offrir des espaces latents adaptés aux caractéristiques variées des différents clusters. Pour atténuer l'instabilité causée par des affectations trop rigides, DML met en œuvre un nouveau mécanisme d'assignation floue pendant l'entraînement, rendant le clustering plus robuste et adaptable. En nous appuyant sur DML, nous proposons ensuite le Deep Clustering with Self-Supervision (DCSS), un cadre économe en mémoire qui intègre simultanément des pertes spécifiques aux clusters et l'apprentissage de similarités par paires au sein d'une architecture unique d'autoencodeur. Cela garantit que les représentations apprises soient à la fois compactes et sémantiquement riches, améliorant ainsi l'efficacité du clustering dans des distributions de données complexes. Pour renforcer encore les performances de clustering, nous introduisons le Cross-instance guided Contrastive Clustering (C3), un nouveau cadre qui affine l'apprentissage contrastif en tirant parti des relations entre instances. C3 accroît la précision des paires positives tout en réduisant l'effet des faux négatifs, offrant ainsi une meilleure évolutivité et une robustesse accrue pour les jeux de données de grande taille.

Pour étendre le clustering aux flux de données, nous proposons le concept novateur de Unsupervised Continual Clustering (UCC), conçu pour traiter et regrouper des flux de données évolutifs sans accès aux étiquettes. Dans ce contexte, nous introduisons un nouveau cadre de Forward-Backward Knowledge Distillation (FBCC) destiné à pallier l'oubli catastrophique—la tendance d'un réseau de neurones à perdre les informations préalablement acquises lors de l'apprentissage sur de nouvelles données—en permettant la rétention et l'adaptation efficaces des connaissances à travers diverses tâches.

Dans leur ensemble, ces approches font progresser l'état de l'art en apprentissage non supervisé et en apprentissage continu, comblant des lacunes cruciales et proposant des solutions innovantes validées par de nombreuses expériences menées sur des jeux de données de référence.

Acknowledgements

I am profoundly grateful to my supervisor, Professor Narges Armanfard, for her unwavering guidance, commitment, and insightful advice. Working with her has been an incredible learning experience, providing valuable lessons and inspiration. Her mentorship has profoundly influenced my personal and academic growth. I extend my thanks to my committee members, Prof. James Clark and Prof. Ioannis Psaromiligkos, for their valuable feedback and suggestions during my dissertation proposal and seminar.

I would like to extend my sincere thanks to Fonds de recherche du Québec (FRQNT) for providing the funding for my PhD, and to ComputeCanada for offering the computational resources necessary for my research.

I am also thankful to my iSMART Lab teammates: Bahar Nikpour, Hadi Hojjati, Ali Karami, Zihan Wang, Thi Kieu Khanh Ho, Dimitrios Sinodinos, Jack Wei, and Thomas Lai. Special thanks to Hadi and Zihan for the enjoyable collaborations and insightful research discussions.

I appreciate my friends—Bahar, Majid, Haj Reza, Mamzi, Ali the Small, Nima, Emad, Mahan, Sara, Ayeh, Ali the Great, Ali Raoofian, Arikush, and others—who made my Ph.D. journey unforgettable. Your companionship and the fun activities we shared have been invaluable.

I am deeply thankful to my parents, Farhad and Masoomeh, for their endless love and support. Your unwavering support and encouragement have driven me to progress. You exemplify how to balance a successful career and family life, achieving your professional aspirations while being exceptional parents. I am grateful for your constant belief in me and your motivation to strive for more. I also want to extend my thanks to my brothers, Hamidreza and Navid, who are an integral part of me and my cherished childhood memories. Finally, I would like to thank my uncle Mohammadjavad for his support and valuable consultations during my study years.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAE	Adversarial Autoencode
ACC	Clustering Accuracy
AE	Autoencoder
ARI	Adjusted Rand Index
C3	Cross-instance guided Contrastive Clustering
CaSSLe	Continual Self-supervised Learning
CC	Contrastive Clustering
CCL	Continual Contrastive Learning
CF	Catastrophic Forgetting
CL	Continual Learning
CNN	Convolutional Neural Network
Co^{2}L	Contrastive Continual Learning
DCN	Deep Clustering Network
DCSS	Deep Clustering with Self-Supervision
DEC	Deep Embedding Clustering

DKM	Deep k-Means
DML	Deep Multi-representation Learning
DSL	Deep Subspace learning
Evolve	Enhancing Continual Learning with Multiple Expert
FBCC	Forward Backward Continual Clustering
GANMM	GAN Mixture Model
IDEC	Improved Deep Embedding Clustering
KD	Knowledge Distillation
KL	Kullback–Leibler
LUMP	Lifelong Unsupervised Mixup
MNet	Mutual Network
NMI	Normalized Mutual Information
OCD-Net	Online Contrastive Distillation Network
POCON	Plasticity-Optimized Complementary Networks
SCALE	Self-supervised Continual Learning
SCL	Supervised continual Learning
SDkC	Scalable Deep k-Subspace Clustering
SimCLR	Simple Contrastive Learning Framework
STAM	Self-Taught Associative Memory
t-SNE	t-Distributed Stochastic Neighbor Embedding

UCC	Unsupervised Continual Clustering
UCL	Unsupervised Continual Learning
VaDE	Variational Deep Embedding

Chapter 1

Introduction

1.1 Clustering

In many scientific and practical scenarios, obtaining category labels for data samples is either infeasible or sprohibitively expensive. Clustering serves as a fundamental tool in data analysis, pattern recognition, and machine learning to address this challenge. It enables the extraction of meaningful patterns and insights from unlabeled data by grouping data points based on predefined similarity metrics. Traditional clustering algorithms, such as k-means [2] and fuzzy c-means [3], operate directly in the original data space, which is often high-dimensional and noisy. However, methods like spectral clustering [4] represent a shift toward identifying latent representations of data samples before applying clustering, allowing better handling of complex data structures and intrinsic relationships. This progression in methodology is driven by the need to address limitations in conventional clustering approaches, such as sensitivity to noise and poor performance in high-dimensional spaces, ensuring more robust and versatile applications across diverse datasets. With the advancements in deep learning, deep learning-based clustering methods have demonstrated remarkable effectiveness across a wide range of applications, including image segmentation [5], social network analysis [6], face recognition [7, 8], and machine vision [9, 10]. These methods leverage the power of deep neural networks to map the original feature space onto a lower-dimensional latent space, where similar samples are grouped more effectively. This latent space transformation not only reduces noise but also enhances the interpretability of the clustering results and facilitates the discovery of the data's intrinsic structure.

Autoencoders (AEs) [11–15] represent a widely-used approach for dimensionality reduction in deep clustering. An AE consists of two parts: an encoder that compresses the input data into a concise latent representation and a decoder that reconstructs the original input from this compressed form. By minimizing reconstruction loss, AEs learn a non-linear, lower-dimensional representation that preserves the most critical features of the original data. Traditional AE-based clustering methods, such [16], involve applying a clustering algorithm like k-means [2] to the latent representation generated by the AE. However, recent advancements in AE-based methods, such as those presented in [15, 17–19], take this concept further by jointly optimizing dimensionality reduction and clustering within the AE framework. These integrated approaches improve clustering performance by leveraging the representation learning and clustering objectives.

Building upon the principles of AE-based clustering, self-supervised learning has emerged as a comprehensive framework that eliminates the dependency on explicit labels for representation learning. This approach leverages the intrinsic structure of data by employing pretext tasks that generate pseudo-labels or learning objectives from the data itself. Examples of such tasks include predicting the rotation of data samples, where the model learns to recognize and correct the orientation of input data [20], or predicting relationships between augmentation of data points [21, 22]. These pretext tasks act as a form of guidance, encouraging models to uncover meaningful patterns and relationships within the data without any external supervision. By integrating these representation learning strategies with clustering objectives, self-supervised methods significantly advance the capabilities of traditional AE-based techniques [23, 24]. While autoencoders focus on learning a compact latent space, self-supervised approaches enhance this process by ensuring the latent representations are not only compact but also semantically enriched.

While deep clustering methods have demonstrated impressive advancements, they still face notable challenges that limit their effectiveness in certain scenarios. To the best of our understanding, all current deep clustering approaches are affected by at least one of the following issues, which we aim to address through the innovative frameworks presented in Chapters 3, 4, and 5.

1- One major issue is the **crisp assignment problem** during training, where data points are assigned to clusters prematurely. Many existing algorithms, such as [19, 25–29], approximate data clustering losses by first performing a crisp cluster assignment and then calculating clustering losses using the crisply clustered data based on criteria like compactness or density within clusters. Crisp assignment (as opposed to soft assignment) assigns a data point to only one cluster, typically the one with the closest center. Since representations are still evolving, this can cause instability, making it difficult for the model to converge effectively.

2- Another major limitation is the reliance on a shared loss function and/or a shared latent space for all clusters. This uniform approach fails to account for the inherent differences between clusters in real-world datasets, where clusters often vary in variance, density, or feature distribution. For example, in a dataset of vehicle images, clusters representing compact cars and trucks may exhibit substantial differences in inter-cluster variability in shapes, sizes, and designs. Ignoring these differences can result in clustering outcomes that do not accurately represent the true structure of the underlying data. To our knowledge, the only method that explicitly incorporates cluster-specific losses and representations is [26]. However, this approach is computationally expensive, as it requires training K separate AEs, where K denotes the number of clusters.

3- Most clustering algorithms **neglect pairwise relationships** between samples, missing out on valuable information about the underlying data structure. While a limited number of deep clustering methods, such as [30–33], incorporate pairwise relationships in an unsupervised setting, they still face notable challenges. For instance, approaches like [30, 32] suffer from high numbers of false positive pairs during training. This issue arises because their loss functions progressively include more sample pairs without focusing on highly confident ones, introducing noise into the clustering process and impacting overall performance. Pairwise information has proven effective in supervised and semi-supervised learning frameworks. For example, metric learning techniques optimize a distance metric by bringing similar samples (belonging to the same class) closer together while increasing the distance between dissimilar samples (from different classes) [34–38]. These methods rely on pairwise relationships guided by class labels, yielding significant improvements in tasks like classification and recognition. Recently, contrastive learning has emerged as a state-of-the-art self-supervised representation learning approach [21]. By bringing augmented views of the same data point closer in latent space and pushing apart other points, contrastive learning has achieved remarkable success in tasks such as image classification [21], anomaly detection [39], and object recognition [40]. Inspired by these results, studies like [23,24] have adapted contrastive learning to clustering, demonstrating significant improvements over traditional methods. These methods primarily use data augmentation to generate pairs, assuming augmented views of the same instance are positive and different instances are negative. However, this approach overlooks deeper semantic relationships across instances. As a result, semantically similar instances may be misclassified as negatives (false negatives), and accurately identifying similar instances (true positives) may be underrepresented in the clustering process.

Research Question: The research question explored in Chapters 3, 4, and 5 is whether a novel framework can be developed to address the critical limitations of existing deep clustering methods, thus improving their performance, stability, and scalability. Specifically, the proposed frameworks aim to tackle three major challenges: (1) the instability caused by the crisp assignment problem, where data points are prematurely and rigidly allocated to clusters during training; (2) the assumption of a shared loss function and latent representation across clusters, which fails to account for the diverse characteristics and inherent variability of real-world clusters; and (3) the inadequate utilization of pairwise relationships, which are essential for understanding complex data structures but are often overlooked in current approaches. By overcoming these limitations, this research aspires to advance deep clustering methodologies and expand their applicability to diverse real-world scenarios.

1.1.1 Contribution to Original Knowledge of Clustering

The primary contribution of this thesis lies in advancing clustering methodologies by introducing novel approaches to address the limitations of existing deep clustering techniques. Below is a summary of the key contributions made in each chapter:

1- In Chapter 3, we propose Deep Multi-representation Learning (DML), a novel framework for unsupervised data clustering designed to address the limitations of existing methods. Traditional clustering techniques typically rely on a single latent space to represent all clusters, which can hinder their ability to effectively separate overlapping or complex data groups. In contrast, DML leverages multiple AEs to create specialized latent spaces tailored to cluster-specific characteristics. Specifically, DML assigns cluster-specific latent spaces to ”difficult” clusters, where cluster centers are closely spaced, enabling better separation and representation. For ”easy” clusters that are well-separated, DML utilizes a shared latent space, efficiently handling simpler cases while reserving specialized resources for more challenging scenarios. To train multiple AEs, DML introduces an innovative loss function that combines weighted reconstruction and clustering losses. The weights dynamically assign higher importance to data points strongly associated with specific clusters, ensuring that the model learns meaningful and tailored representations. Unlike traditional approaches that use crisp assignments during training, DML employs soft assignments, which focus on high-confidence samples to define cluster-specific losses. This strategy enables DML to improve clustering performance significantly, as evidenced by evaluations on various benchmark datasets, demonstrating better separation and compactness of clusters compared to state-of-the-art methods. Despite its strengths, DML has two notable limitations. First, it does not consider the pairwise relationships between samples, which have been shown to be essential for enhancing clustering performance [31, 41]. Second, DML is memory-inefficient due to its reliance on training multiple AEs, each of which may contain millions of parameters, thereby increasing computational overhead. DML idea [42] is published in IEEE Transaction on Neural Networks and Learning Systems (TNNLS).

2- In Chapter 4, we propose a novel framework, Deep Clustering with Self-Supervision using Pairwise Similarities (DCSS). This two-phase approach introduces innovative strategies to significantly enhance clustering effectiveness while addressing the limitations of DML. In the first phase of DCSS, we introduce cluster-specific losses, enabling the model to focus on reconstruction and centering for individual clusters. This results in hypersphere-like representations in the latent space, where each cluster is well-separated and compact. Unlike DML, which trains multiple AEs for different clusters, DCSS achieves memory efficiency by training a single AE using a carefully designed cluster-specific loss function. In the second phase, we introduce a novel approach that utilizes additional layers and an auxiliary network, MNet, to refine representations further. MNet handles pairwise relationships, strengthening similarities between similar data points and weakening dissimilar ones. This refinement ensures that the learned representations are well-suited for clustering tasks, enhancing performance and adaptability to complex data distributions.

3- In Chapter 5, we introduce the novel Cross-instance guided Contrastive Clustering (C3) method, designed to enhance contrastive clustering by leveraging cross-instance relationships. Rather than relying solely on data augmentation, C3 identifies semantically similar instances within the dataset by analyzing their instance-level representations in the feature space. This approach enables the method to more effectively identify potential positive pairs. Furthermore, C3 incorporates a novel weighting mechanism to refine the selection of negative samples. This strategy ensures that negative pairs are chosen to mitigate the impact of false negatives, noisy data, and anomalies during training. C3 is well-suited for the clustering of large-scale datasets, offering improved scalability and robustness in handling complex, high-dimensional data. C3 idea [43] is published in the 34th British Machin Vision Conference (BMVC).

1.2 Continual learning

In real-world scenarios, data is often encountered in a sequential manner, with new information becoming available over time. Traditional machine learning models struggle with this setting because they are designed to learn from static datasets and typically require retraining from scratch when exposed to new data. This process is computationally expensive and often impractical, particularly when previous data cannot be retained due to memory limitations, privacy regulations, or ethical concerns. Continual Learning (CL) addresses these challenges by enabling models to incrementally learn from new data while retaining knowledge from previous tasks. This aligns with how humans learn, allowing systems to adapt to changing environments, handle evolving data distributions, and operate efficiently without constant access to the entire dataset. CL is particularly relevant in scenarios where dynamic, non-stationary data streams are prevalent, ensuring that AI systems remain adaptive, scalable, and efficient. CL has use in many practical applications such as autonomous driving systems [44], healthcare [45], personalized recommendation systems [46], and robotics [47].

The main challenge of all of the CL approaches is "Catastrophic Forgetting" (CF), where learning new tasks leads to the model forgetting previously learned ones [48]. To address CF, some methods [49–51] focus on safeguarding critical model parameters during the learning process for new tasks. This is achieved by introducing a regularization term into the loss function, designed to minimize changes to parameters deemed essential for previously learned tasks. Regularization-based approaches face two key challenges: 1- Excessive emphasis on maintaining prior knowledge through strong regularization can impair the model's ability to learn new tasks effectively. 2- Calculating and storing importance measures for a large number of parameters can be computationally demanding, posing scalability concerns.

An alternative strategy for mitigating CF is the implementation of a memory buffer, as discussed in [52–54]. This technique involves retaining a subset of previously encountered data, enabling the model to revisit and strengthen prior learning during training. However, a significant challenge associated with this approach is the potential privacy concerns arising from storing and utilizing real data in the buffer, especially in applications involving sensitive information. To address this issue, alternative approaches [55–57] employ generative models to synthesize samples from earlier tasks. However, these methods face significant challenges, including the high computational cost of training generative models and the difficulty in achieving stable convergence to optimal solutions. Some other algorithms such as [58, 59] employ a technique known as knowledge distillation (KD) to transfer *insights* from previous tasks. This involves storing models from prior tasks in memory, leveraging them to assist the current model in retaining past knowledge. Nonetheless, these methods suffer from memory inefficiency, particularly with deep models containing millions of parameters. Additionally, solutions like [58] only retain a single previous task model in memory, potentially leading to forgetting when confronted with numerous tasks.

Within the field of Continual Learning (CL), three primary approaches have emerged based on the availability of data labels: Supervised Continual Learning (SCL): This approach relies on providing explicit class or task labels for data samples during training [60–62]. SCL focuses on sequentially learning from labeled data while maintaining performance across tasks. A practical example of SCL is in the healthcare domain, where models are designed to progressively learn from patient data collected over time [45]. 2- Semi-supervised Continual Learning (SeCL): This method combines the strengths of supervised learning (leveraging labeled data) and unsupervised learning (utilizing unlabeled data) to enhance performance over time while adapting to new tasks or evolving data distributions [63,64]. In SeCL, labeled data is used directly for task-specific learning to guide the model's predictions and unlabeled data helps in learning generalized representations that can transfer across tasks. 3- Unsupervised Continual Learning (UCL) [58,65]: Unlike SCL and SeCL, UCL focuses on learning new representations for unlabeled data streams evolving over time. These learned representations find utility in subsequent tasks such as data annotation. UCL holds practical significance in real-world applications, given the frequent unavailability of labeled data such as anomaly detection [66], autonomous driving [67], and healthcare [68]. While numerous UCL studies, such as those by [58, 65, 69], strive to continually refine representations for data, to our knowledge, there exists no continual learning algorithm explicitly tailored for clustering task.

Research Question: The research question addressed in Chapter 6 centers on the development of a novel concept of Unsupervised Continual Clustering (UCC), a scenario that has yet to be explicitly explored within the field of Continual Learning. This research aims to investigate the unique challenges posed by UCC, including issues related to evolving data distributions, the absence of labeled data, and the potential for CF in clustering tasks. Furthermore, the study seeks to design effective strategies to tackle these challenges, ensuring that UCC systems can learn robust representations, adapt to new data streams, and maintain clustering quality over time. By addressing these objectives, the research aspires to advance the state of the art in unsupervised continual learning, with implications for real-world applications where labeled data is scarce, and clustering is a critical task.

1.2.1 Contribution to Original Knowledge of Continual learning

In Chapter 6, we introduce the novel concept of Unsupervised Continual Clustering (UCC) within the realm of UCL, specifically designed for clustering tasks. UCC aims to discern clusters within sequentially arriving data across various tasks, with no overlap in cluster contents. This scenario parallels class-incremental continual learning [70], albeit without access to data sample labels. UCC is ideal for large dataset scenarios, enabling clustering models to identify and adapt to new clusters without forgetting previous ones. For instance, consider healthcare organizations collaborating on real-time image annotation for medical research. Each hospital specializes in a domain, such as cardiac imaging or neuroimaging, and cannot share data due to privacy rules. Traditional methods train separate models for each dataset, which are resource-intensive and lack cross-domain learning. UCC addresses these limitations by allowing a single model to continually learn across organizations without sharing data, ensuring both efficiency and privacy preservation. When a new hospital with an unlabeled dataset (e.g., lung imaging) joins, the UCC model incrementally adapts to learn new clusters while retaining prior knowledge, enabling efficient and integrative cross-domain insights.

The main challenge in UCC, similar to other continual learning approaches, is CF. In Chapter 6, we proposes Forward-Backward Knowledge Distillation (FBCC), which uses a deep "teacher" network and lightweight "student" models. In the Forward Knowledge Distillation phase, the teacher learns task-specific representations and clustering while retaining knowledge of previous tasks through guidance from specialized student models. In the Backward Knowledge Distillation phase, a student model is trained to replicate the teacher's behavior for the current task and is stored to help the teacher recall this task during future training. FBCC balances learning new tasks and retaining knowledge effectively.

1.3 Contribution of Authors

M. Sadeghi, and N. Armanfard. Deep multi-representation learning for data clustering. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 15675–15686, 2024.

- The novel concept of providing multiple latent spaces for data clustering was originally proposed by the supervisor (last author).

- The first author conducted an extensive literature review, implemented the framework and performed tests and experiments using benchmark datasets.
- The first and last authors collaboratively refined and enhanced the proposed idea and formulations to realize the idea, establishing it as a state-of-the-art approach.
- The first author drafted the initial manuscripts.
- The research paper underwent thorough revisions and editorial refinements by the last author.

M. Sadeghi, and N. Armanfard. Deep Clustering with Self-Supervision using Pairwise Similarities. *Under review*, 2024.

- The first and last authors collaboratively developed the methodologies and formulations to realize the novel concepts discussed in the paper, establishing it as a state-of-the-art approach.
- The first author conducted an extensive literature review, implemented the framework and performed tests and experiments using benchmark datasets.
- The first author drafted the initial manuscripts.
- The research paper underwent thorough revisions and editorial refinements by the last author.

M. Sadeghi, H. Hojjati, and N. Armanfard. C3: Cross-instance guided Contrastive Clustering. *The 34th British Machine Vision Conference (BMVC)*, 2023.

- The first author conducted an extensive literature review, implemented the framework and performed tests and experiments using benchmark datasets.
- The first and last authors collaboratively developed the methodologies and formulations to realize the novel concepts discussed in the paper, establishing it as a state-of-the-art approach.

- The first author drafted the initial manuscripts.
- The second author helped in the writing of the paper and provided valuable support in conducting the experiments.
- The research paper underwent thorough revisions and editorial refinements by the last author.

M. Sadeghi, Z. Wang, and N. Armanfard. Forward-Backward Knowledge Distillation for Continual Clustering. *Under review*, 2024.

- The novel concept of deep continual clustering was originally proposed and developed by the supervisor (the last author).
- The first and last authors collaboratively developed the methodologies and formulations to realize the novel concepts discussed in the paper, establishing it as a state-of-the-art approach.
- The first author drafted the initial manuscripts.
- The second author assisted with conducting experiments for other comparining algorithms and contributed to the writing of the manuscript.
- The research paper underwent thorough revisions and editorial refinements by the last author.

1.4 Overview

The remainder of this thesis is organized as follows: Chapter 2 provides a comprehensive review of the existing literature, starting with unsupervised representation learning and extending to clustering methods and continual learning. The focus is on analyzing various approaches and their strategies for addressing challenges in unsupervised learning, clustering, and continual learning. In Chapter 3, we introduce our proposed DML framework, which handles the training of multiple cluster-specific AEs. We discuss the architecture, its design considerations, and its approach to clustering. Experiments are presented to demonstrate the effectiveness of DML across different scenarios. In Chapter 4, we propose the DCSS framework, which addresses the limitations of DML by utilizing a single AE trained by cluster-specific loss and leveraging pairwise relationships between samples. The chapter includes detailed explanations of the framework and experiments to validate its performance improvements over existing methods. Chapter 5 focuses on improving contrastive clustering by introducing the novel C3 framework. C3 provides a new method for identifying potential positive pairs and incorporates a weighting mechanism designed to reduce the effects of false negatives, anomalies, and noisy samples. The effectiveness of C3 is validated through multiple experiments. In Chapter 6, we propose a new concept of unsupervised continual clustering (UCC) and address its challenges with the FBCC framework. Detailed methodologies are provided, along with experiments that showcase the effectiveness of FBCC in handling continual clustering scenarios. In chapter 7, the thesis concludes with a summary of the key findings and contributions. We discuss the implications of the proposed methods, their limitations, and potential avenues for future research.

Chapter 2

Literature Review and Background

In this chapter, we provide a comprehensive overview of the core research and methodologies that form the basis of this thesis. We begin by examining the advancements in unsupervised representation learning, outlining how various approaches have evolved to help machines effectively capture and interpret the underlying patterns in unlabeled data. From autoencoder-based methods to cutting-edge self-supervised learning models, we discuss the merits and drawbacks of each method, highlighting where the field currently stands. Next, we delve into the domain of clustering techniques, emphasizing their importance in grouping data points with shared characteristics. We cover both traditional clustering algorithms and more recent, deep learning–based strategies, detailing the improvements they offer in terms of handling high-dimensional and complex data. Finally, we turn our attention to continual learning, a growing area of research focused on enabling systems to learn from data that arrive in a sequential or incremental manner. We discuss key concepts such as catastrophic forgetting and outline a variety of methods developed to tackle these challenges. By summarizing the progress made in this field, we establish a foundation for understanding how and why our proposed contributions can further advance ongoing research.

Together, these sections provide a clear picture of the current landscape in unsupervised representation learning, clustering, and continual learning. They also help clarify how the new methods introduced in this thesis build upon and extend the state of the art, addressing the gaps and limitations revealed in our review.

2.1 Unsupervised Representation Learning

Unsupervised representation learning has emerged as a cornerstone in modern machine learning, focusing on the extraction of meaningful features from unlabeled data. This field has gained prominence due to its ability to uncover hidden patterns and structures without the need for manual annotations, which are often expensive and time-intensive to obtain. The techniques developed in this domain are instrumental in transforming raw, high-dimensional data into compact and informative representations, enabling downstream tasks like clustering, anomaly detection, and classification. These methods leverage advanced neural network architectures to capture the inherent complexity of data, ensuring robustness and generalizability across various applications.

A key focus in unsupervised representation learning is the design of models that learn from the intrinsic properties of the data itself. Autoencoder-based methods are foundational in this regard, offering a powerful framework for dimensionality reduction and reconstruction. Complementing these are self-supervised approaches that employ creative pretext tasks, such as predicting rotations or reconstructing obscured inputs, to generate pseudo-labels and guide the learning process. Together, these techniques address the challenge of learning rich, hierarchical representations while maintaining computational efficiency and scalability, paving the way for innovative applications in fields ranging from computer vision to bioinformatics. In the rest of this subsection, we will begin by exploring different autoencoder-based representation learning methods and their applications. We will then discuss a range of self-supervised learning techniques and examine how they are applied in practice.

2.1.1 Autoencoder-based Representation Learning

AEs have become a foundational aspect of deep learning, gaining widespread recognition for their role in facilitating unsupervised learning and extracting meaningful features from data. By transforming raw input into a compressed, lower-dimensional representation through their hidden layers, AEs enable the identification of key patterns and structures within the data. Their design focuses on transforming input data into a latent representation and then reconstructing it back to its original form, effectively reconstructing the input while discarding redundant or irrelevant information. This dual functionality makes autoencoders an essential tool for data compression and feature representation, particularly valuable in situations where labeled data is limited or unavailable. The origin of autoencoders can be traced back to their earlier version as "auto-associators," [71] a concept aimed at learning compact representations by reconstructing input data. This early version of autoencoders was primarily focused on capturing the essence of the input within a limited-dimensional hidden space. Their resurgence in popularity coincided with the rise of deep learning, where autoencoders offered a solution to some of the most pressing challenges in training deep neural networks. One such challenge was the difficulty of optimizing deep architectures, which often suffered from issues such as poor initialization [72]. Autoencoders addressed these issues through a process known as layer-wise pretraining [73]. This approach involved training each layer of a deep network sequentially and independently using an autoencoder, which provided well-initialized weights for subsequent layers. This method significantly improved the training process, ensuring better convergence and stability for deep models. Over time, autoencoders have evolved beyond their initial purpose of pretraining. They are now widely employed in various applications, including anomaly detection [74], denoising [75], dimensionality reduction [76], and generative modeling [77]. Their capacity to learn rich, hierarchical data representations without depending on labels has made them exceptionally flexible and effective in tackling complex challenges across numerous domains, ranging from image [78] and speech [79] processing to recommendation systems [80] and medical diagnostics [81]. Autoencoders continue to play a vital role in advancing the capabilities of deep learning, driving both theoretical advancements and practical application. Below, I outline the key categories of autoencoder AE-based approaches, each addressing specific challenges and enabling diverse applications:

Regulare autoencoders: a basic AE consists of two non-linear components: an encoder and a decoder. The encoder f_e maps the input data x_i into a lower-dimensional representation, denoted as $h = f_e(x_i)$. Subsequently, the decoder f_d reconstructs this low-dimensional representation h_i back into the original input space, producing $\hat{x}_i = f_d(h_i)$. An effective AE should ideally reconstruct the input data with minimal loss of information, which shapes its primary optimization objective. This objective is generally formulated as:

$$\mathcal{L} = \sum_i ||x_i - \hat{x}_i||^2 = \sum_i ||x_i - f_d(f_e(x_i))||^2 \tag{2.1}$$

In this context, $||\cdot||_2$ represents the L_2-norm, while $||x_i - \hat{x}_i||^2$ denotes the reconstructed data for the i-th instance.

Regular AEs serve as powerful tools for dimensionality reduction, providing notable benefits over traditional linear techniques like Principal Component Analysis (PCA) [82]. Unlike PCA, which is limited to linear transformations, autoencoders utilize neural networks with non-linear activation functions, allowing them to capture and represent complex, non-linear relationships in high-dimensional datasets. This capability makes autoencoders particularly well-suited for handling data with intricate patterns and structures that linear methods cannot effectively process. Regular AEs are widely utilized in image compression [83, 84], where they significantly reduce the dimensionality of image data while preserving high visual quality, making them ideal for storage and transmission. Additionally, they serve as an effective preprocessing tool for machine learning tasks [85], by extracting informative features that enhance the efficiency and accuracy of downstream algorithms, such as clustering and classification. Moreover, their capacity to capture and process non-linear relationships makes them highly effective for data visualization [86], enabling the projection of high-dimensional datasets into two or three dimensions to uncover hidden patterns and clusters.

Sparse autoencoders: Sparse autoencoders (SAEs) are an extension of regular AEs designed to learn meaningful and efficient data representations by incorporating a sparsity constraint on the hidden layer. This constraint ensures that only a small number of neurons in the hidden layer are activated for any given input, encouraging the model to focus on the most relevant features [87]. To achieve this sparsity, a regularization term, such as the Kullback-Leibler (KL) divergence or an ℓ_1 penalty [88], is added to the reconstruction loss. The KL divergence controls the average activation of hidden neurons, aligning it with a predefined sparsity level to keep most neurons inactive. Alternatively, the ℓ_1 penalty directly suppresses the activity of certain neurons, promoting sparsity in the parameter space. As a pioneer approach, [89] highlights two key aspects of feature distribution: population sparsity and lifetime sparsity. Population sparsity refers to cases where only a small number of neurons are active at a given time for any specific input, while lifetime sparsity describes a scenario where individual neurons activate selectively and infrequently across multiple inputs. [90] introduced the k-sparse autoencoder, which focus on population sparsity (limiting the number of active neurons per input). This type of AE employs a linear activation function and enforces sparsity in its hidden layers by retaining only the top k most active neurons, setting all other activations to zero. This selection process can be done by sorting activations or by using ReLU hidden units with adaptively adjusted thresholds to identify the top k activations. Unlike [90] that concentrate on population sparsity, [91] enforces lifetime sparsity by retaining only the top k% activations during training and uses corrupted inputs to improve noise robustness.

By combining non-linear feature extraction with constraints that promote sparse and interpretable representations, sparse autoencoders have become a valuable tool for a variety of machine learning applications. SAEs are particularly effective in tasks like image recognition [92], audio processing [93], and natural language processing [94], where they extract key features from high-dimensional data while eliminating redundancy and irrelevant information. Inspired by biological neural networks [95], where sparse activity improves efficiency and robustness, SAEs excel at discovering latent patterns within data. They are especially useful for over-complete representations [96], where the hidden layer has more neurons than the input, as they prioritize the most significant features by enforcing sparsity. This makes SAEs resilient to noise and capable of producing highly discriminative features.

Denoising Autoencoders: The Denoising Auto-Encoder (DAE) is a significant variant of the Auto-Encoder architecture, proposed by Vincent et al. in 2008 [97], as an enhancement to traditional feature extraction and unsupervised learning techniques. Unlike conventional Auto-Encoders, which aim to reconstruct inputs directly, DAEs introduce a noise-corruption process during training. This forces the model to reconstruct clean, uncorrupted inputs from their corrupted counterparts, thereby learning representations that are robust and generalizable. The corruption process is typically achieved through various forms of noise, such as additive Gaussian noise [98], salt-and-pepper noise [99], or masking noise [100]. The encoder maps the corrupted input to a latent representation, and the decoder attempts to reconstruct the clean version of the input. The training objective minimizes a reconstruction loss (e.g., mean squared error or cross-entropy) between the clean input and its reconstruction. Injecting noise into the input data is a fundamental aspect of a DAE. This concept can be extended by applying noise not only to the input (visible units) but also to the hidden units of a neural network. This extension forms the basis for a highly efficient yet effective regularization technique known as dropout [101].

DAEs have numerous practical applications in real-world scenarios. One significant use is in missing data imputation, as highlighted in [102], where it helps reconstruct incomplete datasets to enable effective analysis. Additionally, DAEs prove invaluable in domains where acquiring clean and noise-free data is particularly challenging. For instance, in remote sensing [103], DAEs are employed to process and enhance data collected from satellite imagery, which often contains noise or missing values due to environmental and technical factors. Similarly, in the field of medical imaging [98, 104], DAEs assist in improving image quality and filling gaps in medical scans, ensuring more accurate diagnoses and treatment planning. These applications underline the versatility and significance of DAEs in handling complex data challenges across diverse fields.

Variational autoencoders: The use of Variational Autoencoders (VAEs) as a subclass of autoencoders has emerged as a transformative approach within deep learning frameworks, particularly in representation learning and generative modeling. VAEs extend the traditional autoencoder architecture by introducing a probabilistic latent space, facilitating the generation of new data samples and fostering robust learning of underlying data distributions. VAEs operate under the assumption that data x are generated from a latent representation h and then aim to infer the posterior distribution $p(h|x)$ from the observed data. [105] is a prominent method in this category that utilizes variational inference techniques to maximize the evidence lower bound (ELBO) of the data likelihood:

$$\log p(x) \geq \mathbb{E}_{q(h|x)}\left[\log p(x|h)\right] - D_{KL}\left(q(h|x) \parallel p(h)\right) \tag{2.2}$$

where $D_{KL}(\cdot \parallel \cdot)$ indicates the KL-divergence between two probability distributions. The term $p(h)$ is the prior distribution over the latent variables, and $q(h|x;\phi)$ serves as the variational approximation to the true posterior (i.e., $q(h|x;\phi) \approx p(h|x)$), which can be parameterized by the recognition network with parameters ϕ.

[105] proposes optimizing the Evidence Lower Bound (ELBO) on the marginal likelihood as the loss function, which consists of two key components: a reconstruction loss (identifying how effectively the model reconstructs data from latent variables) and a regularization term (quantified as the KL divergence between the approximate posterior and the prior). This objective promotes efficient learning by fostering latent representations that are both informative about the data and consistent with the prior distribution. Building on this framework, [106] introduces a hyperparameter β to modify the balance between reconstruction accuracy and the disentanglement of latent factors. This unsupervised approach is particularly effective in identifying independent generative factors from raw data, enabling the model to develop more interpretable and structured latent representations. The Ladder Variational Autoencoder (LVAE) [107] enhances traditional VAEs by addressing the shortcomings of their solely bottom-up inference process. In a bottom-up approach, data is processed layer by layer, starting from raw inputs (such as image pixels) and progressively forming more abstract representations. While effective for certain tasks, this method lacks feedback from the model's overall understanding of the data, which is crucial for refining and correcting the developed representations. LVAE tackles this issue by incorporating a top-down feedback mechanism that integrates bottom-up signals with adjustments derived from the model's generative understanding. Conditional Variational Auto-Encoders (CVAEs) [108] are an extension of traditional VAEs designed to generate outputs conditioned on specific input data. Unlike standard VAEs that learn an unconditional latent representation of data, CVAEs incorporate additional inputs during both encoding and decoding processes, enabling them to model conditional distri-

butions. This makes CVAEs ideal for tasks such as image inpainting, where the model generates plausible completions of missing parts of an image based on the observed pixels, or other tasks like image colorization and super-resolution.

VAEs are highly versatile tools applied in several domains to solve complex problems. For instance, in missing data imputation [109, 110], VAEs can reconstruct incomplete datasets by leveraging their ability to model the underlying patterns of the data distribution. This ability is particularly crucial in domains where data completeness significantly impacts subsequent analysis and decision-making. In healthcare monitoring [111], VAEs contribute to enhancing patient care by analyzing and synthesizing medical data. They support the detection of abnormalities, facilitate personalized health tracking, and improve diagnostic accuracy by effectively handling high-dimensional and complex health datasets. Within bioinformatics [112, 113], VAEs are used to analyze intricate biological data, such as genetic sequences and protein structures. By learning latent representations, VAEs enable researchers to identify hidden relationships and generate biologically meaningful data, aiding advancements in genomics, proteomics, and other areas of biological research.

2.1.2 Self-Supervised Representation Learning

Self-supervised learning has revolutionized machine learning by allowing models to leverage large volumes of unlabeled data for training. This technique operates in two main steps: first, it defines a synthetic supervised task by generating targets for each input, effectively transforming the problem into a supervised learning framework. Second, it trains the model to map inputs to these generated targets using standard supervised learning methods. These targets, often created through predefined rules or neural networks, can evolve dynamically during training. Despite borrowing principles from supervised learning, self-supervised learning is inherently unsupervised, as it relies entirely on unlabeled data to uncover meaningful representations. This approach driven substantial progress across numerous domains. In computer vision, it is used for tasks like object detection [114] and medical imaging [115], where models pre-trained on unlabeled datasets are fine-tuned for specific applications. In natural language processing, self-supervised learning powers models like GPT [116] and BERT [117], providing applications such as translation, summarization, and conversational AI. In speech processing, methods like Wav2Vec [118] advance speech recognition and audio synthesis, while in robotics, self-supervised learning helps robots understand and navigate environments by generating training signals through interaction. Additionally, it is applied in recommendation systems [119] to predict user preferences without requiring extensive labeled data. By unlocking the potential of unlabeled data, self-supervised learning is transforming industries and redefining the boundaries of artificial intelligence.

It is essential to explore the key categories that define self-supervised methods. These categories encapsulate diverse approaches aimed at extracting meaningful representations from unlabeled data, each with unique strategies and goals. **Pretext task methods** utilize auxiliary tasks, like predicting image rotations or solving jigsaw puzzles, to guide models toward learning features that generalize to downstream tasks like object detection and classification. **Information maximization methods** ensure robust representations by preserving the richness and diversity of features while maintaining stability under transformations like cropping and color adjustments. **Teacher-student methods** employ a dual-network framework, where a teacher model generates stable representations that a student model learns to emulate, fostering invariance across diverse data augmentations. **Contrastive learning methods** train models to distinguish between similar and dissimilar pairs by pulling representations of similar data closer and pushing apart those of dissimilar data, enabling the creation of highly generalizable features. Together, these categories highlight the ingenuity and versatility of self-supervised learning, demonstrating its potential to revolutionize how machines learn from and understand unlabeled data. Below, we delve into each of these categories.

Pretext Task Methods: Pretext task methods [20, 120, 121] create auxiliary tasks to enable the model to learn representations that capture relevant features for downstream tasks. These pretext tasks guide the network to extract meaningful information from data without explicit labels. For instance, Rotation Network (RotNet) [20] trains a model to recognize the angle by which an image has been rotated (e.g., 0°, 90°, 180°, 270°). By solving this task, the model learns to identify spatial and semantic features within the image that generalize to tasks like object recognition and classification. [120] developed a method where an image is divided into patches that are randomly shuffled, and the model is tasked with predicting the correct order of these patches. This task encourages the network to learn spatial and contextual relationships within the image, which improves its ability to recognize complex patterns and structures. Masked AEs [121] extend the traditional autoencoder framework by masking sections of the image and training the model to predict or reconstruct the obscured areas. Typically leveraging Vision Transformers, Masked AEs encourages the model to concentrate on high-level structural and semantic aspects by requiring it to deduce missing information from the visible portions of the image.

Information Maximization Methods: Information maximization techniques [122–124] aim to create robust image representations that remain stable under a range of transformations, such as rotation, cropping, and color adjustments. By maintaining essential details of the image across transformations, these methods prevent the model from collapsing to trivial representations, where it might output identical, non-informative representations for different images. To address this, these methods are designed to maximize the distinctiveness and informational richness in the learned representations. For instance, Barlow twins [122] seeks to reduce repetitive or redundant information in feature representations. It does so by calculating the cross-correlation between representations of different augmented views of the same image. The approach applies a special type of loss that minimizes off-diagonal elements in this correlation matrix, ensuring that each feature dimension independently captures unique information. Barlow twins faces challenges such as representation collapse, where learned representations become trivial or identical, and redundancy across dimensions, leading to inefficient feature learning. [125] tackle these challenges by introducing Mixed Barlow Twins, which improves sample interactions during training using linearly interpolated samples. This method incorporates a regularization term into the original Barlow Twins objective, based on the assumption that linear interpolation in the input domain corresponds to linearly interpolated features in the representation space. VICReg [123] relies on three key regularization terms: variance, invariance, and covariance. Variance regularization ensures that embedding vectors maintain sufficient diversity by enforcing a minimum variance across dimensions, thus avoiding trivial solutions like collapsed embeddings. Invariance regularization aligns the embeddings of augmented views of the same image, ensuring robustness to data augmentations. Covariance regularization reduces redundancy by minimizing off-diagonal terms in the covariance matrix, encouraging each dimension of the embedding to capture distinct information. Unlike earlier methods, VICReg does not rely on negative samples, weight sharing, or memory banks, making it both simple and efficient. VICRegL [124] extends VICReg by incorporating local feature learning alongside global feature learning. While VICReg focuses on producing global image representations invariant to transformations, VICRegL simultaneously learns both global and local features. It applies the VICReg criterion to pairs of global feature vectors as well as pairs of local feature vectors before the final pooling layer.

Teacher Student Models: Teacher-student approaches [126–128] in self-supervised learning use a two-network framework, where one model (the "teacher") provides stable, consistent representations that guide another model (the "student") to learn effective representations. This setup is particularly useful in creating invariance across transformations, as the student model gradually learns to match the teacher's representations despite varying inputs. Bootstrap Your Own Latent (BYOL) [126] uses a teacher-student structure to prevent representational collapse without requiring negative samples. In BYOL, the student model learns to predict the representations produced by the teacher, and the teacher's parameters are updated via a moving average of the student's parameters. This method has shown that meaningful representations can emerge simply by aligning predictions between the student and teacher across different image augmentations. Distillation with No Labels (DINO) [127] takes a similar approach but with some modifications, including the use of Vision Transformers (ViTs) as its base architecture. Instead of directly matching representations, DINO applies a softmax to the teacher's output, creating soft targets for the student to match. This process, known as self-distillation, allows the student to learn finer details by aligning with the teacher's soft probability distributions. DINO also utilizes a multi-crop strategy, where the teacher sees only high-resolution "global" views, and the student sees both global and lowerresolution “local” views, promoting the learning of multi-scale features. [128] enhances the DINO framework by implementing a cluster-aware training strategy, which creates positive pairs using segments from different utterances within the same cluster instead of relying on segments from a single utterance. These clusters are formed through a clustering algorithm like k-means. This method boosts data diversity and reduces segment overlap, thereby decreasing the likelihood of the model attending to irrelevant details, such as content or channel properties, and directing its focus toward speaker-specific features.

Contrastive Learning: Contrastive learning [21,22,129,130] has emerged as a leading strategy in self-supervised learning because it can generate robust representations without relying on labeled data. Its primary mechanism is to train models to distinguish between pairs of data points that are similar (positives) and those that are dissimilar (negatives). By pulling positive pairs closer in the feature space and pushing negative pairs further apart, the model is encouraged to learn features that are both distinctive and broadly applicable. As a result, contrastive learning has proven highly effective in diverse downstream tasks, including classification, object detection, and cross-domain adaptation.

In contrastive learning, an anchor sample serves as a reference point, with a positive sample generated from it through transformations or augmentations that alter the appearance but retain core content. Common transformations include cropping, flipping, color changes, and other image augmentations that create different “views” of the same image. Negative samples, on the other hand, consist of unrelated samples within the batch or dataset. Through this setup, the model learns to associate augmented versions of an anchor as similar while recognizing other samples as distinct. This process is often formalized using the InfoNCE [131] loss, which structures the task as a classification problem. In this context, the model maximizes the similarity between an anchor and its positive counterpart, while minimizing similarity with negatives, often through cosine similarity or other similarity metrics. While larger numbers of negatives in a batch improve the quality of the learned features, memory and computational constraints often pose challenges. Methods like Momentum Contrast (MoCo) [22] address this by introducing a memory bank that stores previous representations, effectively enabling a large pool of negative samples without requiring large batch sizes. MoCo also uses a momentum-based encoder to update stored negatives gradually, ensuring consistency in the learned representations while keeping computational demands manageable. SimCLR [21] introduces a simple yet effective framework for contrastive learning of visual representations, designed to maximize the similarity between differently augmented versions of the same image while minimizing the similarity between these and other samples in the batch. It consists of three main elements: robust data augmentation techniques, a feature encoder (commonly a ResNet), and a projection head that transforms encoded features into a space optimized for contrastive learning. A major limitation of SimCLR is its reliance on large batch sizes to achieve optimal performance. This requirement stems from its contrastive loss, which relies on comparing positive pairs against a diverse set of negatives, achievable only with larger batches. Consequently, this approach significantly increases computational and memory demands, making it resource-intensive and less feasible for systems with limited hardware capabilities. Contrastive Predictive Coding (CPC) [131] expands contrastive learning to sequential data by making predictions about future states or segments within sequences, allowing contrastive learning techniques to apply not only to images but also to time-dependent data, such as audio and text. Contrastive Multiview Coding (CMC) [132] learns from multiple, distinct views of the same data, such as individual color channels, to build robust representations that capture shared, high-level features. By contrasting positive pairs from different views against other unrelated data, CMC encourages consistency across perspectives, making it especially useful for multimodal data. Pretext-Invariant Representation Learning (PIRL) [133] creates positive pairs by transforming images through tasks like jigsaw puzzles or rotations, then training the model to treat the original and transformed images as similar. This strategy drives the model to obtain representations that are invariant to such transformations, allowing it to focus on core semantic content instead of transformation-specific details.

2.2 Clustering

Clustering is a key method in data analysis and machine learning that organizes data points into groups based on shared characteristics. The fundamental idea is that points in the same group have greater similarity to each other than to points in other groups. As an unsupervised learning method, clustering operates without requiring labeled data, making it particularly useful when the structure or categories within the data are not predefined. The process depends heavily on defining measures of similarity, such as distance metrics, and selecting appropriate algorithms and parameters to accurately represent the natural organization of the data. The adaptability of clustering makes it applicable to both structured and unstructured data, offering significant value in areas such as natural language processing, image analysis, bioinformatics, market segmentation, and network studies. Additionally, it is a valuable tool for exploring data, enabling the discovery of hidden relationships, patterns, and trends that are not immediately apparent. By condensing large datasets into manageable and meaningful groupings, clustering improves interpretability, reduces data complexity, and provides a foundation for further analysis and informed decision-making. Its wide range of applications and ability to uncover insights across various fields highlights its importance as a versatile method for analyzing and understanding complex data.

2.2.1 Conventional Clustering Methods

K-means [2] is one of the most widely used clustering methods, applicable to a diverse range of problems. It groups data points into clusters by minimizing the sum of squared distances between each data point and its cluster's centroid, which is calculated as the arithmetic mean of all points in the cluster. Fuzzy C-Means (FCM) [134] is another conventional clustering technique that permits data points to associate with multiple clusters, each with a certain degree of membership. In contrast to other traditional methods like k-means, which allocate each point to a single cluster, FCM provides membership probabilities that indicate the extent to which a point is likely to belong to each cluster. A Gaussian Mixture Model (GMM) [135] is a statistical approach to clustering that models the dataset as a combination of several Gaussian distributions, where each distribution represents a specific cluster. GMM employs a probabilistic perspective, assigning a likelihood of belonging to each cluster for every data point. This is accomplished by estimating parameters such as the mean, covariance, and mixture weight of each Gaussian component, typically using the Expectation-Maximization (EM) algorithm. While these algorithms are fast and straightforward to implement, their performance diminishes in high-dimensional spaces. Additionally, clustering methods such as k-means and FCM struggle to deliver accurate results when data points are unevenly distributed around their centroids in the original feature space, leading to suboptimal clustering outcomes. To handle these difficulties, some algorithms such as [136, 137] perform subspace learning[1] (aka dimension reduction) to learn a low dimensional feature space (aka subspace, latent space) of data points and then employ the trained subspace for data clustering task. Another category of clustering algorithms, such as [4, 140], consider pairwise relationship between data points to embed the original high dimensional data points into a lower dimensional space and then apply k-means algorithm to the new space. These algorithms construct a weighted graph based on the relationships between the data points in the original space; they then solve an optimization problem based on Laplacian matrix of the graph. Although these methods outperform k-means, their computational cost for solving the optimization problem prevents their application when dealing with large datasets. Some studies, e.g. [141–143], try to handle this problem by using stochastic optimization methods. For example, [141] formulates an adaptive stochastic gradient optimization, which is linear in the number of data that does not need storing the complete Laplacian matrix. Although the stochastic optimization based algorithm could outperform the original ones even in some small datasets, these methods only consider linear transformation of the data.

2.2.2 Deep Neural Network based methods

Deep neural networks (DNNs) have been widely studied and utilized to address clustering tasks, leveraging their ability to learn complex and hierarchical representations of data. These methods aim to optimize a DNN model in an unsupervised manner, where the absence of labeled data poses

[1]Note that there is another branch in data clustering called Subspace Clustering [138, 139], which is different from subspace learning. Subspace clustering techniques assume that data points are drawn from multiple subspaces corresponding to different data clusters.

unique challenges and opportunities. Specifically, DNN-based clustering methods are designed to automatically discover meaningful patterns and groupings within datasets by extracting high-level features that preserve the intrinsic structure of the data. For example, [144] introduces a framework designed to jointly optimize the learning of deep representations and image clustering in an unsupervised manner. This method incorporates agglomerative clustering into a recurrent structure built on a CNN. During the training process, clusters of images are updated in the forward pass, while the backward pass refines the deep representations. A unified weighted triplet loss function guides this simultaneous optimization, improving both clustering accuracy and representation quality. As another example, rather than treating clustering as a traditional grouping problem, [145] reformulate it as a pairwise classification task, where the similarity between image pairs determines whether they belong to the same cluster. To overcome the challenge of limited labeled data in an unsupervised setting, the method employs an iterative learning algorithm that alternates between training the network and selecting samples to refine clustering. [146] address limitations in [145], particularly its sensitivity to spatial transformations such as scaling, rotation, and translation, by adding Spatial Transformer Networks (STNs). STNs act as a form of visual attention mechanism, enabling the network to learn invariance to scale, rotation, and broader image deformations. By doing so, the network can focus on extracting meaningful features that are invariant to these transformations, leading to more robust clustering results. Deep Clustering for Unsupervised Learning of Visual Features (DeepCluster) [27] presents a method that integrates clustering with deep learning to train CNN without labeled data. This method involves iteratively grouping image features through k-means clustering, where the resulting clusters are used as pseudo-labels to adjust the CNN's parameters. By repeating this process, the network gradually learns to extract discriminative features without requiring labeled data. [147] presents a flexible vision model that combines clustering mechanisms with a Transformer architecture to tackle a range of visual tasks such as image classification, segmentation, and object detection. The model introduces two main innovations: recurrent cross-attention clustering, which iteratively improves cluster centers for better representation learning, and feature dispatching, which leverages the updated cluster centers to reassign image features based on similarity-based metrics. [148] introduces a method to enhance and control the diversity of clustering results in deep learning frameworks. This approach utilizes a new loss function designed to manage the diversity between multiple clustering solutions, allowing flexibility in achieving desired diversity levels. This approach is computationally efficient and seamlessly integrates with existing deep clustering models.

2.2.3 AE-based methods

AE-based algorithms have emerged as powerful tools in unsupervised learning, particularly for tasks requiring dimensionality reduction and clustering. These algorithms utilize the capabilities of a deep autoencoder to transform high-dimensional input data into a compact, lower-dimensional representation, commonly known as the latent space. By performing this transformation, AE-based approaches allow downstream tasks, such as clustering, to be carried out more effectively in a simplified feature space. In traditional approaches to clustering, such as those described in works like [149] and [150], the process of dimensionality reduction is distinct from the clustering task itself. For instance, to make the learned representations more suitable for clustering, [149] introduce a locality-preserving constraint on the learned representations, designed to embed the original data into its intrinsic manifold space. After training the autoencoder and obtaining the reduced data representation, the k-means clustering algorithm is applied to partition the data into clusters. This two-step approach highlights the independent nature of dimensionality reduction and clustering in conventional AE-based methods. A significant application of these techniques is in graph clustering, which is a specialized domain of clustering aimed at partitioning the nodes of a graph into distinct groups. In graph clustering, the goal is to separate nodes based on the relationships implied by their connections, commonly represented as edges in the graph. This problem is fundamental to graph analysis and has applications across various domains, including social network analysis, bioinformatics, and recommendation systems. Methods such as those outlined by Schaeffer [151] and Nascimento et al. [152] emphasize the importance of leveraging graph structure to inform clustering decisions. Building on these principles, the approach described in [150] incorporates the use of a deep autoencoder to facilitate graph clustering. In this method, the autoencoder is employed to learn a reduced representation of the graph's features, effectively condensing the information contained within the graph's nodes and edges into a lower-dimensional latent space. Once this feature extraction step is completed, k-means clustering is applied to the latent space representation. The graph nodes are then assigned to clusters based on their positions in this lower-dimensional space. This strategy combines the power of deep learning for feature representation with the simplicity and effectiveness of k-means for cluster assignment, providing a structured framework for tackling graph clustering problems.

More recent AE-based algorithms have been developed to address clustering challenges by simultaneously mapping data points onto a lower-dimensional space and performing clustering within this space. This integrated approach aims to enhance the overall clustering performance by leveraging the strengths of dimensionality reduction and clustering in a unified framework. One prominent example of such algorithms is the Deep Embedding Clustering (DEC) method [17]. DEC begins by training a stacked AE in a layer-by-layer fashion. Each layer of the AE functions as a denoising autoencoder, which reconstructs the previous layer's output even after random corruption, thereby ensuring robustness. Once the AE training is complete, the decoder part of the network is discarded, and the encoder part is fine-tuned to optimize clustering performance. This fine-tuning is achieved by minimizing the Kullback–Leibler (KL) divergence between the distribution of soft cluster assignments produced by the model and a pre-defined target distribution. In DEC, the Student's t-distribution is employed to compute soft assignments, which measure the similarity between data points and cluster centers. However, because clustering is inherently unsupervised, the true target distribution of the data remains unknown. DEC circumvents this issue by defining an arbitrary target distribution, which is based on the squared values of the soft assignments. While this approach enables DEC to effectively perform clustering, it has certain limitations, particularly in preserving the data's local structure. To address these limitations, several recent algorithms—such as those presented in [15, 18, 19, 25]—leverage both the encoder and decoder parts of the AE to retain the data's local structure while clustering. These methods aim to improve clustering performance by integrating additional objectives into the optimization process. For instance, the Improved Deep Embedding Clustering (IDEC) algorithm [18] enhances DEC by introducing a reconstruction loss term in addition to the KL divergence loss. By minimizing both the reconstruction loss and the KL divergence, IDEC ensures that the learned representation retains essential data structure while achieving effective clustering. Building on IDEC, the Improved Deep Embedding Clustering with Fuzzy Supervision (IDECF) [15] further enhances the DEC framework. IDECF employs a fully connected network known as the Deep Fuzzy C-Means Network to estimate the target distribution, which introduces an element of fuzzy supervision into the clustering process. Similar to IDEC, IDECF also includes the reconstruction loss in its optimization problem, thereby improving the preservation of local structure while refining the clustering results. Adversarial Deep Embedded Clustering (ADEC) [153] tackle two key issues in deep clustering: feature randomness and feature drift. Feature randomness arises in deep clustering when unreliable pseudo-labels distort the latent space due to weak supervisory signals. Feature drift occurs in autoencoder models when clustering and reconstruction objectives conflict, compromising latent space quality. ADEC incorporates adversarial training to create a balance between clustering and reconstruction tasks, ensuring the latent space remains both meaningful and well-structured for effective clustering. Another notable method is the Deep Clustering Network (DCN) [25], which takes a different approach by optimizing a combination of the reconstruction loss and the k-means clustering objective. This joint optimization ensures that the latent space generated by the AE is k-means friendly, meaning that the cluster centers align well with the underlying data distribution in the latent space. To achieve this, DCN alternates between updating the network weights and optimizing the cluster centers. The cluster centers are refined by solving a discrete optimization problem, which ensures that they are optimally positioned relative to the data points in the latent space. Deep K-Means (DKM) algorithm [19] builds upon the objective function of DCN but approaches the optimization process differently. Instead of solving a discrete optimization problem to update cluster centers, DKM formulates a continuous optimization problem. This continuous approach allows for simultaneous updates of the AE parameters and the cluster centers, leading to a more streamlined and efficient optimization process. The Deep Embedded Probabilistic Clustering (DEPICT) [154] framework takes a probabilistic approach to clustering, combining deep feature learning with probabilistic cluster assignment. At its core, DEPICT uses a deep convolutional autoencoder to extract robust and non-linear features from input data, which are crucial for clustering in complex scenarios. The model incorporates a clustering loss based on Kullback-Leibler (KL) divergence to minimize discrepancies between predicted cluster distributions and their refined versions. Additionally, DEPICT incorporates a data-dependent regularization term to mitigate overfitting. [155] explores the challenges of traditional deep clustering models, which often combine clustering and reconstruction losses through autoencoders. A common issue with these methods is the lack of constraints on the latent space, leading to intermediate features that are not ideal for clustering, particularly when dealing with a high number of clusters. To address this, [155] introduces an innovative clustering network that utilizes a pre-clustering center codebook. This design enforces constraints on the latent space, resulting in intermediate features that are better optimized for clustering tasks, thus enhancing performance even as the number of clusters increases. This method presents a significant improvement in clustering complex datasets. [156] presents an innovative deep clustering approach that combines a Transformer-based Autoencoder (TAE) with a K-means clustering network (KNet) to tackle the complexities of high-dimensional data clustering. The TAE employs self-attention mechanisms to capture global features, offering significant improvements over traditional autoencoders. To enhance clustering performance, the model introduces a Convex Combination Loss (CCL) that encourages features from the same cluster to form a convex hull, alongside contrastive learning to improve feature distinctiveness. KNet is designed using a relaxed and differentiable K-means clustering model, enabling seamless integration with TAE for end-to-end optimization. [157] presents a framework that integrates clustering and feature selection, tasks traditionally performed independently. By integrating a concrete selector layer into an autoencoder, the model selects important features while clustering data in a latent space, ensuring more accurate and interpretable results. The concrete selector layer handles feature selection, while the modified autoencoder facilitates clustering in the latent space by integrating K-means loss and optimizing inter-cluster distances.

AE-based clustering methods have demonstrated exceptional versatility across various application domains. In text and document analysis [158], they enable semantic feature extraction, making them integral to tasks like topic modeling [159], where they uncover latent structures and facilitate effective organization and clustering of textual data. In the field of audio and speech processing [160], AE-based approaches transform complex acoustic signals into compact representations, aiding in pattern recognition and the grouping of similar samples for applications such as speech recognition, speaker identification, and music categorization. In social network analysis [161], these methods analyze user behavior and relational data to identify clusters of users with similar activities, preferences, or connections, which is valuable for community detection, user segmentation, and targeted recommendation systems. These applications underscore the utility of AE-based clustering methods in extracting meaningful patterns and organizing high-dimensional data in diverse fields.

2.2.4 Generative based Methods

Variational autoencoders (VAEs) [105] and Generative adversarial networks (GANs) [162] are two prominent deep generative models that are widely used for tasks such as understanding data distributions and performing clustering. These models are particularly effective because they can capture complex patterns in data and facilitate the discovery of latent structures. In the context of clustering, these generative models provide frameworks for learning underlying distributions and partitioning data into meaningful groups. For instance, Variational Deep Embedding (VaDE) [163] models the data generation process using a combination of a GMM and a DNN: (1) the GMM selects a cluster, (2) a latent embedding is generated from the chosen cluster, and (3) the DNN decodes the latent embedding into an observable output. The model uses a variational inference framework, employing a neural network to map observable data into latent embeddings. The optimization relies on maximizing the evidence lower bound (ELBO) through the Stochastic Gradient Variational Bayes (SGVB) estimator and the reparameterization trick. On the other hand, [41] employs an adversarial autoencoder to build a lower-dimensional code space tailored for clustering, applying a tunable Gaussian mixture prior to facilitate a joint optimization strategy. [164] integrates Variational Autoencoders (VAE) with Gamma Mixture Models (GamMM) to address the challenges of Gaussian-based methods, obtaining higher quality embeddings of the data latent space. By utilizing GamMM for latent space representation and incorporating a reparameterization technique to approximate Gamma distributions, the model improves clustering accuracy and generative capabilities. [165] enhances VAEs by employing prototype-based regularization in the latent space. The method clusters the latent space around defined prototype centroids and utilizes maximum mean discrepancy loss to ensure effective clustering without compromising reconstruction quality. Unlike traditional methods, it eliminates the need for auxiliary networks and allows for controlled data synthesis by introducing noise around prototype coordinates.

GAN-based clustering methods offer an alternative approach by framing clustering as a generative task. GANs are based on a min-max optimization framework involving two competing models: the generator, which trains to create realistic data samples, and the discriminator, which evaluates whether a given sample is real or generated. The adversarial training process drives both models to improve iteratively, leading to the generator producing data samples that closely resemble the true data distribution. Several clustering algorithms based on GANs have been presented in [166]. For example, InfoGAN [167] is an unsupervised GAN variant that learns disentangled representations by maximizing mutual information between certain latent variables and the generated data. This property enables InfoGAN to identify meaningful clusters in an unsupervised manner. Another approach, GAN Mixture Model (GANMM) [168], extends GMM concepts by training a separate GAN for each cluster, effectively modeling the data as a mixture of GANs. GANs are also applied in semi-supervised clustering [169], leveraging a small portion of labeled data combined with a larger pool of unlabeled data to improve clustering performance. By integrating prior knowledge into the GAN architecture, the process allows for refining clustering outcomes to better align with existing patterns or predefined labels. However, GAN-based clustering methods encounter challenges such as vanishing gradients and mode collapse, which hinder stable training and the ability to capture all modes of the data distribution. Despite these issues, GAN-based models remain a powerful tool for clustering due to their ability to model highly complex data distributions and generate synthetic data that can further aid in understanding cluster structures [166].

Generative-based models have been widely applied to clustering tasks due to their capability to model complex data distributions and produce meaningful latent representations. For example, these models can be used in segmentation of medical images [170, 171], where clusters correspond to different tissue types or pathological regions, and analysis of single-cell genomics data.

In social network analysis [172, 173], VAEs and GANs enable the identification of communities or subgroups by clustering latent representations of user interactions. Moreover, VAEs are also utilized in anomaly detection [174], where outliers in the latent space are identified as anomalies, and in personalized recommendation systems [175], where clustering helps group users or items based on shared preferences. The flexibility and robustness of generative-based models make them a versatile tool for clustering across diverse fields.

2.2.5 Self-supervised Methods

Self-supervised clustering has emerged as a pivotal research area within deep learning, addressing the critical challenge of extracting meaningful and structured information from unlabeled data. This paradigm combines the strengths of representation learning and clustering objectives, creating models that not only uncover latent patterns in the data but also optimize the learned representations to enhance clustering performance. Self-supervised clustering aims to bridge the gap between feature learning and clustering accuracy, ensuring that learned embeddings are inherently clustering-friendly. Self-supervised methods often leverage auxiliary tasks or data augmentations, such as generating complementary predictions or modifying input data, to further enhance learning and refine representation quality. Auxiliary tasks, often self-generated, provide pseudo-supervision that guides the model in learning meaningful representations. Data augmentation plays a complementary role by enriching the training process through transformations that preserve the semantic meaning of data points while introducing variability. Techniques like random cropping, flipping, scaling, and color jittering for image data or masking and token replacement for text data ensure the model maintains robustness against variations in input data. Recent algorithms, such as [23, 176–178], simultaneously learn to extract informative features from data and perform clustering in a single framework. For example, [176] presents the Invariant Information Clustering (IIC) objective, which maximizes the mutual information between cluster assignments of paired data samples. This approach encourages the model to find clusters that are both predictable and balanced, effectively avoiding trivial solutions where all data points are assigned to a single cluster. To generate paired data samples, the method applies random transformations to each image, creating pairs that share underlying semantic content. By maximizing the mutual information between the cluster assignments of these pairs, the model learns representations that are invariant to such transformations, capturing the essential features of the data. Deep Comprehensive Correlation Mining (DCCM) [177] integrates three key components in its loss function to enhance clustering performance: pseudo-label supervision, local robustness exploration, and triplet mutual information. 1- pseudo-label supervision, involves assigning provisional labels to data points based on clustering results. These pseudo-labels enable the model to learn category-level distinctions among data points. 2- local robustness exploration, focuses on ensuring that the learned features are stable and consistent under various transformations of the input data. 3- triplet mutual information, extends the traditional notion of mutual information by modeling relationships among triplets of data points. This approach helps the model capture richer dependencies within the data by considering interactions beyond pairwise similarities. Deep Semantic Clustering by Partition Confidence Maximisation (PICA) [178] focuses on improving the overall partition confidence of the clustering process by ensuring that data separation aligns more closely with semantic categories. The approach uses a differentiable partition uncertainty index, combined with a stochastic approximation, to evaluate the quality of clustering. This enables the method to leverage deep networks and support mini-batch training effectively. By optimizing global partition confidence, PICA avoids the issues of low intra-cluster compactness and diminished inter-cluster diversity seen in prior methods. Multi-Modal Deep Clustering (MMDC) [179] utilizes a GMM as the clustering target. The GMM represents clusters as probabilistic distributions in the feature space, and the network learns to map image embeddings to these target distributions. The GMM's parameters are dynamically updated during training, ensuring that the clustering remains consistent with the learned representations. To further enhance the quality of the feature representations, MMDC incorporates a self-supervised auxiliary task of predicting image rotations. This task challenges the network to recognize and predict the degree of rotation applied to input images (e.g., 0°, 90°, 180°, or 270°). By solving this task, the network is forced to learn semantic information about the images, which indirectly improves the clustering results.

As detailed in Section 2.1.2, contrastive learning is a widely used approach within self-supervised learning. Its primary goal is to enhance data representations by bringing similar samples, known as positive pairs, closer together in the feature space while pushing dissimilar or unrelated samples, referred to as negative examples, farther apart. This method is particularly effective in tasks requiring the discovery of underlying data structures, making it highly popular not only for representation learning but also in clustering tasks [23, 24, 180]. By organizing the feature space in a way that naturally groups similar samples, contrastive learning facilitates the formation of well-defined clusters, enabling models to better capture meaningful patterns within unlabeled datasets and improve performance in downstream applications. For example, [180] develop a two-stage framework that separates the tasks of feature learning and clustering, offering an alternative to end-to-end clustering methods commonly used. In the first stage, the framework employs self-supervised learning to extract semantically rich features from images. This is achieved by training a neural network on a pretext task, which requires no labeled data. In the second stage, it uses a contrastive learning loss, which helps refine the clustering process. This loss encourages image pairs that belong to the same cluster to remain close in the feature space while pushing apart those in different clusters. Contrastive Clustering [23] introduces a one-stage online clustering method that integrates both instance-level and cluster-level contrastive learning. In this approach, positive and negative instance pairs are generated through data augmentations and then mapped into a feature space. Within this space, contrastive learning is applied at two levels: instance-level contrastive learning is conducted in the row space, treating rows of the feature matrix as representations of different samples in the batch; cluster-level contrastive learning is performed in the column space, interpreting columns as cluster representations distributed over the dataset. By maximizing the similarity of positive pairs and reducing that of negative pairs at both levels, the model simultaneously learns representations and cluster assignments in an end-to-end fashion. Distinct from conventional methods that depend on direct pairwise comparisons, SwAV [181] integrates clustering with a consistency objective that aligns cluster assignments across different augmentations of the same image. A core innovation is its ”swapped” prediction strategy, which predicts the cluster assignment of one augmented view from another. [182] utilizes both weak and strong augmentations to enhance model performance. Weak augmentations are defined as minimal transformations that preserve the fundamental structure of the data, such as random cropping, horizontal flipping, and slight adjustments to color. On the other hand, strong augmentations involve more drastic changes, including cutout, strong color jittering, and geometric transformations, which significantly modify the input data to increase variation.This framework incorporates a shared-weight model that processes three views of the data: one subjected to strong augmentation and two to weak augmentations. These views are used for both instance-level and cluster-level contrastive learning. The weak-weak and strong-weak view pairs enable the model to learn both fine-grained features and robust representations that are invariant to substantial distortions. [183] introduces a single-stage framework that merges representation learning with clustering, eliminating the need for pre-training. It refines traditional loss functions by excluding the impact of negative instances in cluster center learning, ensuring more stable training in the absence of ground-truth labels. Additionally, it incorporates a global entropy constraint to ensure balanced cluster sizes and reducing the need for hyperparameter tuning.

Self-supervised clustering is a powerful tool for uncovering patterns in data without the need for labeled samples. It plays a vital role in organizing images by grouping them based on shared characteristics, which is useful in fields such as facial analysis [184] and medical imaging [185] for sorting and understanding large datasets. In text processing [186], it helps cluster documents, articles, or feedback into distinct categories or themes, supporting tasks like customer sentiment analysis or topic modeling in research. In the biological sciences, self-supervised clustering aids in grouping genetic sequences [187] or molecular structures [188], offering valuable insights into evolutionary biology and disease mechanisms. Additionally, it enhances recommendation systems [189] by identifying clusters of users or products based on behavioral or attribute similarities, enabling personalized content delivery in sectors like e-commerce and entertainment. By relying on the intrinsic properties of data, self-supervised clustering provides an efficient means to derive structure and actionable insights in various domains.

2.3 Continual Learning

Continual learning (CL), also referred to as lifelong learning, is a key area of artificial intelligence research focused on empowering systems to incrementally acquire, adapt, and expand their knowledge over time. This capability is crucial for intelligent systems functioning in dynamic, real-world environments, where they must constantly adapt to new tasks and evolving data streams. Unlike traditional machine learning models, which are generally trained on static datasets, continual learning systems aim to preserve and enhance previously learned knowledge while incorporating new information. In this section, we first explore how various algorithms address the primary challenge of continual learning, known as catastrophic forgetting. Then, we delve into different scenarios and applications of CL, highlighting its practical significance.

2.3.1 Catastrophic Forgetting (CF)

One of the principal challenges in continual learning is managing catastrophic forgetting (CF), a phenomenon where acquiring new knowledge can disrupt or overwrite previously learned information. This occurs because the neural network's weights, optimized for new tasks, may conflict with those representing earlier tasks. Overcoming this challenge is crucial for maintaining the versatility and utility of continual learning systems over time. A central focus in addressing catastrophic forgetting is achieving the delicate balance between stability and plasticity. Stability ensures the retention of previously acquired knowledge, while plasticity enables the system to integrate new information. Striking this balance is inherently complex: placing too much emphasis on stability risks stifling adaptability, whereas excessive plasticity can lead to the erosion of prior knowledge.

Researchers have developed several strategies to mitigate CF and enhance the performance of continual learning systems. **Regularization-based methods** introduce constraints on model updates, penalizing changes to parameters deemed important for previously learned tasks. This approach encourages the system to preserve critical knowledge while integrating new information. Another popular strategy is **replay-based methods**, which involve retaining a subset of previously encountered data or generating synthetic examples to reinforce earlier learning during training. **Representation-Based methods** mitigate CF by focusing on learning and preserving shared, task-agnostic representations that encapsulate essential features of data across tasks. Lastly, **architecture-based methods** allocate or isolate separate subsets of model parameters for different tasks, effectively minimizing interference between them. In the remainder of this sub-section, we will delve into the methods used to implement and manage these strategies.

Regularization-based methods: These methods apply constraints on model updates, penalizing changes to parameters identified as essential for previously learned tasks. This strategy promotes the retention of crucial knowledge while allowing the integration of new information. some methods in this category [49, 190] perform weight regularization, which specifically targets the variability of network parameters. A common approach involves adding penalty to the loss function, which penalizes the changes in each parameter based on its ”importance” in carrying out the previous tasks. For instance, Elastic Weight Consolidation (EWC) [49, 50] incorporates a quadratic regularization term into the loss function, leveraging the Fisher Information Matrix (FIM) to penalize substantial deviations of critical weight values from their previously learned states. This approach effectively anchors important parameters near their original values, allowing the network to retain prior knowledge while adapting to new tasks. However, EWC assumes the FIM is diagonal, implying independence among parameters—a simplification that often does not hold in practice, thereby limiting its effectiveness. To address this limitation, the reparameterization method proposed in [191] rotates the parameter space to more accurately align with the true structure of the FIM, improving the accuracy of its diagonal approximation and enhancing performance. Synaptic Intelligence (SI) method [51] is another weight regularization method that assigns an importance score to each synaptic weight, indicating its role in the performance of earlier tasks. These scores are calculated based on the contribution of each parameter to the overall loss function during training. When a network learns a new task, a regularization component is added to the training process, discouraging changes to parameters identified as critical for prior tasks. [192] preserves prior knowledge by employing function-space Bayesian inference. It utilizes Gaussian processes to model task-specific functions instead of focusing solely on neural network parameters. The approach summarizes each task using inducing points and posterior distributions, which act as compact knowledge representations. By incorporating KL-divergence for regularization, the model effectively balances acquiring new information while maintaining past learning. Memory Aware Synapses (MAS) [193] draws inspiration from neuroplasticity to prioritize the retention of essential knowledge. MAS works by assigning an importance value to each parameter, determined by how sensitive the model's predicted output is to variations in that parameter. During the learning of a new task, the approach discourages modifications to parameters identified as crucial for earlier tasks, thereby safeguarding vital information from being overwritten. Remarkably, MAS can update these importance values using unlabeled data, allowing the network to adapt to particular test scenarios and discern which information should be preserved or can be disregarded. [194] builds upon a key insight into the behavior of loss functions in neural networks: changes in model parameters affect various regions of the loss landscape asymmetrically. For example, a parameter update that improves performance on one task can inadvertently cause significant degradation on another. [194] models the loss function as an asymmetric quadratic form, deliberately overestimating potential loss in regions that are less directly influenced during training. This strategic overestimation acts as a protective mechanism, preserving knowledge from prior tasks while enabling effective learning of new ones.

Replay based methods: Replay-based strategies in continual learning address the issue of catastrophic forgetting by enabling models to maintain and synthesize knowledge from prior tasks while adapting to new ones. These methods typically utilize a memory buffer to store a selected portion of past task data, which is replayed during subsequent training phases. The stored data can consist of real examples from prior tasks or synthetic data generated by methods like generative models, ensuring a balance between memory usage and data representation. A critical element of these methods is rehearsal, where the model learns simultaneously from new data and replayed samples, maintaining task performance across the learning sequence. This approach helps ensure stability and adaptability in continually learning systems. For example, [52] utilizes experience replay (ER), which combines training on new task data with examples stored from previous tasks in a small memory buffer. [53] presents Meta-Experience Replay (MER), an algorithm that incorporates experience replay with optimization-based meta-learning. MER is designed to adjust model parameters in a way that reduces the likelihood of future gradient-based interference while enhancing the potential for positive transfer. This approach enables the model to retain prior knowledge more effectively and apply it beneficially to new learning scenarios. To enhance storage efficiency, [54] explores methods for compressing data from non-i.i.d. streams in an online learning framework. This approach uses an AE and adaptive quantization technique to efficiently store samples in a memory. [195] balance the quality and quantity of compressed data to optimize memory replay effectiveness. The authors introduce an approach utilizing determinantal point processes to determine the appropriate compression quality for incoming training samples efficiently. This method allows the use of standard data compression algorithms, like JPEG, with carefully selected quality settings to significantly improve performance over existing baselines without substantial computational overhead. [196] stores visual explanations, such as saliency maps, in a memory buffer for each task the model learns. Saliency maps highlight which parts of the input data (e.g., image regions) are most influential for the model's decisions. When the model learns new tasks, [196] ensures that its explanations for the predictions remain consistent with those generated during earlier tasks. By preserving these explanations, the model retains the underlying rationale behind its past decisions, reducing the risk of forgetting critical information. [60] integrates rehearsal methods with knowledge distillation by saving the network's output logits during training and ensuring they align with the model's current predictions, preserving consistency with previously acquired knowledge. The main challenge of retaining data from previous tasks revolves around privacy issues. In cases involving sensitive information such as personal details, medical records, or confidential datasets, storing raw data for later use may breach privacy laws or violate organizational guidelines. To address this issue, researchers introduce generative replay. For instance, [57] integrates a deep generative model (the "generator") with a task-solving model (the "solver"). The generator creates synthetic data that mimics previously learned tasks, allowing this data to be combined with new task data for training the solver. This eliminates the need to store actual past data while retaining knowledge from earlier tasks. [55] decompose the network into a feature extractor and a classifier. To preserve knowledge, they integrate generative feature replay within the classifier and apply feature distillation to the feature extractor. This approach simplifies the generative replay process and effectively addresses the imbalance problem. It is both computationally efficient and scalable to large datasets. Traditional generative replay methods, which involve a generator producing samples of past tasks to train a classifier, often suffer from two main issues (See [56]): a unidirectional flow of information from generator to classifier, and the degradation of sample quality due to the reuse of generated data. To overcome these limitations, [56] utilizes a diffusion model as the generator and introduces an instruction-operator derived from the classifier to guide the sample generation process. By establishing a bidirectional interaction between the generator and classifier, [56] aims to produce higher-quality samples that better represent previous tasks. [197] introduces latent matching to enhance the generated features alignment between the reconstructed and original data. Moreover, based on the observation that reconstructions are more effective at preserving knowledge, this approach involves cycling generated data through a pre-trained model to align them more closely with the original data.

Representation-Learning based methods: The representation-learning based methods in continual learning is designed to enhance the resilience and versatility of representations to effectively address issues like catastrophic forgetting and inter-task interference. This approach emphasizes the creation of generalizable and robust features that can adapt to new tasks while preserving knowledge from previous ones. By minimizing sensitivity to changes specific to individual tasks, the representation-based method supports smoother task transitions and sustained performance. One key category of these methods involves self-supervised learning, which frequently employs contrastive loss to develop robust representations. This technique helps mitigate forgetting and enhances adaptability across tasks. For instance, [198] focuses on preserving the consistency of learned representations by introducing an instance-wise relation distillation loss. It measures the similarity relationships between data points in the current and previously trained models and minimizes any deviation. This self-supervised distillation helps maintain the model's understanding of past tasks. Lifelong Unsupervised Mixup (Lump) [65] generates synthetic data points by interpolating between current and past tasks. This interpolated training data helps the model maintain a balanced representation of old and new knowledge, reducing the bias toward more recent data that typically causes forgetting. [199] employs a dual mechanism consisting of a fast learner for supervised learning of task-specific representations and a slow learner for unsupervised, task-agnostic representation learning through self-supervised methods to mitigate CF. This approach is designed to improve continual learning by reducing catastrophic forgetting and facilitating effective knowledge transfer. [59] presents a dual-network framework composed of a student network and a teacher network. The student network is designed to acquire knowledge from new tasks, while the teacher network is responsible for maintaining and integrating previously learned information. A key innovation of this approach is the application of contrastive distillation, which aligns the feature representations of the two networks to ensure the retention of prior knowledge and consistency. CaSSLe [58] introduces a predictor network that bridges current and past representations. This predictor aligns the embeddings learned at each training stage with those from previous stages, effectively acting as a distillation mechanism. By converting standard self-supervised learning objectives into a form that retains consistency across tasks, CaSSLe preserves essential features of earlier data while learning new information. One notable drawback of [58, 59] frameworks is their memory inefficiency, particularly when applied to deep neural networks with millions of parameters. This inefficiency arises from the need to store a copy of the model from previous tasks in memory to align current and past representations effectively. [69] introduces a novel approach by combining a self-supervised forgetting loss with an online memory update strategy. The self-supervised forgetting loss operates by applying the KL divergence to measure the pairwise similarity between feature representations of previous and current tasks, ensuring the retention of learned knowledge. Simultaneously, the online memory update mechanism maintains a fixed-size buffer, carefully selecting a representative subset of past data to effectively preserve the overall data distribution. [200] addresses CF by introducing a dual-network approach designed to balance adaptability and memory retention. It uses an expert network dedicated solely to learning new tasks, allowing it to focus entirely on acquiring fresh knowledge without constraints related to retaining previous information. This improves the network's plasticity, or its ability to adapt to new data. To prevent forgetting, the newly learned representations are integrated into a main network through an adaptation-retrospection phase. This phase aligns the representations from both networks, ensuring that the main network retains past knowledge while incorporating new information.

Architecture-Based methods: These methods allocate or isolate subsets of model parameters for each task, ensuring effective task-specific knowledge retention while minimizing interference across tasks. By dedicating distinct parameters to individual tasks, these approaches create specialized subspaces within the model, preventing the overwriting of previously learned information. Parameter allocation methods statically or dynamically assign specific parameters to individual tasks, ensuring minimal interference. For instance, [201] enables a fixed deep neural network to perform multiple tasks without degrading its performance on previously learned tasks. This approach utilizes binary masks to adaptively select task-specific parameters from a shared network. These masks are trained in an end-to-end differentiable manner, adding minimal overhead—approximately 1 bit per network parameter for each task. Similarly, [202] introduce a task-based hard attention mechanism that assigns specific network parameters to each task, effectively isolating them to preserve prior learning. [203] leverages network pruning techniques to eliminate redundant parameters in large networks, thereby freeing up capacity to learn new tasks. Multiple tasks can be ”packed” into a single model with little loss of performance and little storage overhead by iteratively pruning and retraining the network. [204] evaluates and manages the relationships between tasks, identifying whether prior knowledge is helpful or harmful to the learning of new tasks. Their framework includes a global feature extractor and task-specific classifiers, enabling the system to retain beneficial knowledge while mitigating interference from detrimental information. A sensitivity measure is introduced to quantify the impact of prior tasks on the current task’s loss, guiding the selective transfer of supportive knowledge. [205] introduces neuron-specific masking to selectively activate particular neurons and their corresponding weights for each task. This approach facilitates forward knowledge transfer by reusing less critical weights while preserving the more significant ones from prior tasks. Fixed allocation approaches like [201, 202] may face difficulties with constrained model capacity as the number of tasks grows. To address these problems, some alternative methods adopt dynamic strategies, expanding the model architecture as necessary to effectively handle new tasks. [206] address CF by allocating a distinct neural network, referred to as a ”column,” for each new task. These columns are interconnected through lateral connections, enabling the transfer of knowledge from earlier tasks to subsequent ones. [207] presents a dynamic expansion strategy that incorporates small task-specific decoders to handle new tasks or classes while preserving knowledge from previously learned tasks. The architecture consists of a shared multi-scale encoder and multiple compact task-specific decoders. The shared encoder is designed to transform the input sample into an intermediate representation, while each task-specific decoder is responsible for performing classification. Dense Network Expansion (DNE) [208] presents a strategy that incorporates dense connections among the intermediate layers of task-specific networks. This approach promotes the transfer of knowledge from prior tasks to new ones by enabling feature sharing and reuse. A central element of DNE is the Task Attention Block, a cross-task attention mechanism that functions at the feature-mixing level, independently of spatial attention. DNE maintains the feature space of old classes while extending the network and feature scale at a slower rate than traditional methods. Dynamic methods address the issue of scalability by increasing the size of the network as new tasks are introduced. This strategy enables the model to flexibly adjust its capacity to meet the demands of additional tasks. However, this flexibility has its trade-offs. Expanding the network can lead to challenges such as overfitting, particularly when the data for the new tasks is insufficient, as the additional parameters might overly specialize to the training data. Moreover, this approach tends to raise computational demands, requiring more resources during both the training phase and inference due to the growing network complexity.

2.3.2 Continual Learning Scenarios

Continual learning comprises diverse strategies designed to handle the challenges of adapting to shifting data and tasks over time, all while maintaining previously gained knowledge. Among these, supervised continual learning (SCL), semi-supervised continual learning (SeCL), and unsupervised continual learning (UCL) highlight the diversity of real-world scenarios that require adaptive solutions.

Supervised Continual Learning: Supervised continual learning (SCL), which relies on labeled data, can be further categorized into task-incremental learning (TIL) [205,206] and class-incremental learning (CIL) [70], each suited for different real-world requirements. TIL involves scenarios where task identifiers are accessible during both training and evaluation. This explicit task information allows the model to tailor its learning strategy for each task, minimizing interference with previously learned tasks. Conversely, CIL introduces a more demanding situation where task labels are provided only during training but are absent during evaluation. Here, the model must differentiate between classes from different tasks without direct cues, necessitating sophisticated strategies to preserve performance on earlier tasks while assimilating new ones. As an example, consider two binary classification tasks: distinguishing between ”zebra” and ”elephant” and another task between ”cat” and ”dog”. In TIL, the model uses task identities to separate predictions for the first task from the second, simplifying classification. In contrast, CIL demands that the model classify all four classes simultaneously without task-specific hints.

TIL has broad applications in various real-world domains, including robotics [62], 3D object detection [209], machinery fault diagnosis [210], and human activity recognition [211]. For instance, [62] address the need for assistive robotics to adapt to dynamic environments. It involves learning tasks sequentially, with each task presenting progressively varied challenges, such as changes in illumination, occlusion, camera-object distance/angle, and clutter. [209] explores 3D object detection within a TIL framework by utilizing a Bayesian-enhanced KD method. It is based on Region Proposal Networks (RPNs) tailored for 3D object detection, which analyze point clouds to extract features and produce 3D bounding boxes for identifying objects. [210] is designed in a typical TIL context, where sequential fault diagnosis tasks for different machinery components (bearings, gears, discs) are introduced incrementally. In human activity recognition, [211] introduces a framework that creates task-specific feature extraction branches for each new task and a feature redistribution layer. This approach emphasizes extending the feature extraction branches to handle new inputs from diverse sensor modalities while ensuring the classifier outputs remain consistent in both dimensions and quantity.

CIL problems are characterized by two core challenges: within-task prediction and task-identity prediction [212,213]. While within-task prediction involves distinguishing between classes within a single task, task-identity prediction addresses the broader challenge of determining which task a sample belongs to in the absence of explicit task labels during inference. This lack of task identity significantly complicates CIL compared to task-incremental learning setups, as models must effectively classify all learned classes simultaneously without the contextual aid of task-specific information. CIL has broad range of real-world applications such as Few-shot learning [214], audio-visual video recognition [215], image classification [216], and object detection [217]. For example, [215] emphasizes the challenges arising from the multi-modal characteristics of audio-visual data. The proposed method maintains semantic similarities both within and across audio and visual modalities throughout tasks, while also preserving audio-guided visual attention mechanisms established in earlier tasks. This approach effectively prevents the model from losing critical cross-modal relationships. [216] enhances memory efficiency in CIL setting by preserving a greater number of low-fidelity auxiliary exemplar samples instead of fewer high-fidelity ones for image classification task.

Semi-Supervised Continual Learning: Semi-supervised continual learning provides an innovative solution to the challenge of limited labeled data by utilizing a combination of labeled and unlabeled datasets. This approach reduces the dependency on extensive labeling efforts, which are often costly and labor-intensive [218, 219]. Semi-supervised methods address CF through techniques like knowledge distillation [220], which helps preserve prior knowledge, and self-supervised learning [221], which identifies intrinsic patterns in data without needing labels. These strategies, along with representation learning, enhance the model's ability to generalize effectively across different tasks and datasets. For example, [222] tackles challenges in semi-supervised continual learning using a nearest-neighbor classifier to partition the feature space non-linearly, promoting robust and adaptable representation learning. Their method ensures the model captures meaningful features for the current task while effectively distilling knowledge from prior tasks through a novel semi-supervised distillation loss. Unlike traditional approaches that focus solely on class- or sample-level information, this method integrates sample-class level relationships within the nearest-neighbor classifier. [223] leverages a combination of metric learning and consistency regularization to optimize learning from limited labeled data. By enforcing consistency among augmented and interpolated examples and imposing contrastive constraints between representations of different tasks, [223] effectively exploits unlabeled data while retaining task-specific information.

Replay mechanisms are also integral to semi-supervised continual learning. Techniques such as generative replay [63, 224], which synthesizes past data examples, and feature preservation, which secures essential learned features, enable the model to retain knowledge from earlier tasks while accommodating new information. These methods ensure that the model remains stable and capable even when faced with dynamically changing data and tasks. For example, [63] presents a method where a classifier is jointly trained with a conditional GAN, which continuously provides the classifier with the learned data distribution. This method enables the classifier to replay data sampled from the conditional generator in an online fashion, leveraging unlabeled data efficiently in terms of both time and storage. Additionally, to address the challenge of CF for unlabeled data, the approach selectively stabilizes the discriminator's parameters that are crucial for distinguishing between pairs of previously seen unlabeled data and their pseudo-labels generated by the classifier. [224] Combines a continually trained classifier with a diffusion-based generative model in a unified, jointly optimized neural network. This method illustrates that shared parameterization, alongside knowledge distillation techniques, ensures stable adaptation to new tasks while effectively reducing catastrophic forgetting (CF).

Unsupervised Continual Learning Unsupervised continual learning is a cutting-edge approach to machine learning that addresses scenarios where labeled data is entirely unavailable, emphasizing the need to extract meaningful patterns and uncover hidden structures from a continuous stream of data. Unlike SCL, which depends on labeled datasets for training, unsupervised continual learning focuses on creating adaptable systems that can learn incrementally without explicit guidance or predefined labels. This paradigm has gained significant attention due to its potential to operate in dynamic and ever-changing environments, where data evolves over time and labels are impractical or impossible to obtain.

Unsupervised continual learning is especially relevant in fields where data evolves dynamically over time rather than remaining static, such as in anomaly detection [66]. The continual prompting module leverages a compact memory bank of key-prompt-knowledge pairs to facilitate task-invariant anomaly predictions by incorporating task-specific ”normal” knowledge. Additionally, structure-based contrastive learning, developed in conjunction with the segment anything model, enhances prompt learning and significantly improves the accuracy of anomaly segmentation. [67] presents a method that enables self-driving car perception systems to adapt continuously to new, unseen environments without requiring access to previous training data. This approach addresses challenges such as varying weather, lighting, and geographical conditions that autonomous vehicles encounter. [225] presents a specialized framework for image classification tasks, focusing on enabling continual learning in an unsupervised environment. This approach replaces ground truth labels with pseudo labels, generated through a global clustering algorithm. The framework utilizes the model from the previous incremental step as a feature extractor to derive these pseudo labels for new data, facilitating a seamless update and adaptation process. [68] tackles a significant challenge in implementing EEG-based models in real-world scenarios, especially in clinical and healthcare applications. The proposed approach facilitates continuous and autonomous adaptation of EEG models to new subjects without the need for labeled data. This method achieves a delicate balance between plasticity (ensuring adaptation to new individuals) and stability (maintaining and generalizing prior knowledge). [226] introduces a framework that bridges unsupervised domain adaptation and CL to tackle gradually evolving target domains. Rather than relying on fully available target data, this method processes sequential, unlabeled batches from dynamic domains, utilizing a memory buffer and contrastive loss to ensure effective domain alignment and retention of learned knowledge.

While unsupervised continual learning frameworks show immense promise in real-world applications, to the best of our knowledge, there is a notable gap in algorithms specifically designed for clustering tasks within this paradigm. Addressing this limitation could open new avenues for leveraging unsupervised continual learning in broader, more complex scenarios.

Chapter 3

Deep multi-representation learning (DML)

In this chapter, we introduce a pioneering approach called Deep Multi-representation Learning (DML), a framework designed to enhance unsupervised data clustering by overcoming the constraints of conventional methods. Traditional clustering approaches often depend on a single latent space to represent all clusters, which can limit their effectiveness in distinguishing complex or overlapping data groups. DML addresses this issue by utilizing multiple AEs to generate tailored latent spaces that align with the specific characteristics of individual clusters. It assigns unique latent spaces to "challenging" clusters, where cluster centers are densely packed and overlapping, while reserving a shared latent space for "easier" clusters that are well-separated and clearly defined. To facilitate the training of these multiple AEs, DML introduces a novel loss function that integrates weighted reconstruction and clustering losses. This loss function adapts dynamically, placing greater emphasis on data points that strongly align with specific clusters, ensuring the model develops meaningful and cluster-specific representations. Unlike conventional methods that rely on rigid cluster assignments during training, DML employs soft assignments, focusing on high-confidence samples to effectively train distinct AEs for different clusters. This multirepresentation approach significantly enhances clustering performance, as demonstrated by experiments on various benchmark datasets, showing improved cluster separation and compactness compared to leading methods.

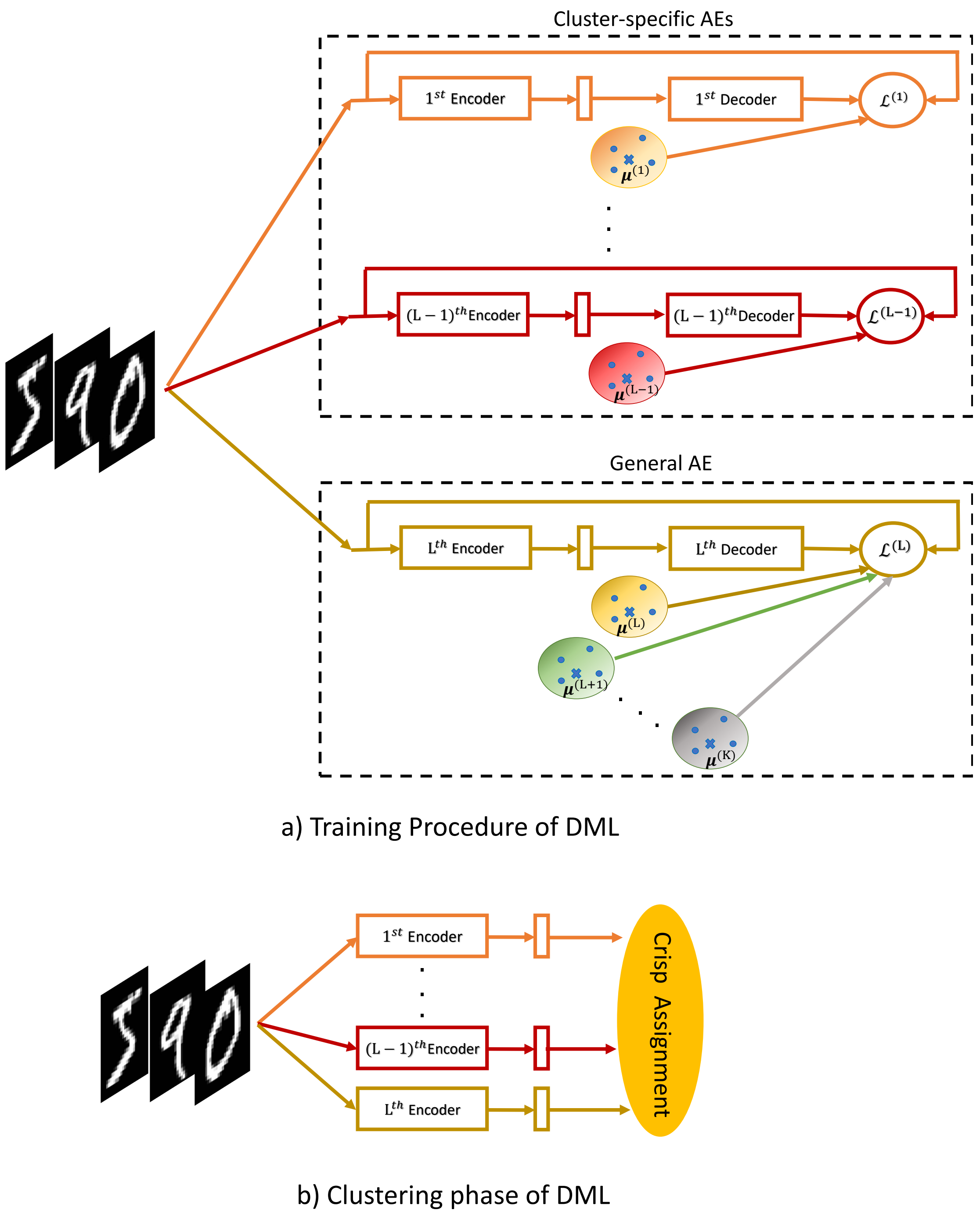


Figure 3.1: Block diagram of the proposed DML framework. (a) Training scheme of DML. (b) Final crisp cluster assignment phase.

3.1 Notation

K-clustering problem aims to divide the given dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ with N samples into K disjoint groups (aka clusters), where the ith data sample is denoted by x_i and K is the predefined number of groups. In this chapter, we aim to address the K-clustering problem by proposing DML framework. Assume there are L-1 difficult clusters (hence there are K-L+1 easy clusters), where L $\leq$ K. DML trains an individual AE for everyone of the difficult clusters and a single common AE for the easy clusters. We refer to the l^{th} AE of DML as AE$^{(l)}, l = 1, \ldots, \mathrm{L}$; where AE$^{(\mathrm{L})}$ denotes the general AE assigned to the easy clusters and the remaining AEs, i.e. AE$^{(l)}$ $l = 1, \ldots, \mathrm{L} - 1$, are assigned to the L$-1$ difficult clusters.

The cluster centers, i.e. $\boldsymbol{\mu}^{(k)}$ $k = 1, \ldots, \mathrm{K}$, are initialized to the centers obtained by $\mathcal{M}$. Consequently, all the DML's autoencoders are initialized with the $\mathcal{M}$'s autoencoder.

The AE of $\mathcal{M}$ is denoted by $\tilde{\mathrm{AE}}$. The encoder and decoder of $\tilde{\mathrm{AE}}$ are respectively denoted by $\tilde{f}(.)$ and $\tilde{g}(.)$. Representation of X in the latent space of $\tilde{\mathrm{AE}}$ is denoted by $\tilde{U} = \{\tilde{\mathbf{u}}_1, ..., \tilde{\mathbf{u}}_N\}$, where $\tilde{\mathbf{u}}_i = \tilde{f}(\mathbf{x}_i; \tilde{\boldsymbol{\theta}}_e) \in \mathbb{R}^d$, d indicates dimension of the latent space, and $\tilde{\boldsymbol{\theta}}_e$ represents parameters of the encoder network. The reconstructed output of $\tilde{\mathrm{AE}}$ is shown by $\tilde{\mathbf{x}}_i = g(\tilde{\mathbf{u}}_i; \tilde{\boldsymbol{\theta}}_d)$, where $\tilde{\boldsymbol{\theta}}_{\boldsymbol{d}}$ denotes the decoder parameters of $\tilde{\mathrm{AE}}$.

The encoder and decoder parameters of the l^{th} autoencoder of DML, i.e. AE$^{(l)}$, are respectively shown by $f^{(l)}(.)$ and $g^{(l)}(.)$. $U^{(l)} = \{\mathbf{u}_1^{(l)}, ..., \mathbf{u}_N^{(l)}\}$ denotes the representation of X in the latent space of AE$^{(l)}$, where $u_i^{(l)} = f^{(l)}(\mathbf{x}_i; \boldsymbol{\theta}_e^{(l)}) \in \mathbb{R}^d$ and $\boldsymbol{\theta}_e^{(l)}$ shows AE$^{(l)}$'s encoder parameters. Also, the reconstructed output of AE$^{(l)}$ is denoted by $\hat{\mathbf{x}}_i^{(l)} = g^{(l)}(\mathbf{u}_i^{(l)}; \boldsymbol{\theta}_d^{(l)})$, where $\boldsymbol{\theta}_d^{(l)}$ denotes the decoder parameters of AE$^{(l)}$.

In the following, for simplicity, we use notation $(.)^{(l_k)}$ to refer to the DML's latent space associated to the k^{th} data cluster. In general, if $\mathcal{D} = \{d_1, ..., d_{\mathrm{L}-1}\}$ is the set of difficult clusters and and $\mathcal{E} = \{e_1, ..., e_{\mathrm{K}-\mathrm{L}+1}\}$ is the set of easy clusters, then l_k can be obtained as below:

$$l_k = \sum_{i=1}^{\mathrm{L}-1} i\mathbb{1}\{k = d_i\} + \mathrm{L}\mathbb{1}\{k \in \mathcal{E}\}. \tag{3.1}$$

As an instance, assume there are four data clusters, i.e. $k \in \{1, 2, 3, 4\}$ and $\mathrm{K} = 4$, of which the second and third ones (i.e. $k \in \{2, 3\}$) are difficult and the first and fourth clusters (i.e. $k \in \{1, 4\}$) are easy. Therefore, in this example, $\mathrm{L} = 3$, $\mathcal{D} = \{2, 3\}$, $\mathcal{E} = \{1, 4\}$, and l_k for k = 1, 2, 3 and 4 are respectively equal to 3, 1, 2 and 3 – e.g. the DML's AE associated to $k = 2$ is AE$^{(1)}$, and the representation of x_i in the latent space corresponding to the data cluster $k = 2$ is $\mathbf{u}_i^{(1)}$ because $l_2 = 1$. Note that $l_k \in \{1, 2, .., \mathrm{L}\}$.

3.2 Soft assignment

To focus AE$^{(l_k)}$ on creating a latent space specialized in the k^{th} cluster, we devise a novel loss function that navigates the training process of AE$^{(l_k)}$'s encoder and decoder networks to pay more attention to the data points similar to $\boldsymbol{\mu}^{(k)}$. To measure the similarity between a data point x_i and the cluster centre $\boldsymbol{\mu}^{(k)}$, denoted by p_{ik} where $\mathbf{p}_i = [p_{i1}, \ldots, p_{iK}]$, we propose to solve the optimization problem shown in (3.2a) where $m \geq 1$ is the level of fuzziness [1]. As is shown in (3.2b), we use the Lagrangian multiplier method [227] to solve (3.2a) where the Lagrange multiplier γ is computed by substituting p_{ik} from (3.2b) in the constraint of (3.2a), as is shown in (3.2c). The final value for the similarity of data point $\mathbf{x}_i$ to the k^{th} cluster, i.e. p_{ik}, is obtained by substituting γ from (3.2c) in (3.2b), as is shown in (3.2d). It can be seen that samples that are closer to $\boldsymbol{\mu}^{(k)}$, in the

[1](3.2a) is inspired by the fuzzy c-means clustering method; the main difference is that in (3.2a) there are multiple representations for a single data point while in fuzzy c-means each data point has a single representation.

corresponding latent space $U^{(l_k)}$, take higher similarity values.

$$\begin{aligned} &\min_{p_{ik}} \textstyle\sum_{k=1}^{K} p_{ik}^{m} ||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}_k||_2^2 \\ &\textbf{s.t.} \quad \textstyle\sum_{k=1}^{K} p_{ik} = 1 \end{aligned} \tag{3.2a}$$

$$S = \textstyle\sum_{k=1}^{K} p_{ik}^{m} ||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}^{(k)}||_2^2 - \gamma(\sum_{k=1}^{K} p_{ik} - 1)$$

$$\xrightarrow{\frac{\partial S}{\partial p_{ik}}=0} \qquad p_{ik} = \left(\frac{\gamma}{m||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}^{(k)}||_2^2}\right)^{\frac{1}{m-1}} \tag{3.2b}$$

$$\xrightarrow{\sum_k p_{ik}=1} \quad \gamma = \frac{1}{m}\Big(\textstyle\sum_{k=1}^{K} \frac{1}{\frac{1}{||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}^{(k)}||_2^{2/(m-1)}}}\Big)^{m-1} \tag{3.2c}$$

$$\xrightarrow{3.2b} \qquad p_{ik} = \frac{\frac{1}{||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}_k||_2^{2/(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||\mathbf{u}_i^{(l_j)} - \boldsymbol{\mu}_j||_2^{2/(m-1)}}} \tag{3.2d}$$

Note that, since $\sum_{k=1}^{K} p_{ik} = 1$, one can also think of p_{ik} as a soft assignment of data point $\mathbf{x}_i$ to the k^{th} cluster; hence, the two terms "similarity" and "soft assignment" are interchangeable throughout this proposal.

3.3 Training Procedure

As discussed before, we propose to use weighted samples when training the autoencoder corresponding to the k^{th} cluster, i.e. AE$^{(l_k)}$. We consider the soft assignment p_{ik} (defined in Section 3.2) as the weight of sample $\mathbf{x}_i$ when training AE$^{(l_k)}$. This is to realize the idea of assigning higher weights to the samples closer to the target cluster centre $\boldsymbol{\mu}^{(k)}$, in the corresponding latent space $U^{(l_k)}$.

We define the loss function $\mathcal{L}^{(l_k)}$, shown in (3.3), to be minimized when training AE$^{(l_k)}$, where $\mathcal{L}_r^{(l_k)}$ and $\mathcal{L}_c^{(l_k)}$ respectively denote the weighted reconstruction and clustering losses, and λ is a hyperparameter that indicates the effect of the clustering loss in the networks training. AE$^{(l_k)}$'s parameters, i.e. $\boldsymbol{\theta}_e^{(l_k)}$and $\boldsymbol{\theta}_d^{(l_k)}$for $k = 1, \ldots,$ K, are optimized in an end-to-end manner using the back-propagation algorithm while minimizing $\mathcal{L}^{(l_k)}, k = 1, \ldots,$ K.

$$\mathcal{L}^{(l_k)} = \mathcal{L}_r^{(l_k)} + \lambda \mathcal{L}_c^{(l_k)} \tag{3.3a}$$

$$\mathcal{L}_r^{(l_k)} = \textstyle\sum_{\mathbf{x}_i \in \mathfrak{B}} p_{ik}^m ||\mathbf{x}_i - \hat{\mathbf{x}}_i^{(l_k)}||_2^2 \tag{3.3b}$$

$$\mathcal{L}_c^{(l_k)} = \textstyle\sum_{x_i \in \mathfrak{B}} p_{ik}^m ||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}_k||_2^2 \tag{3.3c}$$

Most of the existing deep-clustering methods (see Chapter 2) only consider the reconstruction loss, to be minimized, in the hope of making the latent space more discriminative for data clustering, while the reconstruction loss has no substantial connection with the clustering performance. The proposed loss function shown in (3.3) considers both of the influential factors reconstruction and clustering performances through incorporating their corresponding losses. This allows AE$^{(l_k)}$ to simultaneously learn a feature representation and compact data points with similar latent representations around their corresponding cluster center.

By incorporating the soft assignment p_{ik} in the reconstruction loss $\mathcal{L}_r^{(l_k)}$, we direct the encoder and decoder networks to be specialized in reconstructing samples that are more probable to belong to the k^{th} cluster. Similarly, including p_{ik} in the clustering loss $\mathcal{L}_c^{(l_k)}$ encourages data points that are more probable to belong to the k^{th} cluster to sit close to their corresponding cluster center $\boldsymbol{\mu}^{(k)}$, in the latent space $U^{(l_k)}$. Thus, by minimizing $\mathcal{L}_c^{(l_k)}$, we implicitly minimize the intra-cluster distances between samples of the k^{th} cluster.

Finally, to update the cluster center $\boldsymbol{\mu}^{(k)}$, we define total loss function $\mathcal{L}_t^{(l_k)}$, $k = 1, ..., \mathrm{K}$, and set its derivative to 0, as is shown in (3.4).

$$\begin{aligned} \mathcal{L}_t^{(l_k)} &= \sum_{\mathbf{x}_i \in X} p_{ik}^m \Big(||\mathbf{x}_i - \hat{\mathbf{x}}_i^{(l_k)}||_2^2 + \lambda ||\mathbf{u}_i^{(l_k)} - \boldsymbol{\mu}_k||_2^2 \Big) \\ &\xrightarrow{\frac{\partial \mathcal{L}_t^{(k)}}{\partial \boldsymbol{\mu}_k} = 0} \boldsymbol{\mu}_k = \frac{\sum_{\mathbf{x}_i \in X} p_{ik}^m \mathbf{u}_i^{(l_k)}}{\sum_{\mathbf{x}_i \in X} p_{ik}^m} \end{aligned} \tag{3.4}$$

Algorithm 1 DML Algorithm

```
    Input: Data points $X$, $\tilde{\boldsymbol{\theta}}_e$, $\tilde{\boldsymbol{\theta}}_d$, $\boldsymbol{\mu}^{(k)}$ for $k = 1, \ldots,$ K, $\tau$, and MaxIter.
    Output: $\boldsymbol{\theta}_e^{(l)}$, $\boldsymbol{\theta}_d^{(l)}$ for $l = 1, \ldots,$ L.
1: Find sets $\mathcal{D}$ and $\mathcal{E}$ based on Section 3.1.
2: Initialize $\boldsymbol{\theta}_e^{(l)}$ and $\boldsymbol{\theta}_d^{(l)}$ for $l = 1, \ldots,$ L with $\tilde{\boldsymbol{\theta}}_e$ and $\tilde{\boldsymbol{\theta}}_d$ respectively.
3: for $iter \in \{1, 2, ..., \text{MaxIter}\}$ do
4:     for $k \in \{1, 2, ..., \text{K}\}$ do
5:         Compute soft assignments $p_{ik}$ using (3.2d), for $i \in \mathfrak{B}$
6:         Update AE$^{(l_k)}$'s parameters employing loss function (3.3)
7:     end for
8: end for
9: Use crisp assignment based on Section 3.4 to assign data point to clusters
```

3.4 Final Cluster Assignments

We utilize the trained encoders and cluster centers to compute the final degree of membership, p_{ik}, based on (3.2d). Each data point is assigned to the most probable cluster. The pseudo code of DML is presented in Algorithm 1. Block diagram of DML is shown in Fig. 3.1.

3.5 Experiments and Results

In the previous sections of this chapter, we introduced the DML algorithm, designed to enhance autoencoder-based methods by leveraging cluster-specific autoencoders for “difficult” clusters. Within the DML framework, we propose a novel loss function that places greater emphasis on data points closer to cluster centers, ensuring that the model develops meaningful, cluster-specific representations. In this section, we evaluate the effectiveness of our proposed DML framework and its novel loss function using six benchmark datasets: MNIST [228], Fashion MNIST [229], 2MNIST, CIFAR-10 [230], STL-10 [231], and CIFAR-100 [230]. To achieve this, we conduct an extensive set of experiments and assess the performance using two standard clustering metrics: clustering accuracy (ACC) [232] and normalized mutual information (NMI) [233]. ACC represents the percentage of data points correctly assigned to their clusters out of the total number of data points, offering a straightforward measure of the clustering algorithm’s effectiveness. Conversely, NMI evaluates the similarity between two clusterings by determining the mutual information shared among their cluster assignments. Both ACC and NMI scores range from 0 to 1, with higher values indicating superior clustering performance. In accordance with standard practices in the clustering literature [234–236], we present the mean and standard deviation of the results across random experiments. For DML specifically, we conduct 10 experimental runs to ensure the robustness and reliability of our evaluation.

3.5.1 Clustering Performance

The clustering performance of the DML framework is compared against thirteen traditional and state-of-the-art clustering methods. k-means [2], large-scale spectral clustering (LSSC) [237], and locality preserving non-negative matrix factorization (LPMF) [238] are among the most commonly used conventional clustering algorithms. Deep learning-based algorithms include deep embedding clustering (DEC) [17], improved deep embedding clustering (IDEC) [18], deep clustering network (DCN) [25], deep k-means (DKM) [19], variational deep embedding (VaDE) [239], GAN mixture model for clustering (GANMM) [168], and very recent methods contrastive clustering (CC) [23] and deep successive learning (DSL) [240]. In addition to these algorithms, we compare our results with scalable deep k-subspace clustering (SDkC), which offers a scalable and efficient approach for subspace clustering through the application of deep learning. SDkC is one of the few deep subspace clustering (DSC) methods capable of handling large-scale datasets that feature high-dimensional features and multiple subspaces. Furthermore, we report the clustering performance of the baseline approach AE + k-means, in which k-means is simply applied to the latent representation of an AE that has a similar architecture to the AE's used in the DML method; the AE in AE + k-means is trained to minimize the data reconstruction loss. Note that, in all the below Tables and Figures, for the comparison methods, we executed the code released by the authors with the same hyper-parameters specified in the original papers if the results of interest are not reported in the corresponding original paper. When the code is not publicly available or not applicable to the dataset, we put dash marks (-) instead of the corresponding results.

The effectiveness of DML in boosting the performance of the state-of-the-art AE-based clustering methods is shown in Table 3.2. Note that in general, DML would be effective for the AE-based

Table 3.1: ACC and NMI on the benchmark datasets for different clustering methods. The mean is on the first line, and the standard deviation is placed on the second line for our algorithm.

Datasets / Method	MNIST		Fashion MNIST		2MNIST		CIFAR10		STL10		CIFAR100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means	53.20	50.00	47.40	51.20	32.31	44.00	22.90	8.70	19.20	12.50	13.00	8.40
LSSC	71.40	70.60	49.60	49.70	39.77	51.22	21.14	10.89	18.75	11.68	14.60	7.92
LPMF	47.10	45.20	43.40	42.50	34.68	38.69	19.10	8.10	18.00	9.60	11.80	7.90
DEC	84.30	83.72	51.80	54.63	41.20	53.12	30.10	25.70	35.90	27.60	18.50	13.60
IDEC	88.13	83.81	52.90	55.70	40.42	53.56	36.99	32.53	32.53	18.85	19.61	14.58
DCN	83.00	81.00	51.22	55.47	41.35	46.89	30.47	24.58	33.84	24.12	20.17	12.54
DKM	84.00	81.54	51.31	55.57	41.75	46.58	35.26	26.12	32.61	29.12	18.14	12.30
AE + k-means	86.03	80.25	57.94	57.15	44.01	62.80	80.11	70.35	95.89	91.75	49.86	48.57
VaDE	94.50	87.60	50.39	59.63	56.60	51.20	29.10	24.50	28.10	20.00	15.20	10.80
GANMM	64.00	61.00	34.00	27.00	50.12	49.35	-	-	-	-	-	-
SDkC	83.30	77.38	60.02	62.30	-	-	-	-	-	-	-	-
CC	88.56	84.21	**64.52**	61.45	42.15	58.89	77.00	67.80	85.00	76.40	42.30	42.10
DSL	96.22*	90.66*	62.90	63.58*	45.31*	63.00*	83.40*	71.32*	96.02*	91.90*	50.30*	49.80*
DML-DSL	**96.36**	**91.24**	63.20*	**64.80**	**45.83**	**63.40**	**84.15**	**71.70**	**96.45**	**92.11**	**50.68**	**50.19**
	±0.15	±0.22	±0.21	±0.25	±0.34	±0.23	±0.36	±0.29	±0.13	±0.18	±0.22	±0.15

Table 3.2: ACC and NMI on the benchmark datasets for method $\mathcal{M}$ and DML-$\mathcal{M}$ where $\mathcal{M} \in \{$DCN, DKM, DSL$\}$. The mean is on the first line, and the standard deviation is placed on the second line for our algorithms.

Datasets / Method	MNIST		Fashion MNIST		2MNIST		CIFAR10		STL10		CIFAR100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DCN	83.00	81.00	51.22	55.47	41.35	46.89	30.47	24.58	33.84	24.12	20.17	12.54
DKM	84.00	81.54	51.31	55.57	41.75	46.58	35.26	26.12	32.61	29.12	18.14	12.30
DSL	96.22	90.66	62.90	63.58	45.31	63.00	83.40	71.32	96.02	91.90	50.30	49.80
DML-DCN	**87.48**	**81.63**	**55.57**	**55.90**	**44.79**	**58.23**	**31.51**	**25.00**	**34.05**	**25.14**	**21.56**	**13.58**
	±0.68	±0.31	±0.45	±0.71	±0.34	±0.29	±0.15	±0.23	±0.19	±0.21	±0.31	±0.26
DML-DKM	**91.18**	**82.58**	**55.36**	**56.00**	**44.66**	**58.30**	**36.28**	**27.00**	**34.10**	**31.21**	**18.56**	**12.69**
	±0.78	±0.34	±0.16	±0.12	±0.84	±0.72	±0.21	±0.18	±0.29	±0.34	±0.22	±0.24
DML-DSL	**96.36**	**91.24**	**63.20**	**64.80**	**45.83**	**63.40**	**84.15**	**71.70**	**96.45**	**92.11**	**50.68**	**50.19**
	±0.15	±0.22	±0.21	±0.25	±0.34	±0.23	±0.36	±0.29	±0.13	±0.18	±0.22	±0.15

clustering algorithms which aim at creating hyperspheres of data clusters in a lower dimensional space, such as DCN [25], DKM [19] and DSL [240]. In Table 3.2, DML-$\mathcal{M}$ refers to the performance of DML when the AE of algorithm $\mathcal{M}$ is used as the DML's autoencoders. As it can be seen from the table, DML significantly improves the clustering performance of the base method $\mathcal{M}$ mainly due to assigning cluster-specific AEs to the difficult data clusters. On average, DML improves ACC (NMI) of DCN, DKM, and DSL, respectively, by 2.50% (2.48%), 2.84% (2.75%), and 0.42% (0.55%). Less significant improvement in DSL compared to that of DCN and DKM could be associated with the fact that DSL implicitly has some sort of cluster-specific training procedure when training its AE, but note that DSL still trains a single common latent space for all of the data clusters. To further show the effectiveness of the proposed DML framework, considering that DSL [240] is the most recent and effective AE-based clustering algorithm, we pick DSL as the base of DML and compare DML-DSL with more SOTA algorithms in Table 3.1. As can be seen, DML-DSL outperforms all comparison methods on 75 out of 76 reported results. Each dataset's best result is shown in bold. The second-top results are denoted by an asterisk (*).

For the DML-$\mathcal{M}$ method, we use the same network structure suggested by the $\mathcal{M}$ method for each dataset, where $\mathcal{M} \in \{$DCN [25], DKM [19], DSL [240]$\}$. We first train a single AE using the $\mathcal{M}$ method to obtain the initial network parameters ($\tilde{\theta}_e$ and $\tilde{\theta}_d$) and cluster centers $\mu^{(k)}$. We then train L AEs and update cluster centers based on our proposed algorithm in Section 3. For all datasets $\tau = \frac{d_{min}+d_{max}}{4}$, where d_{min} and d_{max} are the minimum and maximum distance between initial cluster centers, respectively. We choose $\lambda \in \{0.001, 0.01, 0.1\}$ based on the accuracy of our model on the validation set for each dataset. Moreover, in DML-$\mathcal{M}$, we use the same data pre-processing technique as what is used in the original $\mathcal{M}$ method.

3.5.2 t-SNE Visualization

In Fig. 3.2, we further demonstrate the effectiveness of the proposed DML framework, by comparing different data representations of the MNIST dataset, using t-SNE visualization [241]. To this end, we use the trained cluster-specific AEs and the general AE to obtain the L latent representations of each data point. Then the most probable one is selected as the latent representation of the

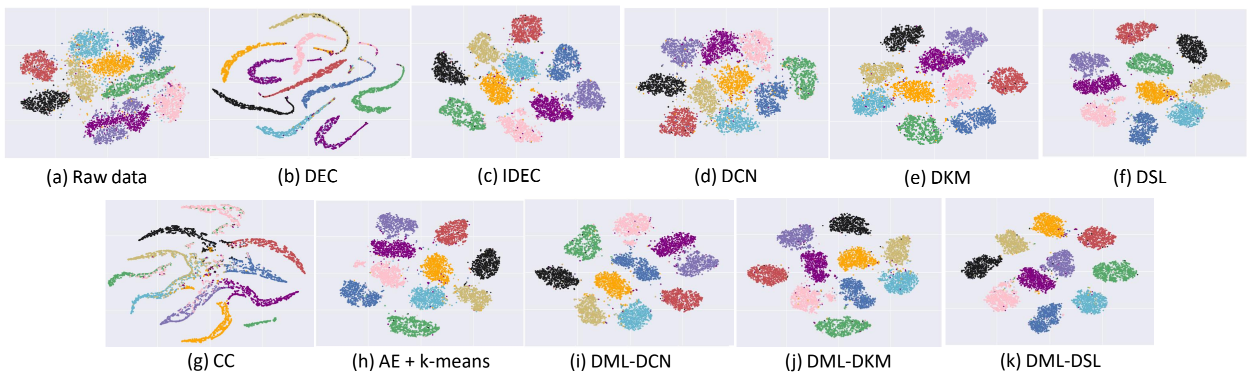


Figure 3.2: Clustering visualization of different methods using t-SNE, for MNIST dataset. Axes range from -100 to 100.

data, i.e., the latent space that provides the closest representation of the data to its corresponding center is chosen. Afterward, t-SNE method is utilized to map the latent representation to a 2D-space. The benefit of the proposed DML method in providing a clear distribution structure is more apparent if we compare t-SNE representation.

The effectiveness of our proposed multi-representation learning framework, and the proposed loss function, in minimizing intra-cluster distance(s) is apparent when we compare the clusters obtained by DML-$\mathcal{M}$ shown in Fig.3.2- (i), (j), (k) with $\mathcal{M}$ shown in Fig. 3.2- (d), (e), (f). The DML-$\mathcal{M}$ clusters are much more compactly distributed around their corresponding cluster centers. The improved performance of DML-$\mathcal{M}$ compared with $\mathcal{M}$ is more significant when looking at the separation of clusters colored in magenta and purple (digits 4 and 9), as well as the separation of clusters in cyan, olive and orange (digits 3, 5 and 8) when comparing Fig. 3.2-(i) with 3.2-(d), and Fig. 3.2-(j) with 3.2-(e), and Fig. 3.2-(k) with Fig. 3.2-(f).

3.5.3 Performance on Imbalanced Datasets

One of the main advantages of the proposed multi-representation learning method is its outstanding ability in dealing with imbalanced datasets since, in contrast with the state-of-the-art methods that provide a common latent space for all clusters, DML dedicates cluster-specific AEs for difficult clusters. To show the effectiveness of the proposed framework on imbalanced data, we sample subsets of MNIST with various retention rates $r \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, where data points of

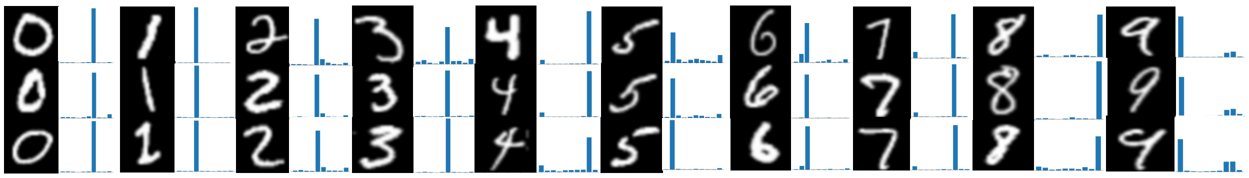
Table 3.3: ACC and NMI on imbalanced datasets for different clustering methods. The mean is on the first line, and the standard deviation is placed on the second line for our algorithms.

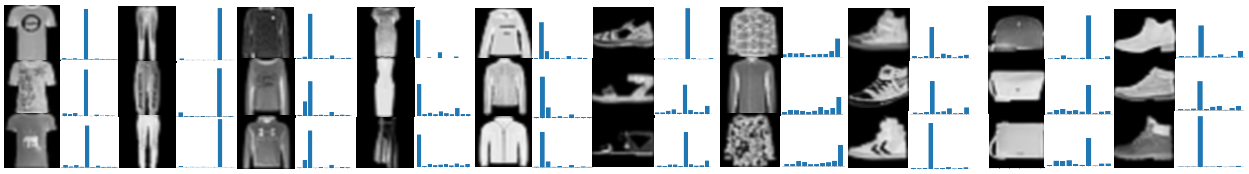
Method \ r	0.1		0.2		0.3		0.4		0.5	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
AE + k-means	82.61	78.51	83.11	78.99	84.78	81.48	84.83	81.55	85.60	83.70
DCN	72.74	68.80	73.02	72.91	79.60	72.80	74.52	73.61	76.32	74.62
DKM	45.96	39.21	46.21	39.54	48.56	38.96	51.25	42.10	50.98	43.27
DSL	83.73	81.38	89.86	82.30	90.40	83.55	92.14	84.71	94.22	88.15
DML-DCN	**75.34**	**69.83**	**80.32**	**73.51**	**81.23**	**73.05**	**79.20**	**75.59**	**77.58**	**75.37**
	±0.35	±0.38	±0.29	±0.28	±0.35	±0.30	±0.20	±0.26	±0.33	±0.28
DML-DKM	**67.25**	**61.32**	**74.35**	**66.24**	**68.31**	**63.17**	**79.24**	**77.12**	**80.16**	**78.12**
	±0.27	±0.25	±0.16	±0.17	±0.28	±0.25	±0.31	±0.39	±0.22	±0.19
DML-DSL	**85.35**	**81.92**	**90.63**	**83.44**	**91.22**	**84.57**	**92.44**	**85.72**	**94.63**	**88.68**
	±0.30	±0.38	±0.27	±0.27	±0.22	±0.26	±0.24	±0.27	±0.19	±0.21

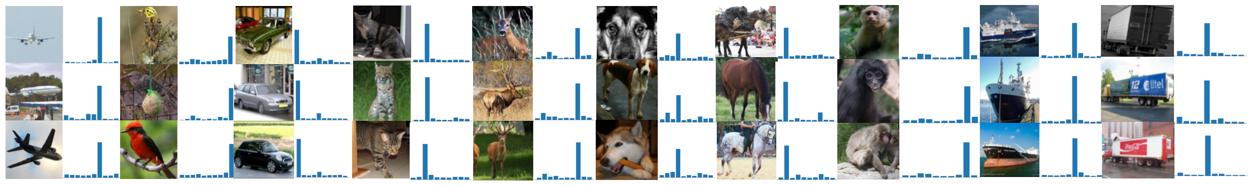
class 0 are kept with probability r and class 9 with probability 1, with the other classes linearly in between 0 and 1. As such, the smallest cluster is r times smaller than the largest cluster. ACC and NMI for various r on all datasets are shown in Table 3.3. It can be seen that the proposed DML-$\mathcal{M}$ method significantly outperforms $\mathcal{M}$ for all r values, where $\mathcal{M} \in \{\text{DSL, DKM, DCN}\}$. DML-DCN, DML-DKM, and DML-DSL improve the performance of DCN, DKM, and DSL by 0.79% (0.85%), 25.25% (28.57%), and 3.51% (0.94%) in ACC (NMI) in average on the five datasets, which shows the effectiveness of our proposed training procedure of DML in learning useful representation of data points. For reference, the performance of the baseline method AE + k-means is also reported in Table 3.3.

3.5.4 Soft Assignments Visualization

Fig. 3.3 depicts soft cluster assignment vectors, i.e. $\mathbf{p_i}$, corresponding to samples from various data clusters for DML-DSL. The k^{th} element of $\mathbf{p_i}$ shows the probability of sample x_i belonging to the k^{th} cluster. We observe that, with a high probability, the index of the highest element in vector $\mathbf{p_i}$ of samples from the same ground-truth labels, are the same – in other words, with a


(a) MNIST


(b) Fashion MNIST


(c) STL-10

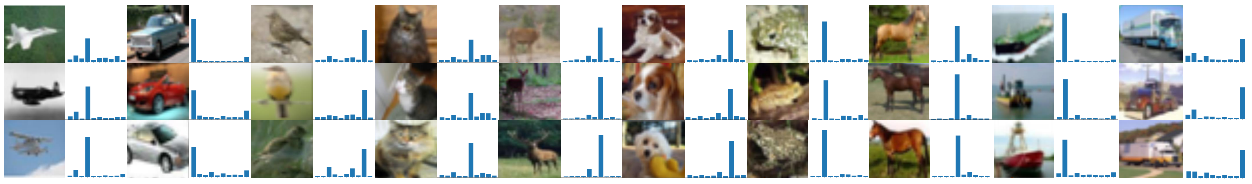

(d) CIFAR-10

Figure 3.3: Visualization of soft assignments vector $\mathbf{p_i}$ for samples from (a) MNIST, (b) Fashion MNIST, (c) STL-10, and (d) CIFAR-10 datasets. The vertical axes range from 0 to 1.

high probability, samples from the same ground truth class are mapped to the same cluster. Since DML dedicates an individual AEs to difficult clusters, it is capable of distinguishing more complex clusters, such as deer and horse. Moreover, DML reasonably recognizes the second most probable cluster for each sample. For example, in the STL-10 dataset, the second most probable cluster for truck is car, which is also a vehicle. One limitation of DML is that soft assignments often fail to fully converge to one-hot vectors when applied to more complex datasets like STL-10 and CIFAR-10. This can lead to negative effects, such as training instability and reduced cluster specificity in autoencoders. A key reason for this issue lies in the DML loss function, which aims to bring data samples closer to cluster centers. However, the cluster centers themselves may be positioned too

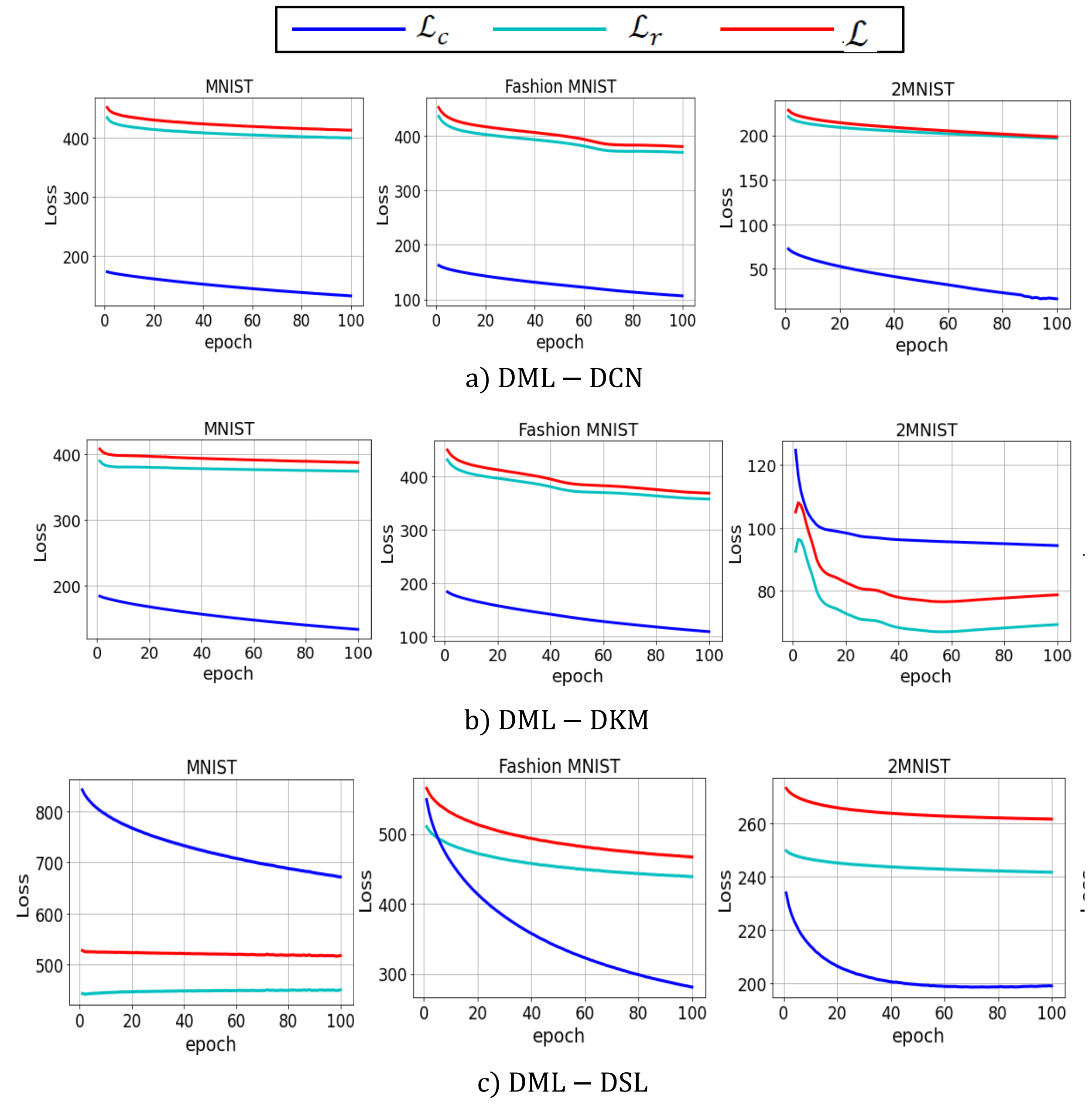


Figure 3.4: The average reconstruction loss $\mathcal{L}_r$, clustering loss $\mathcal{L}_c$, and total loss $\mathcal{L}$ of DML-$\mathcal{M}$ methods, $\mathcal{M} \in \{\text{DCN, DKM, DSL}\}$, for different datasets.

closely in the latent space, exacerbating the problem. In the next chapter, we address this challenge by introducing a novel loss function based on pairwise similarity between samples. This approach encourages cluster assignments to be more closely aligned with one-hot vectors during training.

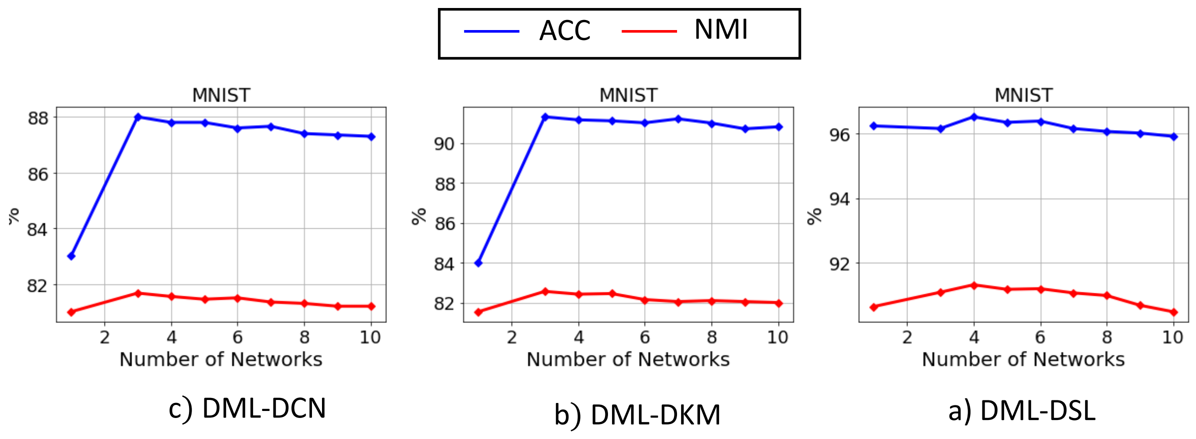


Figure 3.5: Effect of number of networks on clustering performance of DML-$\mathcal{M}$ where $\mathcal{M} \in \{\text{DCN, DKM, DSL}\}$.

3.5.5 Loss Function Convergence

The plot in Fig. 3.4 illustrates the average behavior of the reconstruction loss, clustering loss, and total loss across various DML-$\mathcal{M}$ methods. These values are computed by averaging over multiple networks and different batches of data points, providing a holistic view of the performance and consistency of the methods. The figure captures the convergence of all loss functions as training progresses. Notably, the significant reduction in the clustering loss demonstrates the effectiveness of our DML framework in enhancing clustering performance. This is achieved by moving data points closer to their corresponding cluster centers in the latent space, ultimately resulting in a more structured and optimized latent representation. Such an outcome indicates that the framework successfully aligns the latent space geometry with the underlying cluster structure in the data, improving the framework's utility for clustering tasks. Moreover, the reduction in reconstruction loss reflects the framework's ability to maintain accurate reconstructions of the original data. When combined, these improvements in reconstruction and clustering losses contribute to the minimization of the total loss, which is evident from the observed convergence at the end of training. These results underline the capability of the DML framework to balance reconstruction and clustering objectives effectively, enabling it to build a robust and meaningful latent space representation.

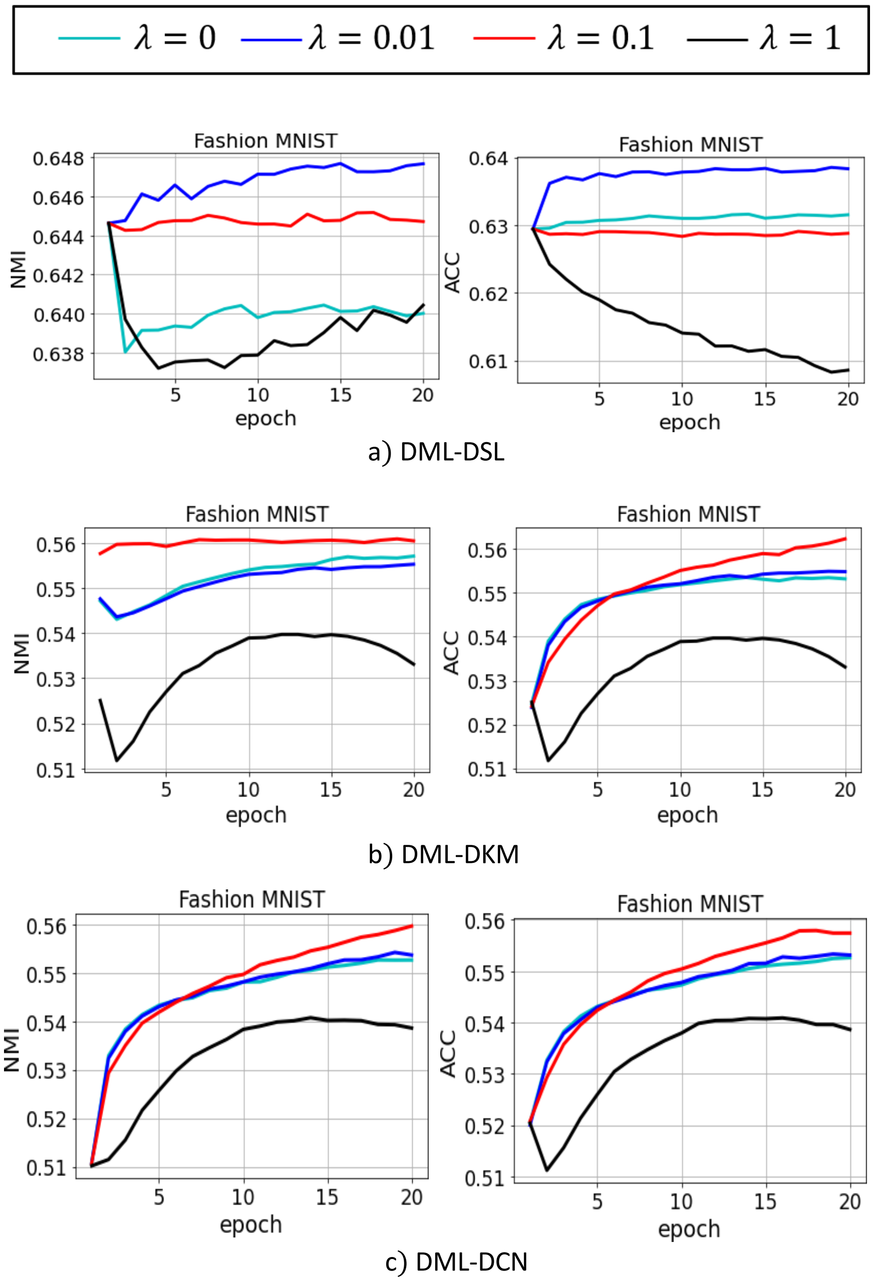


Figure 3.6: Effect of hyperparameter λ on clustering performance of DML-$\mathcal{M}$ where $\mathcal{M} \in \{$DCN, DKM, DSL$\}$.

3.5.6 Hyperparameters Sensitivity

In Fig. Fig. 3.5 and 3.6, we investigate the effect of the DML's hyperparameters λ and the number of AE networks on the clustering performance of DML-$\mathcal{M}$.

In Fig. 3.5, we explore the impact of the total number of AEs on the clustering performance of DML-DCN, DML-DKM, and DML-DSL for the MNIST dataset. What we can infer from this figure are as follows. 1) there is a significant improvement in the clustering performance of DML-DCN and DML-DKM when we increase number of networks from one to three, which shows the effectiveness of having multiple AEs in finding effective representations for data points. The fact that the DSL itself implicitly includes some form of cluster specific training technique when training its AE could explain the less significant improvement obtained for DSL compared to the DCN and DKM cases. 2) As is expected, increasing the number of networks does not always lead to a model with better performance. For example, the best model performance in terms of ACC and NMI for DML-DSL is achieved when four cluster-specific networks are trained for the MNIST dataset. Note that, we proposed an automatic approach, using τ, to determine the number of required AEs per dataset (see Section 3.5.1).

In Fig. 3.6, we scrutinize the effect of hyperparameter λ on clustering performance of DML-$\mathcal{M}$ for the Fashion MNIST dataset, where $\lambda \in \{0, 0.01, 0.1, 1\}$. λ indicates importance of the clustering loss in the total loss function of DML (See Section 3.3). We observe that for large values of λ, e.g., $\lambda = 1$, DML mainly concentrates on centering the data points near cluster centers and ignores the informative features that provide a low data reconstruction error. For relatively small values of λ, e.g. $\lambda = 0$, DML ignores the clustering loss and only focuses on minimizing the data reconstruction loss, which may mislead the DML in assigning the data points to the correct clusters. The best clustering performance of DML-DCN, DML-DKM, and DML-DSL on the Fashion MNIST dataset are respectively obtained when λ is 0.1, 0.1, and 0.01.

3.6 Limitations of DML

In this chapter, we have explored Deep Multi-Assignment Learning (DML), a framework designed to improve clustering performance in AE-based methods by assigning cluster-specific autoencoders to challenging clusters. While DML shows promise in addressing some of the challenges inherent in clustering complex datasets, it also presents some limitations that hinder its scalability and effectiveness in certain scenarios.

One of the primary limitations of DML is its memory inefficiency, which becomes particularly problematic when dealing with a large number of difficult clusters. Since DML assigns a separate autoencoder to each challenging cluster, it requires storing all these networks in memory during both the training and inference phases. As the number of clusters increases, the memory requirements scale proportionally, creating a significant bottleneck for applications involving large datasets. Also, assigning samples to different clusters can be computationally expensive and time-intensive, making DML less suitable for real-time clustering applications where efficiency is critical. Another drawback of DML is its inability to utilize the pairwise relationships between samples for clustering, a strategy proven effective in other methods such as [31,43]. This limitation can result in the soft assignments of samples failing to converge to one-hot vectors during training, particularly for challenging datasets like STL-10 and CIFAR-10 (see Section 3.5.4). When cluster assignments do not converge, the associated AEs face difficulty in adequately specializing for their respective clusters, which ultimately reduces the overall performance of the framework.

To address these challenges, we propose significant improvements in the next chapter. First, we replace the cluster-specific autoencoders with a single unified autoencoder that is trained to handle all clusters. This approach drastically reduces memory overhead while retaining the capacity to process challenging clusters effectively. Second, we introduce a novel loss function based on pairwise similarity between samples. Unlike the current DML loss, this new objective encourages cluster assignments to align more closely with one-hot vectors during training. By fostering clearer and more distinct cluster assignments, the new loss function enhances the stability and specificity of the clustering process, even for complex datasets.

3.7 Summary

In this chapter, we presented an effective and practical method for obtaining cluster-specific latent spaces for data clustering. Unlike most deep learning-based clustering methods, which rely on a single global latent space to represent all clusters, the proposed algorithm introduces a novel approach that assigns individual cluster-specific latent spaces to challenging clusters while leveraging a shared, common latent space for clusters that are comparatively easier to separate. This adaptive strategy is designed to enhance the representation quality and clustering performance by addressing the diverse complexities of different clusters within the data. The core of our method lies in the introduction of cluster-specific losses, which are formulated based on a combination of weighted reconstruction losses and clustering losses. These losses are carefully designed to ensure that the latent representations are not only compact and discriminative but also tailored to the specific characteristics of each cluster. The proposed multi-representation learning framework dynamically allocates resources to improve the separability of difficult clusters without compromising the overall performance of the model on easier clusters. The effectiveness of the proposed framework and loss functions is rigorously validated through an extensive set of experiments conducted on multiple benchmark datasets. Our results consistently demonstrate that the method outperforms existing state-of-the-art clustering approaches in terms of clustering accuracy, robustness, and representation quality. Additionally, we analyze the impact of the cluster-specific latent spaces on the overall clustering process and provide insights into how the proposed method achieves superior performance by mitigating challenges such as imbalanced cluster sizes.

Chapter 4

Deep clustering with self-supervision (DCSS)

In this chapter, we present a groundbreaking framework called Deep Clustering with Self-Supervision using Pairwise Similarities (DCSS). This method employs a two-phase approach designed to overcome the challenges of DML while significantly improving clustering performance. In the first phase, DCSS introduces cluster-specific loss functions, enabling the model to focus on reconstructing and centering individual clusters. This process creates hypersphere-like representations in the latent space, ensuring clusters are both compact and well-separated. Unlike DML, which requires multiple AEs for different clusters, DCSS optimizes memory usage by employing a single AE with a specially designed cluster-specific loss function. The second phase incorporates a novel strategy using additional layers and an auxiliary network, MNet, to refine the representations. MNet manages pairwise relationships by enhancing the similarity of related data points while diminishing the similarity of unrelated ones. This refinement produces representations tailored for clustering, improving both performance and adaptability to complex data structures. Our DCSS framework demonstrates superior performance, outperforming state-of-the-art and conventional algorithms on challenging datasets. This advantage highlights its robustness and effectiveness in handling complex data distributions, setting a new benchmark for clustering methodologies.

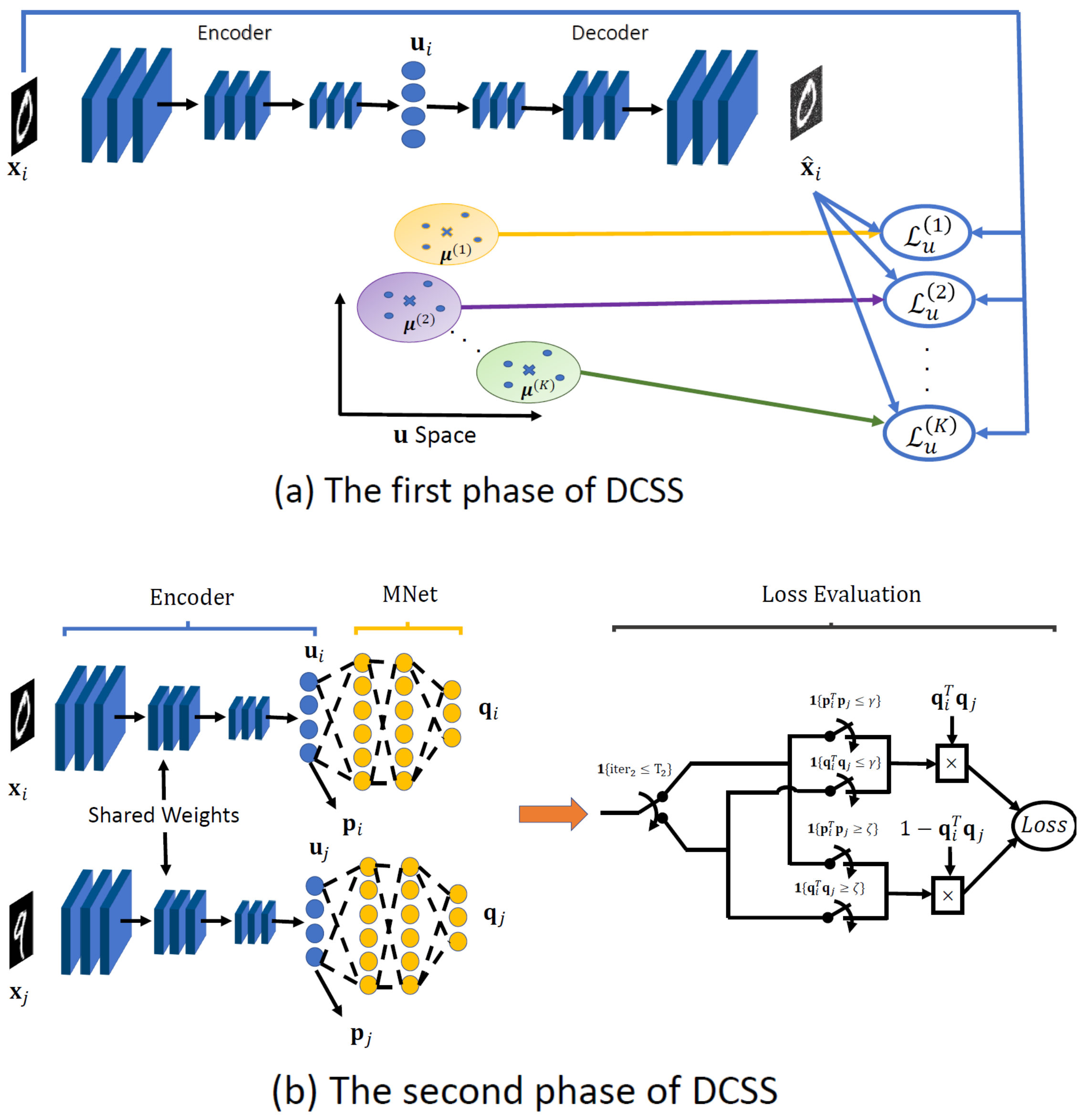

Encoder
$\mathbf{u}_i$
Decoder
$\mathbf{x}_i$
$\hat{\mathbf{x}}_i$
$\boldsymbol{\mu}^{(1)}$
$\boldsymbol{\mu}^{(2)}$
$\boldsymbol{\mu}^{(K)}$
$\mathcal{L}_u^{(1)}$
$\mathcal{L}_u^{(2)}$
$\mathcal{L}_u^{(K)}$
$\mathbf{u}$ Space
(a) The first phase of DCSS
Encoder
MNet
Loss Evaluation
$\mathbf{u}_i$
$\mathbf{q}_i$
$\mathbf{x}_i$
$\mathbf{p}_i$
Shared Weights
$\mathbf{u}_j$
$\mathbf{q}_j$
$\mathbf{x}_j$
$\mathbf{p}_j$
$\mathbf{1}\{\text{iter}_2 \leq \text{T}_2\}$
$\mathbf{1}\{\mathbf{p}_i^T\mathbf{p}_j \leq \gamma\}$
$\mathbf{1}\{\mathbf{q}_i^T\mathbf{q}_j \leq \gamma\}$
$\mathbf{1}\{\mathbf{p}_i^T\mathbf{p}_j \geq \zeta\}$
$\mathbf{1}\{\mathbf{q}_i^T\mathbf{q}_j \geq \zeta\}$
$\mathbf{q}_i^T\mathbf{q}_j$
$1 - \mathbf{q}_i^T\mathbf{q}_j$
×
Loss
(b) The second phase of DCSS

Figure 4.1: (a) Training scheme of DCSS Phase 1. (b) Phase 2 training of DCSS: Initially, when $\text{iter}_2 \leq T_2$, MNet uses pairwise similarities in the $\mathbf{u}$ space, defined by the dot product of $\mathbf{p}_i$ and $\mathbf{p}_j$. For $\text{iter}_2 > T_2$, similarities are measured directly in the $\mathbf{q}$ space using $\mathbf{q}_i^T\mathbf{q}_j$.

4.1 Notation

The K-clustering problem seeks to partition a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, consisting of N samples, into K non-overlapping groups (or clusters). Here, the i-th data sample is denoted by $\mathbf{x}_i$, and K is the predefined number of clusters. In this chapter, we consider K-clustering and propose DCSS frameworks to address the problem. DCSS utilizes an AE, consisting of an encoder and a decoder network respectively denoted by $f(.)$ and $g(.)$. Latent representation of X is denoted by $U = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$, where $\mathbf{u}_i = f(\mathbf{x}_i; \boldsymbol{\theta}_e) \in \mathbb{R}^d$, d indicates dimension of the latent space, and $\boldsymbol{\theta}_e$ denotes parameters of the encoder network. The reconstructed output of the AE is denoted by $\hat{\mathbf{x}}_i = g(\mathbf{u}_i; \boldsymbol{\theta}_d)$, where $\boldsymbol{\theta}_d$ represents the decoder parameters. Center of the kth data group in the $\mathbf{u}$ space is denoted by $\boldsymbol{\mu}^{(k)}$. To accommodate complex cluster distributions, we propose to employ pairwise data similarities in DCSS. To this end, we employ the fully connected network MNet which takes the latent representation of each data point, i.e. $\mathbf{u}_i$, as input and maps it to a K-dimensional vector $\mathbf{q}_i$ whose kth element indicates the probability of $\mathbf{x}_i$ belonging to the kth data cluster. In this chapter, the output of MNet for the ith data point is denoted by $\mathbf{q}_i = M(\mathbf{u}_i; \boldsymbol{\theta}_M)$, where $M(.)$ and $\boldsymbol{\theta}_M$ respectively shows MNet and its corresponding parameters.

The proposed DCSS method consists of two phases. The first phase is to provide hypersphere-like data clusters through training an AE using weighted reconstruction and centering losses, and the second phase is to employ pairwise data similarities to self-supervise the remaining training procedure.

4.2 Phase 1: AE training

At each training batch $\mathfrak{B}$, we propose to train the AE in K successive runs, where at each run a specific loss corresponding to a specific data cluster is minimized. More specifically, at the kth run, the AE focuses on reconstruction and centering of the data points that are more probable to belong to the kth data cluster.

Loss function of the kth run, i.e. $\mathcal{L}_u^{(k)}$, is shown in (4.1a) where $\mathcal{L}_r^{(k)}$ and $\mathcal{L}_c^{(k)}$, shown in (4.1b) and (4.1c), respectively denotes weighted summation of the sample reconstruction and centering losses. α is a hyperparameter indicating the importance of centering loss vs. reconstruction loss.

m indicates the level of fuzziness and is set to 1.5 in all experiments.

$$\mathcal{L}_u^{(k)} = \mathcal{L}_r^{(k)} + \alpha \mathcal{L}_c^{(k)} \tag{4.1a}$$

$$\mathcal{L}_r^{(k)} = \textstyle\sum_{\mathbf{x}_i \in \mathfrak{B}} p_{ik}^m ||\mathbf{x}_i - \hat{\mathbf{x}}_i||_2^2 \tag{4.1b}$$

$$\mathcal{L}_c^{(k)} = \textstyle\sum_{\mathbf{x}_i \in \mathfrak{B}} p_{ik}^m ||\mathbf{u}_i - \boldsymbol{\mu}^{(k)}||_2^2 \tag{4.1c}$$

$$p_{ik} = \frac{\frac{1}{||\mathbf{u}_i - \boldsymbol{\mu}^{(k)}||_2^{2/(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||\mathbf{u}_i - \boldsymbol{\mu}^{(k)}||_2^{2/(m-1)}}} \tag{4.2}$$

Since data clustering is an unsupervised task, the data cluster memberships are unknown at the problem outset. As such, at the kth run, we use the Euclidean distance between $\mathbf{u_i}$ and $\boldsymbol{\mu}^{(k)}$ as a means of measuring the membership degree of $\mathbf{x}_i$ to the kth data cluster, denoted by p_{ik} defined in (4.2) where $\mathbf{p}_i = [p_{i1}, \dots, p_{iK}]$. The cluster memberships are used as the sample weights in (4.1b) and (4.1c). The closer a sample is to the cluster center $\boldsymbol{\mu}^{(k)}$, the higher contribution that sample has in minimizing the loss function corresponding to the kth run.

Every T_1 training epochs, we update the centers to the average of weighted samples in the $\mathbf{u}$ space, as is shown in (4.3), where samples closer to $\boldsymbol{\mu}^{(k)}$ have more contribution to updating.

$$\boldsymbol{\mu}^{(k)} = \frac{\sum_{\mathbf{x}_i \in X} p_{ik}^m \mathbf{u}_i}{\sum_{\mathbf{x}_i \in X} p_{ik}^m} \tag{4.3}$$

Block diagram of the first phase is shown in Fig. 4.1(a). As is demonstrated in our experiments (see Section 4.6.4), minimizing (4.1a), results in forming hypersphere-like groups of similar samples in the $\mathbf{u}$ space, one hypersphere per cluster. A preliminary version of the first phase of DCSS [240] is accepted by International Joint Conference on Neural Networks (IJCNN).

4.3 Phase 2: self-supervision using pairwise similarities

To allow accommodating non-hypersphere shape distributions and to employ the important information available in the pairwise data relations, we propose to append a fully connected network, called MNet, to the encoder part of the AE, trained in Phase 1, while discarding its decoder network. The MNet's output layer, i.e. the $\mathbf{q}$ space, consists of K neurons where each neuron corresponds to a data cluster. We utilize the soft-max function at the output layer to obtain probability values employed for obtaining the final cluster assignments. More specifically, for an input sample $\mathbf{x}_i$, the output value at the jth neuron, i.e. q_{ij}, denotes the probability of $\mathbf{x}_i$ belonging to the jth cluster.

MNet aims to strengthen (weaken) similarities of two similar (dissimilar) samples. MNet parameters, i.e., $\boldsymbol{\theta}_M$, are initialized with random values. Hence, at the first few training epochs, when $\mathbf{q}$ is not yet a reliable space, pairwise similar and dissimilar samples are identified in the $\mathbf{u}$ space. Then after a few training epochs, pairs of similar and dissimilar samples are identified in the $\mathbf{q}$ space. In both of the $\mathbf{u}$ and $\mathbf{q}$ spaces, we define two samples as similar (dissimilar) if the inner product of their corresponding cluster assignment vectors is greater (lower) than threshold ζ (γ). More specifically, knowing that the kth element of $\mathbf{p}_i$ ($\mathbf{q}_i$) denotes the membership of $\mathbf{x}_i$ to the kth cluster in the $\mathbf{u}$ ($\mathbf{q}$) space, the inner product of $\mathbf{p}_i$ ($\mathbf{q}_i$) and $\mathbf{p}_j$ ($\mathbf{q}_j$) is considered as the notion of similarity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$.

The loss function proposed for the MNet training, at the first T_2 training epochs, is shown in (4.4) where ζ and γ are two user-settable hyperparameters, and $\mathbb{1}\{.\}$ is the indicator function.

$$\mathcal{L}_M = \sum_{\mathbf{x}_i,\mathbf{x}_j \in \mathfrak{B}} \mathbb{1}\{\mathbf{p}_i^T \mathbf{p}_j \geq \zeta\}(1 - \mathbf{q}_i^T \mathbf{q}_j) + \mathbb{1}\{\mathbf{p}_i^T \mathbf{p}_j \leq \gamma\}(\mathbf{q}_i^T \mathbf{q}_j) \tag{4.4}$$

As can be inferred from (4.4), only similar and dissimilar samples, identified in the $\mathbf{u}$ space, contribute to the MNet training and a pair of samples with a similarity value between ζ and γ, i.e. in the ambiguity region, does not contribute to the current training epoch. Therefore, minimizing $\mathcal{L}_M$ strengthens (weakens) the similarity of similar (dissimilar) samples, in the $\mathbf{q}$ space. Along with training the MNet parameters, the encoder parameters $\boldsymbol{\theta}_e$ are also updated through back-propagation, in an end-to-end manner. After completing each training epoch, centers $\boldsymbol{\mu}^{(k)}, k = 1, \ldots,$ K, are also updated using (4.3).

After T_2 epochs, when $\mathbf{q}$ becomes a relatively reliable space for identifying similar and dissimilar samples, we further train MNet using the loss function $\mathcal{L}'_M$ defined in (4.5). A pair contributes to $\mathcal{L}'_M$ if its corresponding similarity value, in the $\mathbf{q}$ space, is not in the ambiguity region. As is demonstrated in Section 4.6, as the MNet training phase progresses, more and more pairs contribute to the training procedure. Again, the $\mathbf{u}$ space receives small updates through the back propagation process when minimizing $\mathcal{L}'_M$.

$$\mathcal{L}'_M = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathfrak{B}} \mathbb{1}\{\mathbf{q}_i^T \mathbf{q}_j \geq \zeta\}(1 - \mathbf{q}_i^T \mathbf{q}_j) + \mathbb{1}\{\mathbf{q}_i^T \mathbf{q}_j \leq \gamma\}(\mathbf{q}_i^T \mathbf{q}_j) \tag{4.5}$$

As is proved in Section 4.5, a proper choice of hyperparameters ζ and γ is respectively $\frac{2}{3} < \zeta$ and $\gamma < \zeta^2$. In our experiments ζ and γ are set to 0.8 and 0.2, respectively. Moreover, Section 4.5 presents several mathematical proofs that show, under certain assumptions, the final $\mathbf{q}$ vector for a query sample is very close to an one-hot vector where the index of the maximum element of the vector indicates the cluster label. This also shows that similar (dissimilar) samples tend to sit in the same (different) data cluster(s). Fig. 4.1(b) shows the overall training procedure of the DCSS's second phase.

4.4 Final Cluster Assignments

To determine the final cluster assignment of a data point $\mathbf{x}_i$, we utilize the trained encoder and MNet networks to obtain the data representation in the $\mathbf{q}$ space, i.e. $\mathbf{q}_i$. $\mathbf{x}_i$ is assigned to the most probable cluster, i.e. the index corresponding to the highest element of $\mathbf{q}_i$ is the cluster label of $\mathbf{x}_i$.

Algorithm 2 Clustering procedure using DCSS

```
    Input: Data points $X$, $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$, $\boldsymbol{\theta}_M$, $\boldsymbol{\mu}^{(k)}$ for $k = 1, \dots, \mathrm{K}$
    Output: $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_M$

    Phase 1:
 1: Initialize $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ with a pre-trained network (see Section 4.6.2).
 2: for iter$_1 \in \{1, 2, ..., \mathrm{MaxIter}_1\}$ do
 3:     for $k \in \{1, 2, ..., \mathrm{K}\}$ do
 4:         Compute $p_{ik}$ using (4.2), for $i \in \mathfrak{B}$
 5:         Update AE's parameters by employing (4.1a) as loss function
 6:     end for
 7:     Every $T_1$ iterations, update cluster centers using (4.3)
 8: end for

    Phase 2:
 9: for iter$_2 \in \{1, 2, ..., \mathrm{MaxIter}_2\}$ do
10:     if iter$_2 \leq T_2$ then
11:         Compute vectors $\mathbf{p}_i$ for $i \in \mathfrak{B}$
12:         Compute vectors $\mathbf{q}_i$ for $i \in \mathfrak{B}$
13:         Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_M$ to minimize (4.4)
14:         Update centers $\boldsymbol{\mu}^{(k)}$, $k = 1, \dots, \mathrm{K}$ , using (4.3)
15:     else
16:         Compute $\mathbf{q}_i$ for $i \in \mathfrak{B}$
17:         Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_M$ to minimize (4.5)
18:     end if
19: end for

    Final Cluster Assignments:
20: Compute $\boldsymbol{q}_i$ for $\boldsymbol{x}_i$, $i = 1, \dots \mathrm{N}$
21: Assign each data sample to the most probable cluster
```

4.5 Proper choice of ζ and γ

In this section, we discuss the selection of optimal hyperparameters ζ and γ for our model. It is crucial to choose appropriate values for these hyperparameters, as the model's performance is greatly impacted by their values. When the value of ζ is high and γ is small (e.g., $\zeta = 0.9$ and $\gamma = 0.1$), the model tends to consider only a small subset of the available data during the second training phase, resulting in the neglect of crucial information between true similar and dissimilar samples. Conversely, when the value of ζ is low and γ is high (e.g., $\zeta = 0.5$ and $\gamma = 0.5$), the model may struggle to accurately distinguish between similar and dissimilar samples. Therefore, selecting optimal values for ζ and γ is critical to ensure the effectiveness of our model.

Notation clarification: Representation of the ith sample in the $\mathbf{q}$ space is shown by $\mathbf{q}_i$. The kth element of $\mathbf{q}_i$ is shown by $q_{ik}, k = 1, \dots, K$, where K is the number of data clusters. Note that $\mathbf{q}_i$ is the MNet output when the input sample is x_i. Since we employ soft-max as the final layer of MNet, $0 \leq q_{il} \leq 1$ where $1 \leq l \leq K$ and the ℓ_1-norm of $\mathbf{q}_i$ is equal to 1. Furthermore, parameters ζ and γ are values between 0 and 1.

Definition 4.5.1. Two data points, i.e. i and j, are adjacent (aka similar) if and only if $\mathbf{q}_i^T \mathbf{q}_j \geq \zeta$.

Definition 4.5.2. Two data points, i.e. i and j, are in the same cluster if and only if the index of the maximum value in their corresponding $\mathbf{q}$ vector (i.e. $\mathbf{q}_i$ and $\mathbf{q}_j$) are equal.

Theorem 1. *Consider the ith and jth data points. Then :*

$$\mathbf{q}_i^T \mathbf{q}_j \leq \min \left\{ \max_l \{q_{il}\}, \max_l \{q_{jl}\} \right\} \tag{4.6}$$

where $\mathbf{q}_i^T \mathbf{q}_j$ is the inner product of the two vector $\mathbf{q}_i$ and $\mathbf{q}_j$.

Proof. Assume $\mathbf{q}_j^*$ is a maximal vector that satisfies the below inequality:

$$\mathbf{q}_i^T \mathbf{q}_j \leq \mathbf{q}_i^T \mathbf{q}_j^*, \tag{4.7}$$

where $||\mathbf{q}_j^*||_1 = 1$. In addition, assume the index of the maximum element of $\mathbf{q}_i$ is r:

$$r = \arg \max_l \{q_{il}\} \tag{4.8}$$

In the following, we first prove by contradiction that $\mathbf{q}_j^*$ must be a one-hot vector. Then we prove (4.6).

Assume $\mathbf{q}_j^*$ is not a one-hot vector. Therefore, there exists at least one index, i.e. e, whose corresponding element q_{je}^* is non-zero:

$$\exists\ e : e \neq r \ \text{ and } \ q_{je}^* \neq 0. \tag{4.9}$$

Now, let's define a vector $\hat{q}_j$ as follows:

$$\hat{q}_{jl} = \begin{cases} q_{je}^* + q_{jr}^* & \text{if } \ l = r \\ 0 & \text{if } \ l = e \\ q_{jl}^* & O.W \end{cases}, \tag{4.10}$$

where $\hat{q}_{jl}$ denotes the l^{th} element of $\hat{\mathbf{q}}_j$. Since $||\mathbf{q}_j^*||_1 = 1$, we can immediately show that $||\hat{\mathbf{q}}_j||_1 = 1$. Moreover, we can represent the inner products $\mathbf{q}_i^T\mathbf{q}_j^*$ and $\mathbf{q}_i^T\hat{\mathbf{q}}_j$ as shown in (4.11) and (4.12), respectively.

$$\mathbf{q}_i^T\mathbf{q}_j^* = q_{ir}q_{jr}^* + q_{ie}q_{je}^* + \sum_{l\neq r,e} q_{il}q_{jl}^* \tag{4.11}$$

$$\mathbf{q}_i^T\hat{\mathbf{q}}_j = q_{ir}(q_{jr}^* + q_{je}^*) + q_{ie} \times 0 + \sum_{l\neq r,e} q_{il}q_{jl}^* \tag{4.12}$$

Since q_{ir} is the maximum element of $\mathbf{q}_i$, we can readily show that $\mathbf{q}_i^T\mathbf{q}_j^* \leq \mathbf{q}_i^T\hat{\mathbf{q}}_j$, which contradicts the assumption shown in equation (4.7); thus, $\mathbf{q}_j^*$ must be a one-hot vector. Therefore:

$$\mathbf{q}_i^T\mathbf{q}_j^* \leq \max_l\{q_{il}\} \tag{4.13}$$

Considering (4.13) and (4.7), we have:

$$\mathbf{q}_i^T\mathbf{q}_j \leq \max_l\{q_{il}\}. \tag{4.14}$$

Similarly, for the ith sample, we can show that $\mathbf{q}_i^{*T}\mathbf{q}_j \le \max_l\{q_{jl}\}$, hence:

$$\mathbf{q}_i^T\mathbf{q}_j \le \max_l\{q_{jl}\}. \tag{4.15}$$

(4.14) and (4.15) proves (4.6).

□

Corollary 1.1. *For two adjacent samples, denoted as i and j, the maximum value among their* $\mathbf{q}$ *representations (i.e* $\mathbf{q}_i$ *and* $\mathbf{q}_j$*) are greater than* ζ*.*

Proof. This statement serves as a corollary derived from Theorem 1, which is established through equations (4.6), (4.14), and (4.15), along with the adjacency definition provided in Definition 4.5.1. In this context, we denote the indices of the maximum elements of $\mathbf{q}_i$ and $\mathbf{q}_j$ as r and o respectively, where $r = \arg\max_l\{q_{il}\}$ and $o = \arg\max_l\{q_{jl}\}$.

$$\zeta \le \mathbf{q}_i^T\mathbf{q}_j \le min(q_{ir}, q_{jo}) \quad \rightarrow \quad \begin{cases} \zeta \le q_{ir} \\ \zeta \le q_{jo} \end{cases} \tag{4.16}$$

□

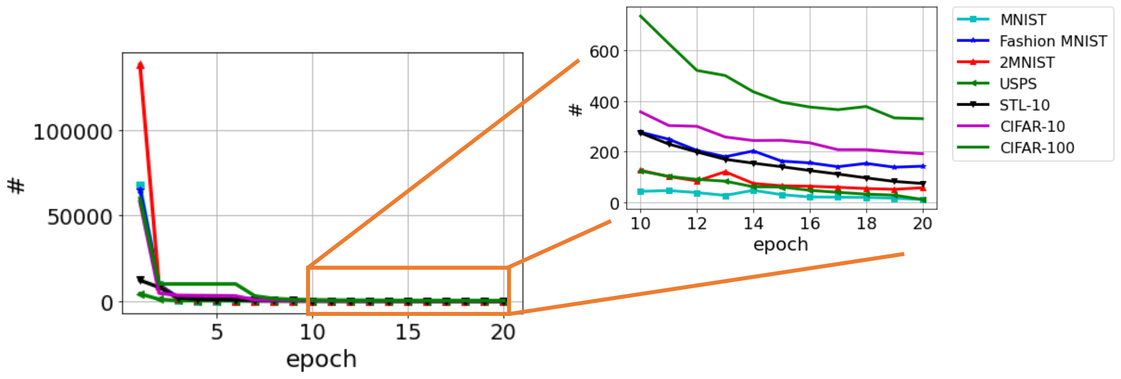


Figure 4.2: Number of data points that do not have any adjacent neighbors during the DCSS training in the second phase.

Corollary 1.2. *If a data point has at least one adjacent neighbor, the maximum element of its corresponding* $\mathbf{q}$ *is greater than* ζ*.*

Proof. We first empirically test the validity of the employed assumption, i.e. the existence of at least one adjacent (i.e. similar) sample for a data point, on our datasets. Fig. 4.2 shows the number of data samples that are not similar to any other data points in the $\mathbf{q}$ space. As it can be seen, at the beginning of the second phase of DCSS, since MNet is initialized randomly, many data points do not have any adjacent neighbor (i.e. almost $\forall\ i, j,\ i \neq j :\ \mathbf{q}_i^T \mathbf{q}_j < \zeta$.). By minimizing (4) and (5) in the second phase of DCSS, similar samples are tightly packed in the $\mathbf{q}$ space; therefore, almost all samples have at least one adjacent neighbor. For example, only 0.5% of the samples in the CIFAR-100 dataset have no adjacent sample by the end of the training phase. Note that CIFAR-100 presents the worst case among the other datasets shown in Fig 4.2. All in all, we can roughly assume that each data point has at least one adjacent sample.

Let us consider an arbitrary data point, i, and one of its adjacent data points, j. From Corollary 1.1, we can conclude that:

$$\begin{cases} \zeta \leq \max_l\{q_{il}\} \\ \zeta \leq \max_l\{q_{jl}\} \end{cases} . \tag{4.17}$$

Thus, we proved that the maximum element of $\mathbf{q}_i$, where i is an arbitrary data point, is greater than ζ. □

Corollary 1.3. *Assume each data point has at least one adjacent neighbor and* $\gamma < \zeta^2$. *If two data points, i and k, are dissimilar, i and k are not from the same cluster.*

Proof. We prove this corollary by contradiction where the contradiction assumption is: i and k are dissimilar, yet from the same cluster where $\gamma < \zeta^2$.

Since i and k are in the same cluster, the index of the maximum element of $\mathbf{q}_i$ and $\mathbf{q}_k$ are the same. (i.e. $\beta = \arg\max_l\{q_{il}\} = \arg\max_l\{q_{kl}\}$). Since each data point has at least one adjacent neighbor, from Corollary 1.2, we can conclude that:

$$\begin{cases} \zeta \leq q_{i\beta} \\ \zeta \leq q_{k\beta} \end{cases} . \tag{4.18}$$

and we can represent $\mathbf{q}_i^T \mathbf{q}_k$ as follow:

$$\mathbf{q}_i^T \mathbf{q}_k = q_{i\beta} q_{k\beta} + \sum_{l \neq \beta} q_{il} q_{kl} \tag{4.19}$$

Therefore, $q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k$. Also, we know i and k are dissimilar. From (4.18), we can conclude that:

$$\zeta^2 \leq q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k \leq \gamma \tag{4.20}$$

(4.20) contradicts the assumption of $\gamma < \zeta^2$. Hence, i and k are in different clusters. □

Theorem 2. *For $\frac{2}{3} \leq \zeta$, if i and j are adjacent, they are in the same cluster – i.e. r is equal to o where $r = \arg\max_l \{q_{il}\}$ and $o = \arg\max_l \{q_{jl}\}$.*

Proof. First, we find an upper bound for $\mathbf{q}_i^T \mathbf{q}_j$, when the ith and jth samples are adjacent but from different clusters.

Since i and j are not from the same cluster, we can represent $\mathbf{q}_i^T\mathbf{q}_j$ as follows:

$$\begin{aligned}
\mathbf{q}_i^T\mathbf{q}_j &= q_{ir}q_{jr} + q_{io}q_{jo} + \sum_{l\neq o,r} q_{il}q_{jl} \\
&\xrightarrow{\forall l:\ q_{jl}\leq q_{jo}} \leq q_{ir}q_{jr} + q_{io}q_{jo} + \sum_{l\neq o,r} q_{il}q_{jo} \\
&= q_{ir}q_{jr} + q_{jo}\Big(\sum_{l\neq r} q_{il}\Big) \\
&\xrightarrow{\sum_{l\neq r} q_{il}=1-q_{ir}} = q_{ir}q_{jr} + q_{jo}(1-q_{ir}) \\
&\xrightarrow{q_{jr}\leq 1-q_{jo}} \leq q_{ir}(1-q_{jo}) + q_{jo}(1-q_{ir}) \\
&\xrightarrow{\text{from (4.16)}} \leq q_{ir}(1-\zeta) + q_{jo}(1-\zeta) \\
&\xrightarrow{\{q_{ir},\, q_{jo}\}\in[0,1]} \leq 2(1-\zeta).
\end{aligned} \tag{4.21}$$

Note that $q_{jr} \leq 1 - q_{jo}$ because $\sum_{l\neq o} q_{jl} + q_{jr} + q_{jo} = 1$. Note that all elements of a $\mathbf{q}$ vector are probability values between 0 and 1.

Hence, as is shown (4.21), if two samples are not from the same cluster, then the inner product of their corresponding $\mathbf{q}$ has an upper bound of $2(1-\zeta)$. Therefore, if two samples i and j are adjacent (see Definition 1) but from different clusters, then:

$$\begin{aligned}
&\zeta \leq \mathbf{q}_i^T\mathbf{q}_j \leq 2(1-\zeta) \\
\rightarrow &\zeta \leq 2(1-\zeta) \rightarrow \zeta \leq \frac{2}{3}.
\end{aligned} \tag{4.22}$$

Thus, for $\frac{2}{3} < \zeta$, i and j cannot be from two different clusters. In other words, if two samples i and j are adjacent AND the user-settable parameter ζ is set to a value greater than $\frac{2}{3}$, then the two samples are from similar clusters, i.e., $r = o$. Based on the experiment is Section 4.6.8, we set $\zeta = 0.8 > \frac{2}{3}$. □

Corollary 2.1. *Assume $\zeta > \frac{2}{3}$. Consider three data points: i, j, and k. If i and j, and also i and k are adjacent, then j and k are from the same cluster.*

Proof. Since i and j (i and k) are adjacent and $\zeta > \frac{2}{3}$, from Theorem 2, we can conclude that i and j (i and k) are in the same cluster; hence, the three samples i, j, and k all are in the same cluster. $\square$

Theorem 3. *Consider three data points i, j, and k where i and j also i and k are adjacent (aka similar). Assume $\zeta > \frac{2}{3}$. If $\gamma < \zeta^2$, then the two samples j and k are not dissimilar (i.e. $\mathbf{q}_j^T\mathbf{q}_k \not< \gamma$).*

Proof. Considering Theorem 2, i and j and k are in the same cluster. Therefore, we do not want to include the pair of j and k samples as a dissimilar pair when minimizing the loss function defined in equation (4.5).

Since j and k are in the same cluster and knowing that the index of the maximum element in a $\mathbf{q}$ vector shows the cluster of the corresponding sample, we have:

$$\eta = \arg\max_l \{q_{jl}\} = \arg\max_l \{q_{kl}\}. \tag{4.23}$$

Thus,

$$\mathbf{q}_j^T\mathbf{q}_k = \sum_{l \neq \eta} q_{jl}q_{kl} + q_{j\eta}q_{k\eta}. \tag{4.24}$$

Therefore,

$$\mathbf{q}_j^T\mathbf{q}_k \geq q_{j\eta}q_{k\eta}. \tag{4.25}$$

Since $\zeta \leq \mathbf{q}_i^T\mathbf{q}_j$ and $\zeta \leq \mathbf{q}_i^T\mathbf{q}_k$, we can infer (4.26) from (4.16).

$$\begin{cases} \zeta \leq q_{j\eta} \\ \zeta \leq q_{k\eta} \end{cases}. \tag{4.26}$$

From (4.25) and (4.26), we can obtain below:

$$\zeta^2 \leq q_{j\eta}q_{k\eta} \leq \mathbf{q}_j^T\mathbf{q}_k. \tag{4.27}$$

Thus, if we choose $\gamma < \zeta^2$, we will not include the pair of samples j and k as a dissimilar pair in equation (4.5). Based on the experiment is Section 4.6.8, γ is set to 0.2, i.e. $\gamma = 0.2 < 0.8^2$. $\square$

4.6 Experiments and Results

In the previous sections of this chapter, we introduced a two-phase DCSS framework. In the first phase, DCSS leverages a novel cluster-specific loss function to train a single autoencoder, enabling the model to reconstruct and center individual clusters. In the second phase, DCSS incorporates an innovative strategy using additional layers and an auxiliary network, MNet, to further refine these representations. To train MNet, we propose a novel loss function that manages pairwise relationships by enhancing the similarity of related data points while reducing that of unrelated ones, thereby producing representations optimized for clustering. In this section, we assess the performance of our proposed DCSS framework and our novel loss functions using eight benchmark datasets: MNIST [228], Fashion MNIST [229], 2MNIST, USPS [242], CIFAR-10 [230], STL-10 [231], CIFAR-100 [230], and ImageNet-10 [243]. To accomplish this, we perform a thorough series of experiments and evaluate performance using two widely used clustering metrics: clustering accuracy (ACC) [232] and normalized mutual information (NMI) [233]. ACC measures the proportion of correctly clustered data points relative to the total, providing a simple yet effective way to gauge the algorithm's clustering accuracy. In contrast, NMI assesses the similarity between two clustering outcomes by quantifying the mutual information shared between their cluster assignments. Both metrics, ACC and NMI, fall within the range of 0 to 1, where higher values indicate better clustering quality. Consistent with established practices in clustering research [234–236], we report the mean and standard deviation of the results from multiple random experiments. Specifically, for DCSS, we carry out 5 experimental runs to ensure a more consistent and reliable evaluation of our results.

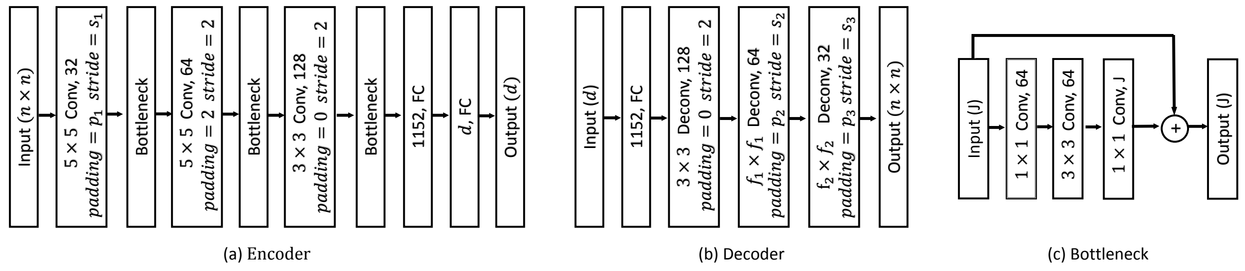

Input ($n \times n$)
5×5 Conv, 32 $padding = p_1$ $stride = s_1$
Bottleneck
5×5 Conv, 64 $padding = 2$ $stride = 2$
Bottleneck
3×3 Conv, 128 $padding = 0$ $stride = 2$
Bottleneck
1152, FC
d, FC
Output (d)
(a) Encoder
Input (d)
1152, FC
3×3 Deconv, 128 $padding = 0$ $stride = 2$
$f_1 \times f_1$ Deconv, 64 $padding = p_2$ $stride = s_2$
$f_2 \times f_2$ Deconv, 32 $padding = p_3$ $stride = s_3$
Output ($n \times n$)
(b) Decoder
Input (J)
1×1 Conv, 64
3×3 Conv, 64
1×1 Conv, J
+
Output (J)
(c) Bottleneck

Figure 4.3: Structure of the proposed asymmetric autoencoder. In the encoder part, in order to obtain an informative lower-dimensional representation of the data points, we propose to use a Bottleneck layer. Following [1], we use the bottleneck layer after 5×5 and 3×3 convolutional layers. The hyperparameter values are presented in Section 4.6.2.

4.6.1 Networks Architecture

The proposed DCSS method includes an autoencoder and a fully connected MNet. This section presents the structure of these networks. We use two variations of autoencoders, depending on the nature of the dataset (i.e. RGB or gray-scale), when training the proposed DCSS framework.

For gray-scale datasets, we propose to use an asymmetric autoencoder, where, following [1], we propose to use the bottleneck layer shown in Fig. 4.3(c) in the encoder structure. Fig. 4.3(a) and (b) respectively show the encoder and decoder structures of the proposed asymmetric AE. Employing such an asymmetric structure provides a more discriminative latent space. Hyperparameters of the proposed AE for each dataset are indicated in Section 4.6.2

For the RGB datasets, we use a simple framework for Contrastive Learning (SimCLR) [21] algorithm as a means of feature extractor. The extracted features are then fed to the DCSS method to perform clustering. In the feature extraction phase using SimCLR, we employ ResNet-34 as the backbone network, and we resize images of all RGB datasets to 224. Also, we consider the output of the ResNet after the average pooling layer as the final extracted features. Then, we feed the extracted features to the DCSS's AE, which is a symmetric, fully connected autoencoder. Inspired by [17], we set the AE architecture to 512-500-500-2000-d for RGB datasets, where the ReLU activation function is utilized in all layers.

MNet is a fully connected network that takes the d dimensional latent space of the AE ($\mathbf{u}$ space) as input and generates a K dimensional output $\mathbf{q}$. The architecture of MNet is d-128-128-128-K

for all datasets except CIFAR-100. Since CIFAR-100 is a more complicated dataset, it needs a more complex MNet architecture, so we set the MNet architecture for CIFAR-100 to d-1000-1000-1000-K. Batch normalization and ReLU activation functions are utilized for all datasets in all layers of MNet except the last layer in which we use the soft-max function.

4.6.2 Implementation Details

In this section, we discuss hyperparameter values and implementation details of DCSS. Hyperparameters n, p_1, s_1, p_2, s_2, p_3, s_3, f_1, and f_2 (shown in Fig. 4.3) are respectively set to 28, 2, 2, 1, 2, 2, 2, 5, and 4 for MNIST, Fashion MNIST, and 2MNIST; these parameters are set to 16, 1, 1, 2, 2, 0, 1, 4, and 5 for the USPS dataset. The latent space dimension d is set to 10 for gray-scale images and 20 for RGB images. Following [17–19, 25], in order to initialize $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$ and $\boldsymbol{\mu}^{(k)}$ for $k = 1, \dots, K$, we train an autoencoder where the end-to-end training is performed by only minimizing the samples reconstruction losses. Adam optimization method [244], with the same parameters mentioned in the original paper are used for training. $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$ are then initialized with the parameters of the trained autoencoder's parameters. We apply k-means algorithm [2] to the latent space of the trained autoencoder and initialize $\boldsymbol{\mu}^{(k)}, k = 1, \dots, K$ to the centers defined by k-means. For all datasets, hyperparameters α, Maxiter$_1$, T$_1$, and m are respectively set to 0.1, 200, 2, and 1.5. The second phase hyperparameters ζ, γ, T$_2$, and MaxIter$_2$ are respectively set to 0.8, 0.2, 5, and 20. We utilize the Adam optimizer for updating weights of the AE and MNet and their learning rates are set to 10^{-5} and 10^{-3}, respectively. All algorithms were implemented in Python using the PyTorch framework.

4.6.3 Clustering Performance

The effectiveness of our proposed DCSS method is compared against seventeen well-known algorithms, including conventional and state-of-the-art deep-learning-based clustering methods. The conventional clustering methods are k-means [2], large-scale spectral clustering (LSSC) [237], and locality preserving non-negative matrix factorization (LPMF) [238]. Deeplearning-based al-

Table 4.1: ACC and NMI on the benchmark datasets for different clustering methods.

Method \ Datasets	MNIST		Fashion MNIST		2MNIST		USPS		CIFAR-10		STL-10		CIFAR-100		ImageNet-10	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means	53.20	50.00	47.40	51.20	32.31	44.00	65.67	62.00	22.90	8.70	19.20	12.50	13.00	8.40	24.10	11.90
LSSC	71.40	70.60	49.60	49.70	39.77	51.22	63.14	58.94	21.14	10.89	18.75	11.68	14.60	7.92	-	-
LPMF	47.10	45.20	43.40	42.50	34.68	38.69	60.82	54.47	19.10	8.10	18.00	9.60	11.80	7.90	-	-
DEC	84.30	83.72	51.80	54.63	41.20	53.12	75.81	76.91	30.10	25.70	35.90	27.60	18.50	13.60	38.10	28.20
IDEC	88.13	83.81	52.90	55.70	40.42	53.56	75.86	77.68	36.99	32.53	32.53	18.85	19.61	14.58	39.40	29.00
DCN	83.00	81.00	51.22	55.47	41.35	46.89	73.00	71.90	30.47	24.58	33.84	24.12	20.17	12.54	37.40	27.30
DKM	84.00	81.54	51.31	55.57	41.75	46.58	75.70	77.60	35.26	26.12	32.61	29.12	18.14	12.30	38.10	29.80
VaDE	94.50	87.60	50.39	59.63	56.60	51.20	40.35	58.37	29.10	24.50	28.10	20.00	15.20	10.80	33.40	19.30
DAC	97.75*	93.51*	62.80	58.90	-	-	-	-	52.18	39.59	46.99	36.56	23.75	18.52	52.70	39.40
GANMM	64.00	61.00	34.00	27.00	-	-	50.12	49.35	-	-	-	-	-	-	-	-
CC	88.56	84.21	64.52	61.45	42.15	58.89	81.21	79.45	77.00	67.80	85.00	76.40	42.30	42.10	89.30	85.00
PICA	-	-	-	-	-	-	-	-	69.60	59.10	71.30	61.10	33.70	31.00	87.00	80.20
EDESC	91.30	86.20	63.10*	67.00*	-	-	-	-	62.70	46.40	74.50	68.70	38.50	37.00	-	-
IDFD	-	-	-	-	-	-	-	-	81.50	71.10	75.60	64.30	42.50	42.60	95.40*	89.80*
MICE	-	-	-	-	-	-	-	-	83.50*	73.70*	75.20	63.50	44.00	43.60	-	-
DCCM	-	-	-	-	-	-	68.60	67.50	62.30	49.60	48.20	37.60	32.70	28.50	71.00	60.80
AE + k-means	86.03	80.25	57.94	57.15	44.01	62.80	75.11	74.45	80.23	68.65	85.29	76.00	43.81	42.86	87.23	85.22
DCSS$_u$	95.99	89.95	62.90	63.58	45.31*	63.00*	82.93*	81.84*	82.43	71.32	86.47*	76.52*	44.08*	43.70*	89.46	86.30
DCSS	**97.87** ± 0.31	**94.61** ± 0.24	**66.32** ± 0.20	**67.25** ± 0.28	**48.68** ± 0.29	**67.70** ± 0.37	**87.08** ± 0.19	**86.26** ± 0.25	**84.24** ± 0.32	**75.00** ± 0.24	**87.95** ± 0.16	**77.28** ± 0.22	**45.03** ± 0.26	**44.41** ± 0.23	**95.55** ± 0.17	**90.58** ± 0.27

gorithms are deep embedding clustering (DEC) [17], improved deep embedding clustering (IDEC) [18], deep clustering network (DCN) [25], deep k-means (DKM) [19], variational deep embedding (VaDE) [239], GAN mixture model for clustering (GANMM) [168], deep adaptive clustering (DAC) [30], and the very recent clustering methods such as contrastive clustering (CC) [23], deep semantic clustering by partition confidence maximization (PICA) [178], efficient deep embedded subspace clustering (EDESC) [245], instance discrimination and feature decorrelation (IDFD) [246], Mixture of contrastive experts for unsupervised image clustering (MICE) [247], and deep comprehensive correlation mining (DCCM) [31]. In addition, we report the clustering performance of a baseline method AE + k-means in which k-means is simply applied to the latent representation of an AE that has a similar architecture as the AE used in the DCSS method, trained based on minimizing the dataset reconstruction loss.

We also demonstrate the success of the first phase of DCSS, presented in Section 4.2, in creating the reliable subspace $\mathbf{u}$ in which the data points form hypersphere-like clusters around their corresponding cluster center. To this end, we only implement the first phase of the DCSS algorithm – i.e., we train the DCSS's AE through minimizing the loss function presented in (4.1), where the AE architecture and its initialization are similar to those presented in Section 1 of the supplementary material file. After training the $\mathbf{u}$ space, we perform a crisp cluster assignment by considering each data hypersphere-like group in the $\mathbf{u}$ space as a data cluster and assigning each data point to the one with the closest center. In the following tables and figures, clustering using only the first phase is shown as DCSS$_u$. A preliminary version of DCSS$_u$ is presented in [240].

The clustering performance of DCSS$_u$ and DCSS, along with the comparison algorithms, are shown in Table 4.1. For the comparison methods, if the ACC and NMI of a dataset are not reported in the corresponding original paper, we ran the released code with the same hyper-parameters discussed in the original paper. When the code is not publicly available or not applicable to the dataset, we put dash marks (-) instead of the corresponding results. The best result for each dataset is shown in bold. The second top results are shown with *. Also. the standard deviation over 5 experiments is reported in this table.

Several observations can be made from Table 4.1: (1) The proposed DCSS method outperforms all of our comparison methods on all datasets. (2) The first phase of DCSS (shown as DCSS$_u$) effectively groups the data points around their corresponding centers. This can be inferred from DCSS$_u$'s ACC and NMI values. When measuring ACC, DCSS$_u$ outperforms other methods in five out of eight datasets, and it exhibits even stronger performance by outperforming competitors in six out of eight datasets when considering NMI. (3) Effectiveness of the self-supervision with similar and dissimilar pairs of samples can be inferred by comparing DCSS with DCSS$_u$. It can be seen that DCSS significantly outperforms DCSS$_u$ on all datasets. (4) Effectiveness of the AE's loss function proposed in equation (4.1) compared to the case of training AE with only the reconstruction loss can be inferred by comparing the DCSS$_u$ performance with the baseline method AE+k-means. As can be seen, the DCSS$_u$ clearly outperforms AE+k-means on all datasets.

4.6.4 t-SNE visualization

Fig. 4.4 illustrates the effectiveness of different phases of our proposed DCSS framework for all datasets, where t-SNE [241] is used to map the output of DCSS's encoder and MNet to a 2D space. The different colors correspond to the different data clusters.

The first row of Fig. 4.4 shows the representation of different data points in the $\mathbf{u}$ space, i.e. the latent space of the DCSS's AE, only after completing the first phase discussed in Section 4.2. As it can be seen, after completing the first phase of DCSS, different clusters of data points are fairly separated, sit near their corresponding centers, and form spheres; however, not all clusters are well separated. For example, in the USPS dataset, the data clusters shown in pink, purple, and magenta are mixed together. This indicates the insufficiency of the reconstruction and centering losses for the clustering task.

The second row of Fig. 4.4 shows the data representations in the $\mathbf{u}$ space after completing the second phase of DCSS discussed in Section 4.3, where $\mathbf{u}$ is refined by minimizing (4.4) and (4.5). As it can be seen, refining the $\mathbf{u}$ space employing pairwise similarities results in more dense and separate cluster distributions. For example, the pink, purple, and magenta clusters of USPS are now well distinguishable in the new refined $\mathbf{u}$ space. As another example, see samples of the three

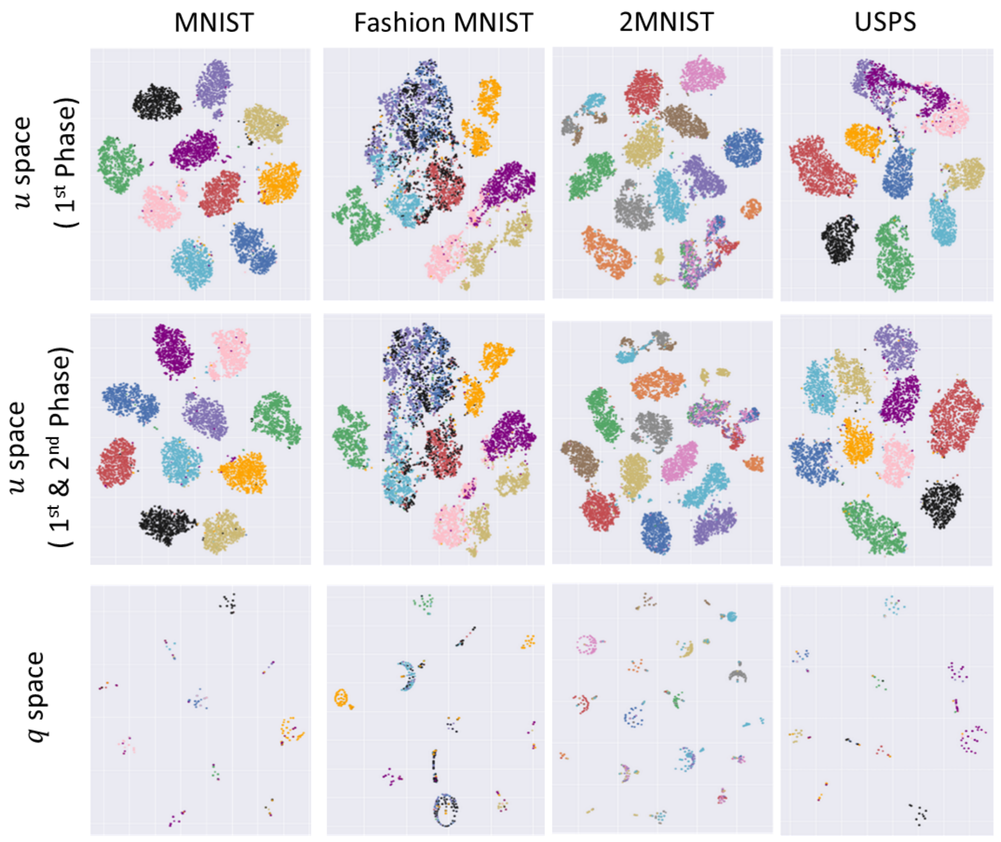


Figure 4.4: Clustering visualization of different phases of DCSS using t-SNE for different benchmark datasets. For reference, the visualization for the baseline model AE+k-means is shown in the first row. Axes range from -100 to 100.

clusters shown in red, olive, and brown of the 2MNIST dataset. These clusters are more separable in the refined $\mathbf{u}$ space compared to the corresponding representation shown in the first row.

The last row in Fig. 4.4 depicts the output space of MNet (i.e. the $\mathbf{q}$ space), in which we make decisions about final cluster assignments of data points. As is expected, clusters in this space have low within- and high between-cluster distances, and cluster distributions can take non-hypersphere patterns. As an example, consider the cyan and the purple clusters of the Fashion MNIST. These clusters are mixed in the $\mathbf{u}$ space, but they are completely isolated in the $\mathbf{q}$ space.

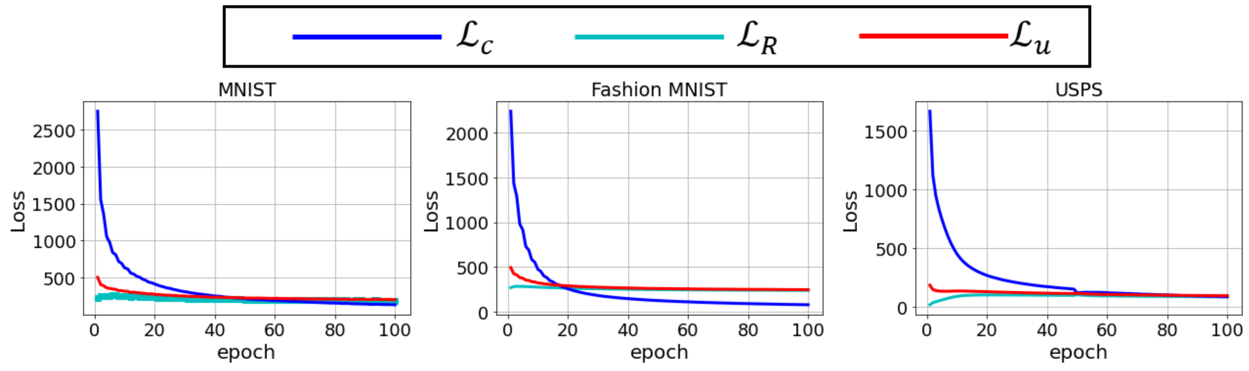


Figure 4.5: The reconstruction loss $\mathcal{L}_r$, centering loss $\mathcal{L}_c$, and total loss $\mathcal{L}_u$ of the first phase of DCSS vs. training epochs, for different datasets.

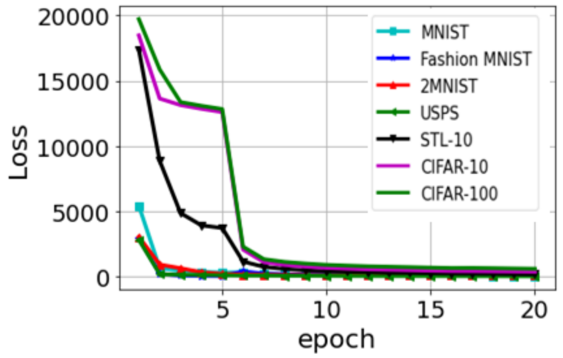


Figure 4.6: The second phase's loss function of DCSS for different benchmark datasets.

4.6.5 Loss function convergence

Fig. 4.5. depicts the average, over different clusters on different batches of data points, of the reconstruction, centering, and total losses corresponding to the first phase of DCSS (i.e. DCSS$_u$) shown in (4.1). As can be seen, all losses are converged at the end of training. The noticeable reduction in the centering loss shows the effectiveness of our proposed approach in creating a reliable $\mathbf{u}$ space in which the data points are gathered around the centers. Moreover, the figures show that at the first training epochs, our method trades the reconstruction loss for improved centering performance. This proves the insufficiency of the reconstruction loss in creating a reliable latent space for data clustering.

In Fig. 4.6, we investigate the convergence of the second phase losses, shown in equations (4.4) and (4.5). Since we initialize the MNet randomly, at the first few epochs, MNet has little knowledge about the lower-dimension representation of the data points in the $\mathbf{q}$ space; thus, we face a high loss value. As the training process progresses, the loss value drops and converges to zero at the end of the training process. In the first T_2 epochs ($T_2 = 5$), the algorithm minimizes the loss presented in (4). It minimizes (5) in the remaining epochs. The continuity of the loss reduction over epochs, along with the sharp loss drop at the 5th epoch, confirms the effectiveness of our proposed strategy in employing $\mathbf{u}$ for similarity measurements in the early epochs and then $\mathbf{q}$ in the later epochs.

4.6.6 Performance on Imbalanced Dataset

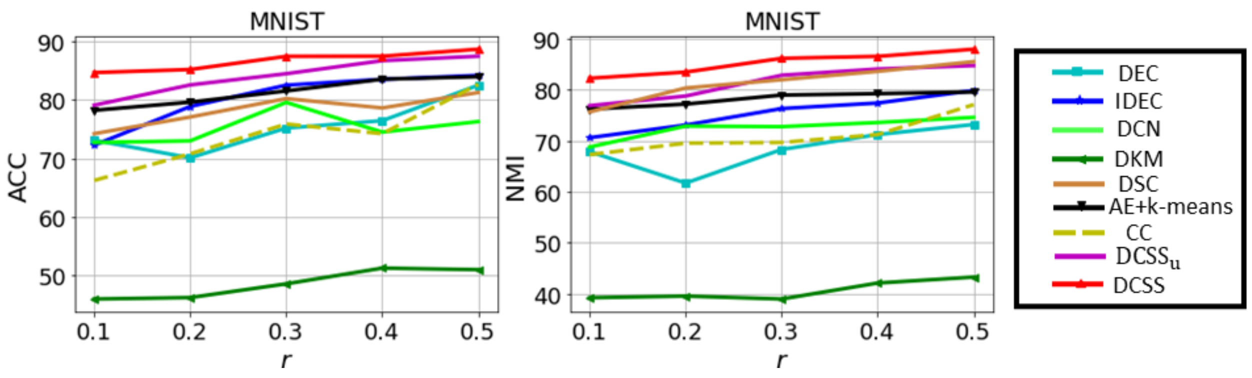
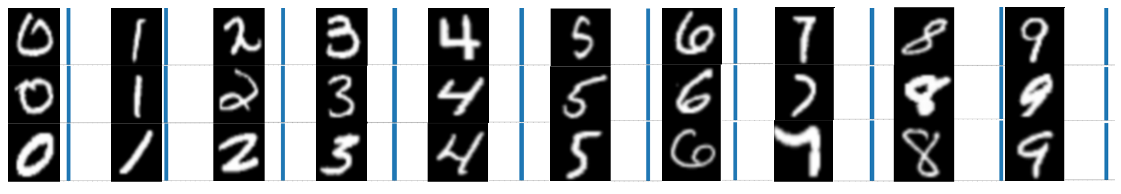


Figure 4.7: The clustering performance of different methods on imbalanced samples of MNIST.

To demonstrate the effectiveness of our proposed DCSS method on an imbalanced dataset, we randomly collect five subsets of the MNIST dataset with different retention rates $r \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, where samples of the first class are chosen with the probability of r and the last class with probability of 1, with the other classes linearly in between. Hence, on average, the number of samples for the first cluster is r times less than that of the last cluster. As is shown in Fig. 4.7, our proposed DCSS framework significantly outperforms our comparison methods for all r values. This indicates the robustness of DCSS on imbalanced data. As is expected, in general, for all methods, increasing r results in a higher performance because the dataset gets closer to a balanced one. Higher performance of DCSS on imbalanced datasets can be associated with two factors: (1) considering an individual loss for every cluster in the 1st phase, and (2) considering the pairwise data relations.

4.6.7 Visualization of q vectors


(a) MNIST

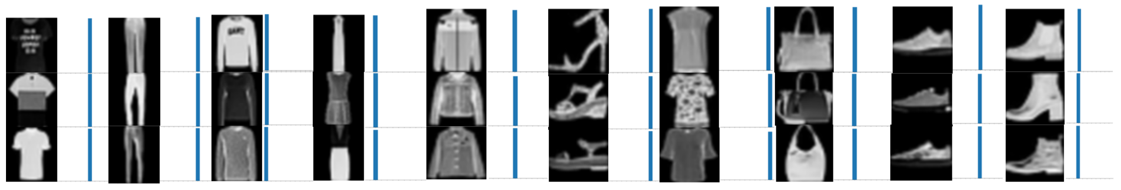

(b) Fashion MNIST

Figure 4.8: Visualization of $\mathbf{q}$ for samples from (a) MNIST and (b) Fashion MNIST datasets. The $\mathbf{q}$ vector for each image is depicted beside the image. The vertical axes range from 0 to 1.

Fig. 4.8 shows the representations of the data points from various clusters in the $\mathbf{q}$ space. As can be seen, the proposed DCSS method results in representations that are very close to the one-hot vectors. Note that the kth element of $\mathbf{q}_i$ denotes the probability of sample $\mathbf{x}_i$ being in the kth data cluster. The closer $\mathbf{q}_i$ is to the one-hot vector, the more confidently a crisp cluster assignment can be made. As is proved in Corollary 1.2 of Section 4.5, if data point $\mathbf{x}_i$ has at least one similar neighbor, the maximum element of $\mathbf{q}_i$ is greater than ζ. In our experiments, ζ is set to 0.8. This can justify the aggregation of the data points near the one-hot vectors in the $\mathbf{q}$ space.

To further demonstrate convergence of the $\mathbf{q}$ representations to one-hot vectors, histogram of residuals $h_i = ||\mathbf{I}_i - \mathbf{q}_i||_1$, $i = 1, \ldots, N$ for MNIST, Fashion MNIST, 2MNIST, and USPS are shown in Fig. 4.9; where $||.||_1$ indicates the ℓ_1-norm and $\mathbf{I}_i$ is the one-hot crisp assignment corresponding to $\mathbf{q}_i$ – i.e. the index of the non-zero element of $\mathbf{I}_i$ is equal to the index of the maximum element of $\mathbf{q}_i$. As can be seen in Fig. 4.9, the representation of almost all data points in the $\mathbf{q}$ space is very close to their corresponding one-hot vector.

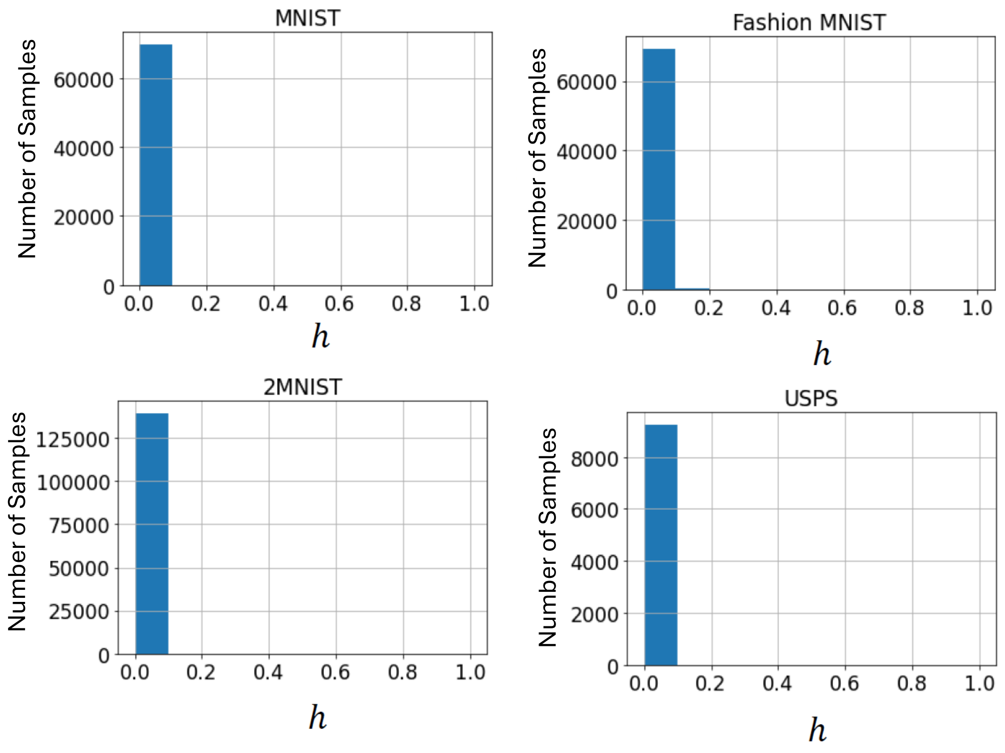


Figure 4.9: Histogram plot of $h_i = ||\mathbf{I}_i - \mathbf{q}_i||_1$, $i = 1, \ldots, N$ where $\mathbf{I}_i$ is the one-hot crisp assignment corresponding to $\mathbf{q}_i$.

4.6.8 Ablation Studies

Effectiveness of the First Phase: To demonstrate the efficacy of introducing cluster-specific losses and the adopted approach in iteratively updating the network's parameters over K successive runs, as is discussed in Section 4.2, we present a comparative analysis against an alternative approach where all loss terms are aggregated in the initial phase, followed by a single backward pass to update the network parameters at once. The results are presented in Table 4.2 as DCSS$_{agg}$, highlighting a consistent trend. Across all experiments and datasets, the performance of DCSS$_u$ surpasses that of DCSS$_{agg}$ showing the advantages of employing cluster-specific loss functions and iteratively updating the network parameters in K successive runs. To ensure a fair comparison, in all experiments, the number of iterations of DCSS$_{agg}$ is K times greater than the number of iterations in DCSS$_u$.

Hyperparameters Sensitivity: In Fig. 4.10, we investigate the effect of different hyperparameters

Table 4.2: ACC and NMI on the benchmark datasets for different methods. We report the average of 5 random runs.

Datasets / Method	MNIST		Fashion MNIST		USPS		CIFAR-10		STL-10		CIFAR-100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DCSS$_{agg}$	87.22	81.39	59.76	58.63	76.46	75.09	81.41	70.64	85.51	76.19	43.52	43.02
DCSS$_u$	95.99	89.95	62.90	63.58	82.93	81.84	82.43	71.32	86.47	76.52	44.08	43.70

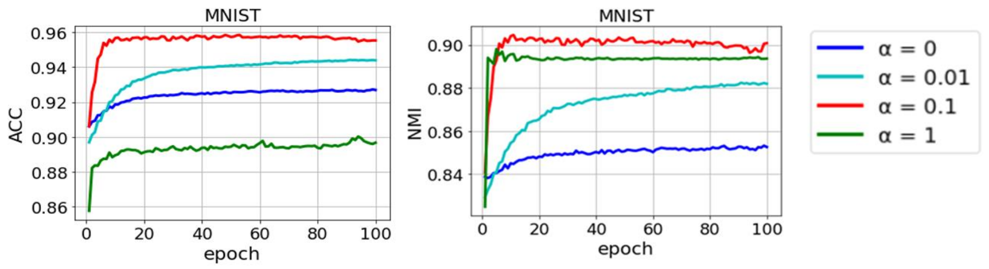
on DCSS clustering performance. For hyperparameters of the first phase (i.e. α, m, and T_1), we report the performance of clustering using DCSS$_u$ (as is presented in Section 4.6.3).

In our proposed method, hyperparameters are fixed across all datasets, i.e. no fine-tuning is performed per dataset. Hence, one may obtain more accurate results by tuning the hyperparameters per dataset.

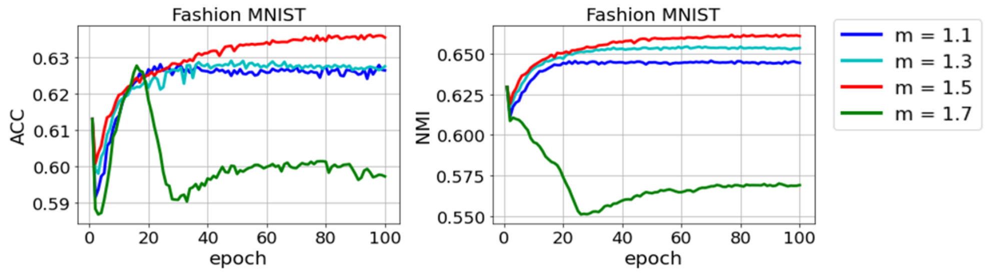
In Fig. 4.10 (a), we explore the importance of the centering loss in the first phase's loss function, shown in (4.1), by changing $\alpha \in \{0, 0.01, 0.1, 1\}$ for MNIST dataset. As is shown in this figure, by increasing the value of α from 0 to 0.1, our DCSS performance significantly enhanced in terms of ACC and NMI, which demonstrates the effectiveness of incorporating the centering loss beside the reconstruction loss in the first phase's loss function. We observed a similar trend across all the other datasets. In all our experiments, for all datasets, α is set to 0.1.

Fig. 4.10 (b) shows the impact of the level of fuzziness m on the clustering performance of DCSS$_u$ for the Fashion MNIST dataset, where $m \in \{1.1, 1.3, 1.5, 1.7\}$. In the case of m$\rightarrow$ 1 (m$\rightarrow \infty$), group membership vectors converge to one-hot (equal probability) vectors. As shown in this figure, as desired, the DCSS method is not too sensitive to m when it is set to a reasonable value. We observed a consistent pattern across all the remaining datasets. In all our experiments, for all datasets, m is set to 1.5.

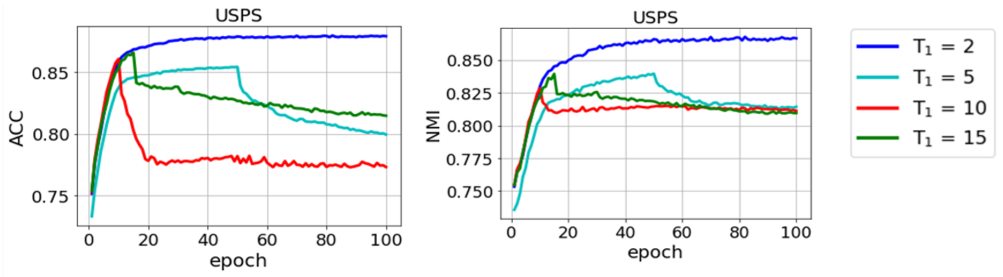
In Fig. 4.10 (c), we scrutinize the effect of update interval T_1 in the clustering performance of the first phase for $T_1 \in \{2, 5, 10, 15\}$. As is expected, better clustering performance in terms of ACC and NMI is acquired for a smaller value of T_1 for the USPS dataset. Consistently, the same behavior was observed across all remaining datasets. In our experiments, for all datasets, T_1 is set to 2.



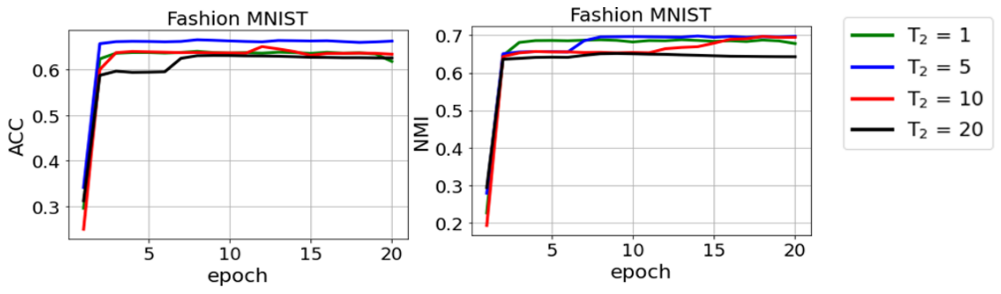
(a) Effect of α on clustering performance



(b) Effect of m on clustering performance



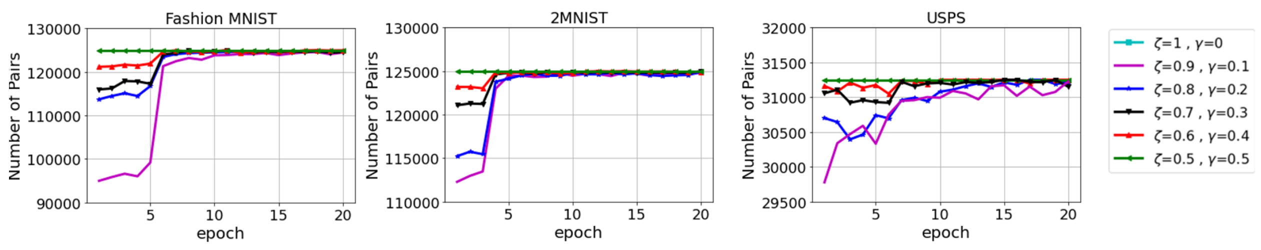
(c) Effect of T_1 on clustering performance



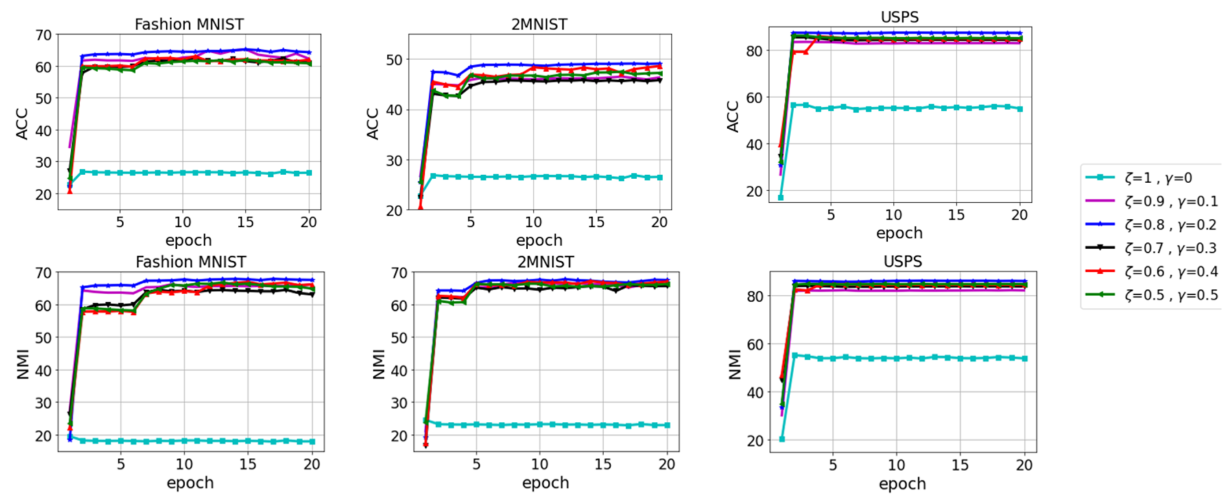
(d) Effect of T_2 on clustering performance

Figure 4.10: Sensitivity of DCSS to different hyperparameters.

In Fig. 4.10(d), we change the number of training epochs T_2, defined in Section 4.3, for Fashion MNIST dataset, where $T_2 \in \{1, 5, 10, 20\}$. As is expected, for a very small T_2 value, e.g. $T_2 = 1$, where training the $\mathbf{q}$ space is mainly supervised by the $\mathbf{q}$ space itself even at the MNet training



(a) Number of data samples participating in the second phase of DCSS



(b) ACC and NMI for different γ and ζ

Figure 4.11: Changing hyperparameters ζ and γ for different datasets. (a) Number of data pairs participating in the second phase of DCSS. (b) Clustering performance in terms of ACC and NMI for different datasets for different values of ζ and γ; the clustering performance of DCSS is less sensitive to the choice of ζ and γ in the range of [0.5,0.9] and [0.1,0.5], respectively.

outset, DCSS cannot provide a proper $\mathbf{q}$ space, since $\mathbf{q}$ is not a sufficiently reliable space to be used for self-supervision. The figure also shows that for a very large T_2 value, e.g. $T_2 = 20$, when we only trust the $\mathbf{u}$ space for supervising the $\mathbf{q}$ space, we cannot train an effective $\mathbf{q}$ space. As shown, a good clustering performance can be obtained when T_2 is set to a moderate value. In our experiments, for all datasets, T_2 is set to 5. This demonstrates the effectiveness of the proposed strategy in supervising the MNet training using both the $\mathbf{u}$ and $\mathbf{q}$ spaces.

In Fig. 4.11, we change ζ and γ in range [0,1], where $\zeta + \gamma = 1$, to observe model convergence and accuracy for different lengths of the ambiguity interval, defined as $\zeta - \gamma$, ranging from 1 (when $\zeta = 1$) to 0 (when $\zeta = 0.5$). Fig. 4.11(a) shows the number of pairs participating in minimizing the loss functions defined in (4.4) and (4.5). As can be seen, at the beginning of the second phase, our model can make a decisive decision only about a few pairs, and the remaining pairs are in the ambiguous region. As the second phase of the training process progresses, more and more pairs are included in the loss functions optimization process. Finally, at the end of the second phase, almost all pairs contribute to the training.

Furthermore, in Fig. 4.11(b), we investigate the influence of ζ and γ in clustering performance. As it can be seen, as is desired, the final clustering performance of our DCSS framework is not highly sensitive to the choice of ζ and γ when these are set to reasonable values. In all our experiments, $\zeta = 0.8$ and $\gamma = 0.2$ for all experiments and datasets.

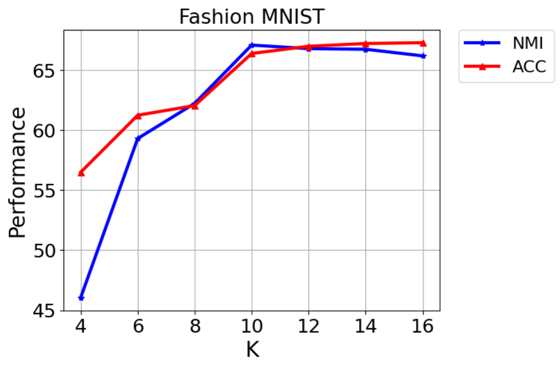


Figure 4.12: Sensitivity of DCSS model to the initial number of clusters (K).

In Figure 4.12, we investigate the impact of initializing the number of clusters to values smaller or larger than the actual number of clusters on the performance of our DCSS model using the Fashion MNIST dataset, where the true number of clusters is 10. When the initial number of clusters (e.g., K = 4) is smaller than the actual number, the model is forced to group data samples from multiple clusters into a single cluster, resulting in a decrease in the DCSS model's performance in terms of ACC and NMI. Conversely, with higher values of K (e.g., K = 16), the algorithm may tend to over-segment the data, creating smaller clusters that may not well align with the true structure of the data. In an extreme scenario where K is set to a very large value, the over-segmenting of the data may cause overfitting to the training data. Employing a validation set becomes valuable when conducting a parameter sweep to identify an appropriate value for K.

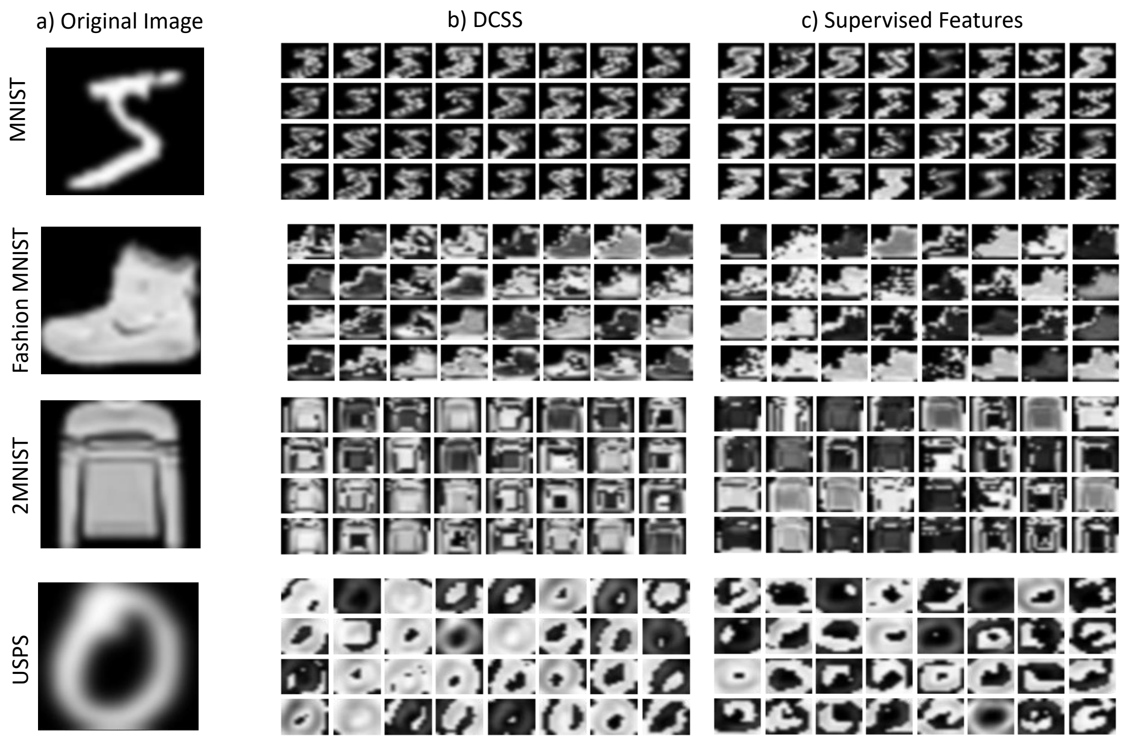


Figure 4.13: (a) samples of MNIST, Fashion MNIST, 2MNIST, and USPS. The output of the first convolutional layer using (b) the unsupervised DCSS method, (c) a supervised manner employing the same network structure.

4.6.9 Features visualization

In order to investigate the effectiveness of our model in extracting useful features for different datasets, we train a deep neural network with the same structure as is presented in Section 1 of the supplementary material file in a *supervised* manner, and then we compare the output of the first convolutional layers for the trained model and our proposed DCSS model. As it can be seen in Fig. 4.13, our DCSS learns a variety of low- and high-frequency features, which are similar to features learned in a supervised manner. This demonstrates the effectiveness of our framework in finding informative features in an unsupervised manner.

4.7 Limitation of DCSS

In this chapter, we introduced the DCSS framework, a novel two-phase approach designed to enhance clustering performance. The framework operates as follows: in the first phase, it focuses on defining cluster-specific loss functions to train a single AE. In the second phase, it determines clusters by leveraging pairwise relationships between data samples. While the DCSS framework demonstrates remarkable performance on challenging datasets, it is not without its limitations. A significant limitation of the DCSS framework is its reliance on a single network for training, which constrains its ability to capture cluster-specific characteristics when the number of clusters is large. This issue becomes especially pronounced when the network's capacity remains constant while the complexity of the clustering task increases. For instance, as highlighted in Table 4.1, DCSS exhibits lower performance on the CIFAR-100 dataset compared to others. This is because CIFAR-100's large number of clusters makes it challenging for DCSS to extract and encode cluster-specific patterns effectively. To address this limitation, employing networks with greater capacity could help capture the intricate patterns across numerous clusters. However, this solution comes with trade-offs: larger networks demand more memory and significantly increase training time, potentially reducing the framework's practicality for large-scale or resource-constrained applications.

4.8 Summary

In this chapter, we introduced a novel deep clustering framework, DCSS, which leverages the concept of self-supervision to improve clustering performance. The proposed framework is designed as a two-phase approach, each contributing uniquely to the clustering process. In the first phase, we employed an autoencoder trained with cluster-specific losses to form hypersphere-like groups of similar data points in its latent space. Each cluster is represented as a hypersphere, enabling the model to capture fundamental clustering structures effectively. This latent space representation serves as the foundation for the subsequent phase. In the second phase, pairwise data similarities were utilized to project the data into a K-dimensional space, where K represents the number of clusters. This additional transformation enables the framework to accommodate more complex cluster distributions, improving the granularity and accuracy of the clustering process. By integrating pairwise relationships, this phase enhances the clustering performance by providing a richer representation of the underlying data structure. The effectiveness of the DCSS framework was rigorously evaluated across eight benchmark datasets, demonstrating its ability to outperform traditional and state-of-the-art clustering algorithms. Our experiments highlight the complementary nature of the two phases, with the latent representations generated in the first phase significantly contributing to the success of the second phase. Together, these phases offer a cohesive and powerful approach to tackling challenging clustering tasks.

Chapter 5

Cross-instance guided Contrastive Clustering (C3)

In this chapter, we present a novel framework, Cross-instance Guided Contrastive Clustering (C3), which addresses key limitations in traditional contrastive clustering approaches by leveraging the power of cross-instance relationships. Unlike conventional methods that predominantly rely on data augmentation, C3 delves deeper into the semantic connections among instances within the dataset. By analyzing instance-level representations in the feature space, it identifies semantically similar samples, enabling the formation of more meaningful positive pairs. This approach captures richer, more nuanced patterns in the data, resulting in significantly enhanced clustering performance. A distinguishing feature of C3 is its sophisticated weighting mechanism for refining negative sample selection. In many contrastive learning paradigms, the choice of negative pairs can introduce errors, particularly in the presence of false negatives (semantically similar instances incorrectly treated as dissimilar), noisy data, or anomalies. C3 tackles this challenge with an intelligent weighting strategy that minimizes the impact of such detrimental factors. By carefully selecting negative pairs that contribute constructively to the learning process, C3 avoids pitfalls that could compromise model performance, ensuring the integrity and robustness of the clustering outcomes. To validate the effectiveness of the proposed framework, we conducted extensive experiments on a variety of datasets, spanning different domains and complexities. The results, discussed in detail in the subsequent sections, demonstrate that C3 consistently outperforms state-of-the-art methods across multiple metrics. These findings underscore the framework's ability to produce robust and high-quality clustering outcomes, further affirming its potential for practical applications.

5.1 Notation

Like other contrastive learning methods, we apply two data augmentations $\mathcal{T}^a$ and $\mathcal{T}^b$, sampled randomly from a pool of transformations, $\mathcal{T}$, to form a *true positive pair* (x_i^a, x_i^b) for a sample x_i, where $x_i^a = \mathcal{T}^a(x_i)$ and $x_i^b = \mathcal{T}^b(x_i)$. In this chapter, we used SimCLR transformation pool [21]. As is shown in Fig. 5.1. We use the encoder network $f(.)$ to extract features of augmented samples, i.e, $h_i^a = f(x_i^a)$ and $h_i^b = f(x_i^b)$. Inspired by CC [23], we devise instance-level and cluster-level contrastive networks, denoted by $g_I(.)$ and $g_C(.)$, respectively. The instance-level network maps the extracted feature of augmented samples to the latent representation (aka z-space), i.e. $z_i^a = g_I(h_i^a)$ and $z_i^b = g_I(h_i^b)$. The output of the cluster-level network is the cluster assignments of samples to different clusters, i.e. $c_i^a = g_C(h_i^a)$ and $c_i^b = g_C(h_i^b)$. We call the output of the cluster-level network c-space. We first initialize our networks, i.e. $f(.)$, $g_I(.)$, and $g_C(.)$ using the CC algorithm. If we do such initialization process for a sufficient number of epochs, a partially reliable z-space will be obtained. However, the z-space obtained by minimizing the CC loss is sub-optimal for clustering, which results in a large false-negative-pair rate and a low true-positive-pair rate. To mitigate this issue, our method incorporates cross-sample similarities.

5.2 Proposed C3 Loss

Since our framework is unsupervised and we do not have access to the data labels to identify samples belonging to the same cluster, we use a notion of self-supervision to refine clusters. We employ the cross-sample similarities in the partially trained z-space to realize the self-supervision concept. The cross-sample similarity is measured by the cosine distance of samples in the z-space.

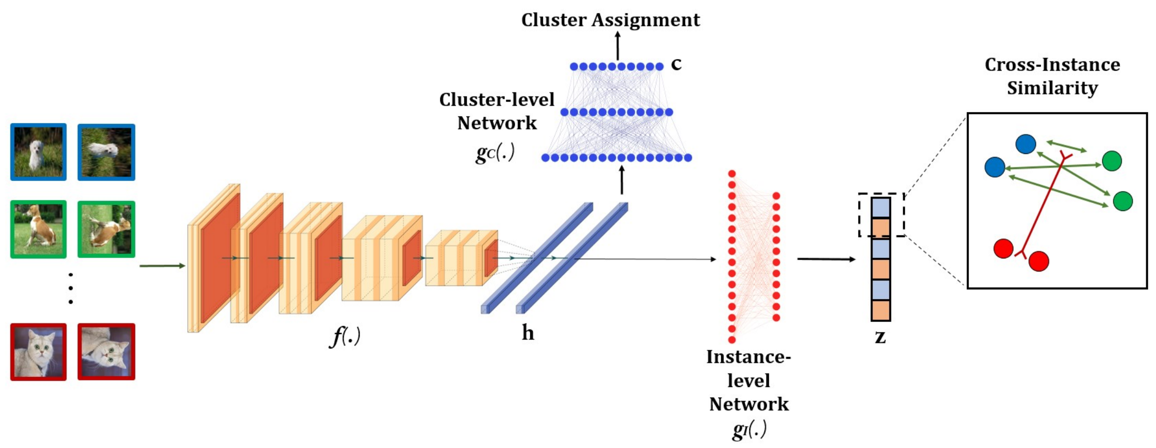


Figure 5.1: An overview of the training phase of our proposed C3 method.

If, for a pair of instances, the similarity is greater than or equal to a threshold ζ, we consider those samples to be similar and pull them closer together by minimizing the loss function $\mathcal{L}_{C3}$ defined below:

$$\mathcal{L}_{C3} = \frac{1}{2N} \sum_{i=1}^{N} (\tilde{\ell}_i^a + \tilde{\ell}_i^b) \tag{5.1}$$

$$\tilde{\ell}_i^a = -\log \frac{\sum_{k \in \{a,b\}} \sum_{j=1}^{N} \mathbb{1}\{\mathrm{z_i^{a\intercal} z_j^k} \geq \zeta\} \exp(\mathrm{z_i^{a\intercal} z_j^k})}{\sum_{k \in \{a,b\}} \sum_{j=1}^{N} \mathrm{w_{ij}^k} \exp(\mathrm{z_i^{a\intercal} z_j^k})}, \tag{5.2}$$

where $\mathbb{1}\{.\}$ denotes the indicator function. Analogous to $\tilde{\ell}_i^a$, we define $\tilde{\ell}_i^b$ that considers similarity of z_i^b and other samples in the batch. Furthermore, in the denominator of the proposed loss function, we included w_{ij}^k to consider higher weights for the samples that are neither close together nor far from each other. In this way, we realize the goal of decreasing the false negative pair selection rate as, in the traditional CC-based methods, all augmented samples in the batch are equally considered when forming negative pairs, regardless of the possibility of them belonging to the same cluster and the difficulty of the samples to cluster.

We assume that the weight terms are given when minimizing the C3 loss defined in (5.2). To obtain an optimum value for a weight term w_{ij}^k, we propose solving the below optimization problem while the networks are frozen. The first term of the below optimization problem is defined based on our motivation to assign a very low weight to too close and too far away samples and, instead, let the remaining samples, which are more probably located on the cluster boundaries, take higher weights. The second term is to avoid the trivial solution of assigning a weight equal to one to the sample providing the maximum value of $1 - |z_i^{a\intercal} z_j^k|$. To avoid instability, we include the constraint by which the summation of all weights must be equal to 1.

$$\min_{w_{ij}^k} \sum_{k\in\{a,b\}} \sum_{j=1}^{N} -w_{ij}^k (1 - |z_i^{a\intercal} z_j^k|) - \frac{1}{\Gamma}\, \mathrm{H}(W_i) \qquad s.t. \sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k = 1 \tag{5.3}$$

In the above equation, Γ is a hyperparameter, $W_i = \{w_{ij}^k | j \in \{1, 2, .., N\}, k \in (a, b)\}$ is the set of all weights, and H(.) is the entropy function. We solve the optimization problem defined in (5.3) using the Lagrange multiplier technique as below:

$$L = \sum_{k\in\{a,b\}} \sum_{j=1}^{N} -w_{ij}^k (1 - |z_i^{a\intercal} z_j^k|) + \frac{1}{\Gamma} \sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k \log(w_{ij}^k) + \lambda(\sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k - 1) \tag{5.4}$$

$$\frac{\partial L}{\partial w_{ij}^k} = 0 \rightarrow -\,(1 - |z_i^{a\intercal} z_j^k|) + \frac{1}{\Gamma}\log(w_{ij}^k) + \frac{1}{\Gamma} + \lambda = 0$$

$$w_{ij}^k = \exp(\Gamma(1 - |z_i^{a\intercal} z_j^k|) - 1 - \Gamma\lambda) \tag{5.5}$$

Where λ is the Lagrange multiplier. By substituting (5.5) into the constraint of (5.3), we have:

$$\sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k = 1 \rightarrow \sum_{k\in\{a,b\}} \sum_{j=1}^{N} \exp(\Gamma(1 - |z_i^{a\intercal} z_j^k|) - 1 - \Gamma\lambda) = 1$$

$$\exp(-1 - \Gamma\lambda) = \frac{1}{\sum_{k\in\{a,b\}} \sum_{j=1}^{N} \exp(\Gamma(1 - |z_i^{a\intercal} z_j^k|))} \tag{5.6}$$

If we substitute (5.6) to (5.5), we have the final values for w_{ij}^k as below:

$$w_{ij}^k = \frac{\exp(\Gamma(1 - |z_i^{a\intercal} z_j^k|))}{\sum_{k\in\{a,b\}} \sum_{j=1}^N \exp(\Gamma(1 - |z_i^{a\intercal} z_j^k|))} \tag{5.7}$$

When comparing the loss function of other contrastive clustering algorithms, such as CC, with the loss of C3 (shown in Eq. (5.2)), several differences become apparent. Firstly, while other contrastive algorithms typically only allow one positive pair to appear in the numerator, C3 enables the networks to be trained by considering a much larger number of positive pairs. This leads to more efficient training and better performance. Secondly, C3 adopts a weighting scheme when creating the negative pairs; the scheme assigns lower weights to samples that are either too close or too far from each other while assigning higher weights to those that are more in the mixing cluster areas. This approach reduces the impact of false positive, noisy, and anomaly samples on the learning of representations.

5.3 C3 Experiments

In the previous sections of this chapter, we introduced the C3 algorithm, designed to enhance the clustering performance of the CC. Our approach leverages additional potential positive pairs and mitigates the effects of false negatives samples through a novel weighting scheme. In this section, we demonstrate the effectiveness of the proposed C3 algorithm and its components by presenting rigorous experimental evaluations. We evaluated our method on five challenging computer vision benchmark datasets: CIFAR-10, CIFAR-100 [230], ImageNet-10, ImageNet-Dog [243], and Tiny-ImageNet [248]. For CIFAR-10 and CIFAR-100, we combined the training and test splits. Also, for CIFAR-100, instead of 100 classes, we used the 20 super-classes as the ground-truth. To evaluate the performance, we use three commonly-used metrics in clustering namely clustering accuracy (ACC) [232], Normalized Mutual Information (NMI) [233], and Adjusted Rand Index (ARI) [249]. ACC quantifies the proportion of data points that are placed in the correct cluster relative to the entire dataset, providing a clear-cut gauge of a clustering algorithm's accuracy. In contrast, NMI measures how closely two sets of clusters match by capturing the mutual information shared in their assignments. Both ACC and NMI produce values between 0 and 1, with higher scores indicating better clustering performance. ARI evaluates the similarity between two clusterings while accounting for random chance; it spans from -1, denoting weak agreement, to 1, denoting a perfect match, making it a dependable tool for comparing clustering outcomes.

For the sake of fair comparison, for all datasets, we used ResNet34 [1] as the backbone of our encoder $f(.)$, which is the same architecture that previous algorithms have adopted. We set the dimension of the output of the instance-level projection head g_I to 128, for all datasets. The output dimension of the cluster-level contrastive head is set to the number of classes, M, in each dataset. All networks are initialized by the CC [23] algorithm with the hyperparameters suggested by its authors. The Adam optimizer with an initial learning rate of 0.00001 and batch size of 128 is used for C3. All networks are trained for 20 epochs. The experiments are run on NVIDIA TESLA V100 32G GPU.

5.3.1 Comparison with State-of-the-art

Table 5.1 shows the results of our proposed method on benchmark datasets, compared to state-of-the-art and some common traditional clustering methods. For CC, we run the code provided by its authors for all datasets, and the results are indicated by (*). As is evident in this table, our proposed method significantly outperforms all other baselines in all datasets. Quantitatively, comparing to the second best algorithm (i.e. CC), C3 improves the ACC by 6.6%, 3.3%, 5.0%, 1.3% and 0.3%), the NMI by 6.5%, 1.4%, 5.5%, 1.1% and 0.4%, and the ARI by 9.6%, 1.3%, 4.9%, 1.2% and 0.2%, respectively on CIFAR-10, CIFAR-100, ImageNet-10, ImageNet-Dogs, and Tiny-ImageNet. The main difference between our framework and other baselines, such as CC, is that we exploit an additional set of information, i.e. the similarity between samples, to further enhance the learned representation for clustering. In our approach, we increase the number of positive samples, which improves the true-positive-pair rate, and put more weight on clustering the samples that are more probably located on the boundary of the clusters, which leads to a decrease in

Table 5.1: Clustering performance of different methods. We conduct five random experiments to report mean and standard deviation.

Algorithm	CIFAR-10			CIFAR-100			ImageNet-10			ImageNet-Dogs		
	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI
K-means [2]	0.087	0.229	0.049	0.084	0.130	0.028	0.119	0.241	0.057	0.055	0.105	0.020
SC [250]	0.103	0.247	0.085	0.090	0.136	0.022	0.151	0.274	0.076	0.038	0.111	0.013
AC [251]	0.105	0.228	0.065	0.098	0.138	0.034	0.138	0.242	0.067	0.037	0.139	0.021
NMF [252]	0.081	0.190	0.034	0.079	0.118	0.026	0.132	0.230	0.065	0.044	0.118	0.016
AE [253]	0.239	0.314	0.169	0.100	0.165	0.048	0.210	0.317	0.152	0.104	0.185	0.073
DAE [254]	0.251	0.297	0.163	0.111	0.151	0.046	0.206	0.304	0.138	0.104	0.190	0.078
DCGAN [255]	0.265	0.315	0.176	0.120	0.151	0.045	0.225	0.346	0.157	0.121	0.174	0.078
DeCNN [256]	0.240	0.282	0.174	0.092	0.133	0.038	0.186	0.313	0.142	0.098	0.175	0.073
VAE [105]	0.254	0.291	0.167	0.108	0.152	0.040	0.193	0.334	0.168	0.107	0.179	0.079
JULE [144]	0.192	0.272	0.138	0.103	0.137	0.033	0.175	0.300	0.138	0.054	0.138	0.028
DEC [17]	0.275	0.301	0.161	0.136	0.185	0.050	0.282	0.381	0.203	0.122	0.195	0.079
DAC [41]	0.396	0.522	0.306	0.185	0.238	0.088	0.394	0.527	0.302	0.219	0.275	0.111
ADC [257]	-	0.325	-	-	0.160	-	-	-	-	-	-	-
DDC [32]	0.424	0.524	0.329	-	-	-	0.433	0.577	0.345	-	-	-
DCCM [31]	0.496	0.623	0.408	0.285	0.327	0.173	0.608	0.710	0.555	0.321	0.038	0.182
IIC [258]	-	0.617	-	-	0.257	-	-	-	-	-	-	-
PICA [178]	0.591	0.696	0.512	0.310	0.337	0.171	0.802	0.870	0.761	0.352	0.352	0.201
GATCluster [259]	0.475	0.610	0.402	0.215	0.281	0.116	0.609	0.762	0.572	0.322	0.333	0.200
CC [23]	0.678*	0.770*	0.607*	0.421*	0.423*	0.261*	0.850*	0.893*	0.811*	0.436*	0.421*	0.268*
EDESC [245]	0.627	0.464	-	0.385	0.370	-	-	-	-	-	-	-
C3 (Ours)	**0.742**	**0.836**	**0.702**	**0.436**	**0.456**	**0.274**	**0.905**	**0.944**	**0.858**	**0.447**	**0.433**	**0.279**
	± 0.002	**± 0.001**	**± 0.004**	**± 0.002**	**± 0.001**	**± 0.003**	**± 0.001**	**± 0.002**	**± 0.004**	**± 0.002**	**± 0.002**	**± 0.003**

the false-negative-pair selection rate. We believe this is the main reason our method's performance is superior compared to the baselines.

As opposed to the CC method that misses the global patterns present in each data cluster, C3 correctly tries to consider such patterns (at least partially) by employing cross-instance data similarities. Looking at (5.2) and (5.3), one can infer that C3 implicitly reduces the intra-cluster distance while maximizing the inter-cluster distance, which is what an efficient grouping technique would do in the presence of the data labels. This can be confirmed by visualizing the clusters before and after applying the C3 loss. As Fig. 5.4 show, the samples are fairly clustered after initialization with CC, i.e. before the start of training using the C3 loss. However, some of the difficult clusters are mixed in the boundaries. After initialization, clusters are expanded with a considerable number of miss-clustered data. However, after training with the proposed C3 method, we observe that the new cluster space is much more reliable, and individual clusters are densely populated while being distant from each other.

5.3.2 Convergence Analysis

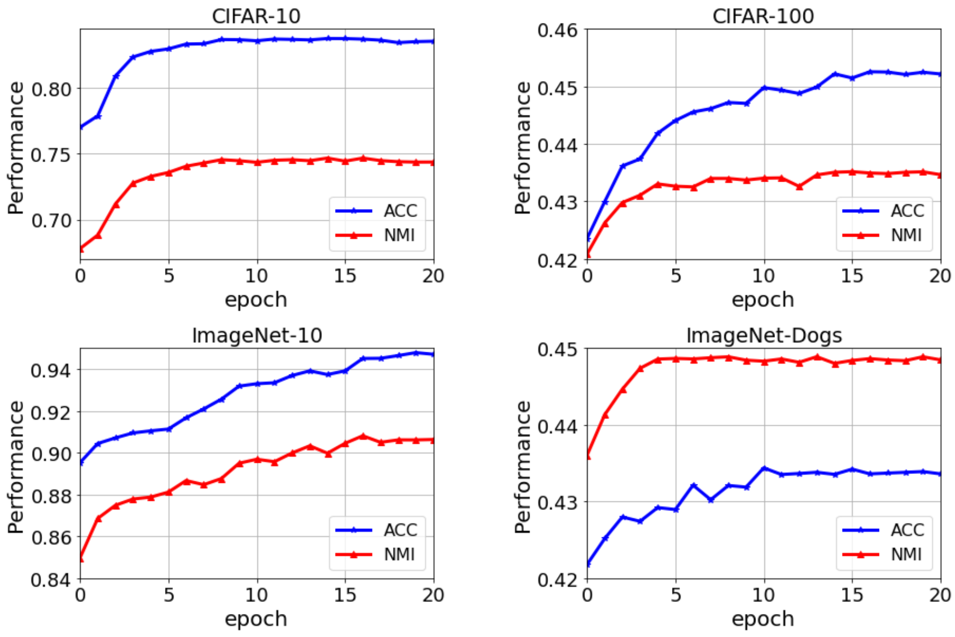


Figure 5.2: C3's performance vs epochs.

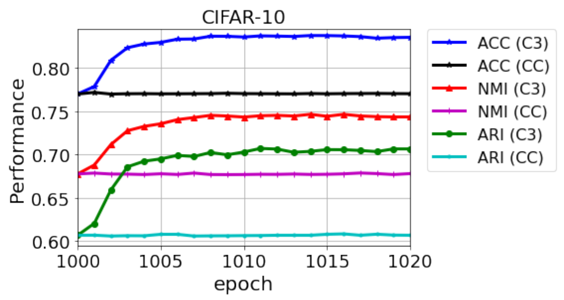


Figure 5.3: Performance of CC and C3 after 1020 epochs.

Results of section 5.3.1 depict the superiority of our proposed scheme. Now, we analyze C3's convergence and the computational complexity to evaluate at what cost it makes such an improvement over other baselines. We plotted the trend of clustering accuracy and NMI for four datasets during the training epochs in Fig. 5.2. We can readily confirm that although we are just training the C3 step for 20 epochs, the graphs quickly converge to a settling point, which corresponds to the peak performance. Also, we can observe that both ACC and NMI are improved throughout the C3 training phase in all datasets. The performance at $\mathrm{epoch} = 0$ corresponds to the clustering performance after initialization with CC. These figures clearly show that C3 improves its clustering quality and justifies the qualitative results shown in Section 5.3.1.

One may naively think that the better performance of C3 is because it is being trained for 20 more epochs; note that in all our experiments, as suggested by the CC authors, we trained the CC algorithm networks for 1000 epochs. We train the C3 networks for 1020 epochs. We reject this argument and support it by training the CC networks for the same number of epochs as what the C3 is trained for, i.e. 1020 epochs. We observe that no improvement is obtained for CC when trained for an extra 20 epochs. The result of such an experiment on CIFAR-10 is shown in Fig. 5.3.

5.3.3 How does C3 loss improve the clusters?

As we saw in Fig. 5.4, the improvement that C3 achieves is mainly because it is able to reduce the distance between instances of the same cluster while repelling them from other clusters. We can justify this observation by considering the loss function of C3, i.e. Eq. (5.2). In this function,

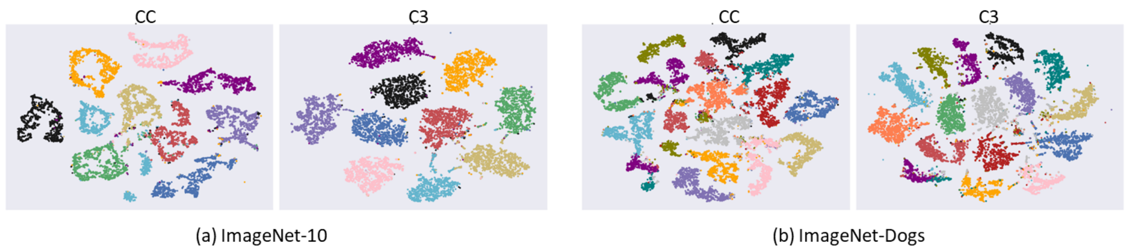


Figure 5.4: t-SNE visualization of clusters learned by the CC and C3 methods.

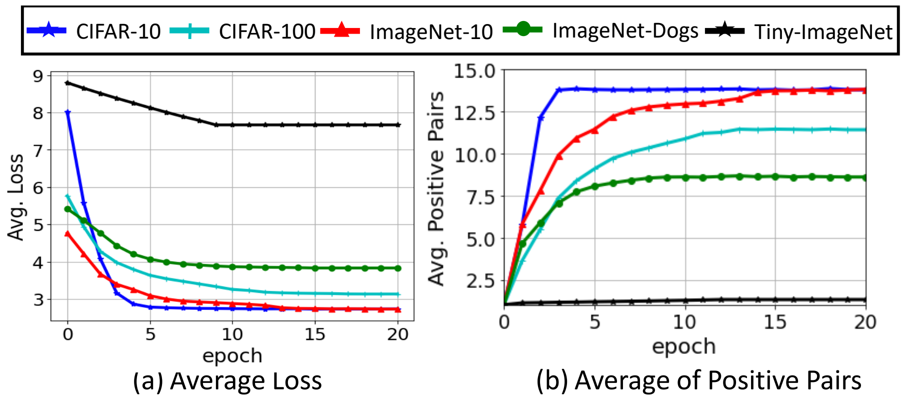


Figure 5.5: Plot of loss and number of positive pairs versus epoch.

the term $\mathbb{1}\{z_i^{a^\intercal} z_j^k \geq \zeta\}$ indicates that if the cosine similarity of two samples is greater than ζ, they should be moved further close to each other.

At the beginning of training with the C3 loss, since the networks are initiated with CC, we can assume that the points that are very similar to each other have a cosine distance, in the z-space, less than threshold ζ, so they will become further close by minimizing the C3 loss. For instance, take two points z_1 and z_2 with a cosine similarity larger than ζ, and assume that z_3 has a similarity greater than ζ with z_1, but its similarity with z_2 is smaller than ζ. Therefore, according to the loss function, z_1 and z_2, as well as z_1 and z_3, are forced to become closer, but it is not the case for z_2 and z_3. However, these two points will also implicitly move closer to each other because their distance to z_1 is reduced. As the training continues, at some point, the similarity of z_2 and z_3 also may pass the threshold ζ. Therefore, as the similar pairs move closer to each other during the training, a series of new connections will be formed, and the cluster will become denser. To support this hypothesis, we plotted the average of the loss function and the average number of positive pairs of each data sample in Fig. 5.5-a and Fig. 5.5-b, respectively. We can observe that the number of positive pairs exponentially increases during the training until it settles to form the final clusters.

Corresponding to this exponential increase, we can see that the loss is decreasing, and the network learns a representation in which clusters are distanced from each other while samples of each cluster are packed together.

We can also deduct from this experiment that the number of positive pairs is also related to the number of classes in each dataset. For example, if we have an augmented batch size of $N = 256$, for Tiny-ImageNet that has 200 classes, we expect to have $\frac{256}{200} = 1.28$ positive pairs per sample which is very close to 1 and it is the reason that we do not see the same sharp increasing trend as other datasets in Tiny-ImageNet.

5.3.4 Effect of Hyperparameter ζ and Γ

Our method, C3, introduces two new hyperparameter ζ and Γ. ζ is a threshold for identifying similar samples. Throughout the experiments, we fixed $\zeta = 0.6$, which yielded consistent results across datasets. Now, we carry out an experiment in which we change ζ and record the performance. Note that since $z_i^{a\intercal} z_j^k \in [-1, 1]$, we can technically change ζ from -1 to 1. Intuitively, for a small or negative value of ζ, most points in the z-space will be considered similar, and the resulting clusters will not be reliable. Therefore, in our experiment, we change ζ from 0.4 to 0.9 in 0.1 increments for CIFAR-10. For Tiny-ImageNet, as there are lots of clusters, we set $\zeta \in \{0.60, 0.85, 0.90, 0.95\}$. We then plot the accuracy, NMI, average loss, and the average of positive pairs per sample. The graphics are shown in Fig. 5.6.

In CIFAR-10 experiments, in Fig. 5.6-a and Fig. 5.6-b, we see that for $\zeta = 0.4$, accuracy and NMI are indeed decreasing during the C3 training. This is because this value of ζ is too lenient and considers the points not in the same cluster to be similar. We can confirm this explanation by looking at Fig. 5.6-d. We can see that for smaller ζs, we will have more average positive pairs per sample. As we increase the ζ, we can see that the performance begins to improve. For larger values such as $\zeta = 0.9$, we can see that the performance does not significantly change during the training. This is because $\zeta = 0.9$ is a strict threshold, and if we look at the number of positive pairs, only a few instances are identified as similar during the training.

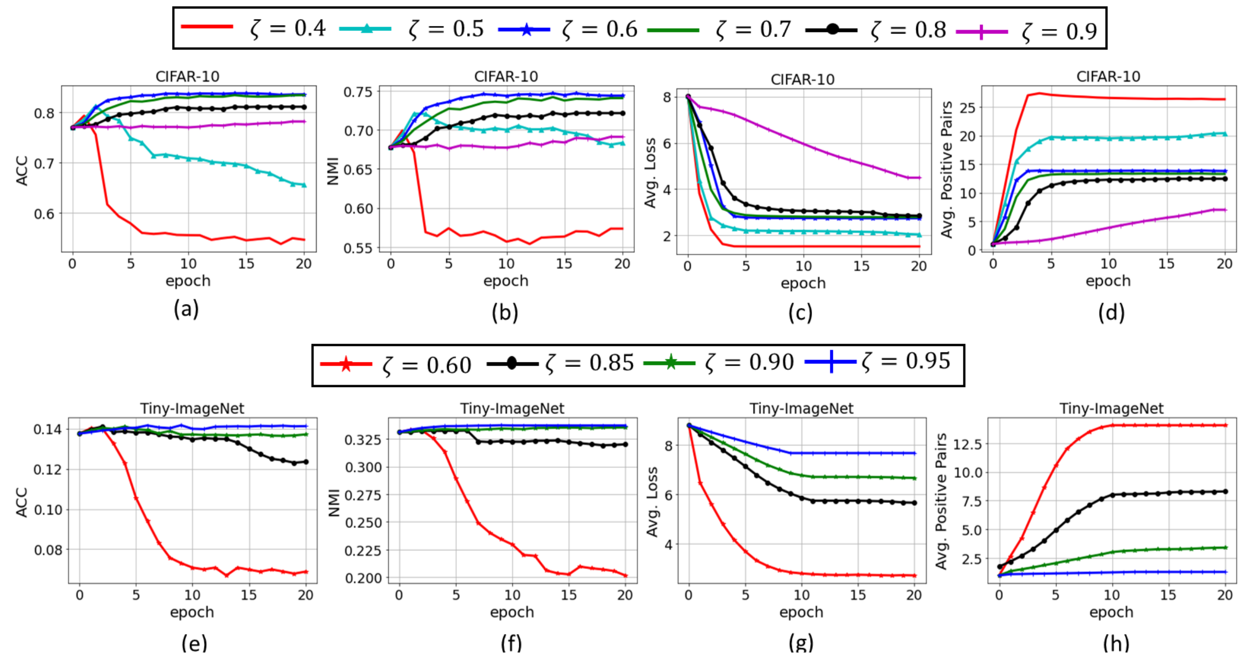


Figure 5.6: Performance and behavior of C3 for different values of ζ, for CIFAR 10 (top row) and Tiny-ImageNet (bottom row).

Comparing the results of CIFAR-10 and Tiny-ImageNet experiments shows that the value of ζ also depends on the number of clusters. Since we have 200 classes in Tiny-ImageNet, a smaller value of ζ might yield two or more clusters merging together and this would decrease the accuracy. Therefore, we should choose a more strict threshold such as $\zeta = 0.9$ or $\zeta = 0.95$ to improve. For this dataset, if ζ is set to a small value, the model might mistakenly include less similar or even false negative samples as positive pairs, weakening the quality of the training signal. This can severely impact the model's ability to learn strong feature representations, potentially hindering its convergence to an optimal solution. In Fig. 5.6-c and Fig. 5.6-g, the average loss plot also conveys interesting observations about the behaviour of ζ. We can see that for smaller values, the loss is exponentially converging to the minimum, but for larger ζ, the rate is much slower. This can be due to the fact that a smaller ζ considers most points to be similar and of the same class, and therefore, it can yield the trivial solution of considering all points to be similar and mapping them into one central point. In the extreme case of $\zeta \to 1$, C3 considers a few samples as positive pairs, and therefore, we will not have any major improvement. In contrast, if we set $\zeta \to -1$, the loss considers all points to be positive and the numerator and denominator of Eq. (5.2) become equal.

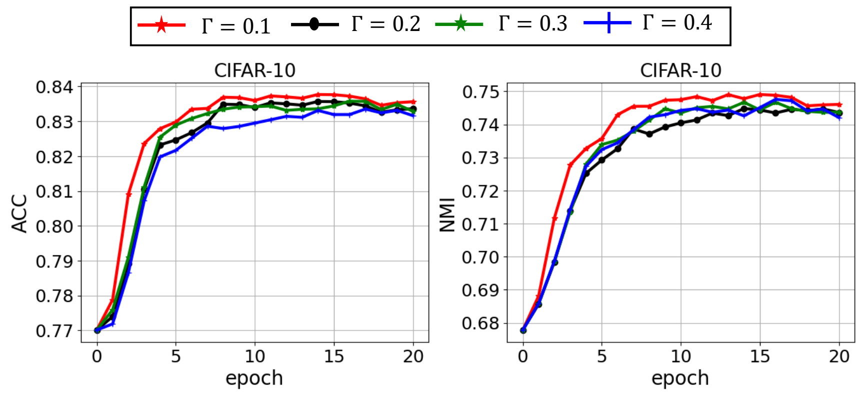


Figure 5.7: Performance of C3 for different values of Γ, for CIFAR 10.

Therefore, the loss function becomes zero and the network does not train. Following the above discussion, we suggest a value like $\zeta = 0.6$, which is a good balance. However, the choice of ζ might be influenced by the number of clusters in the data. If we have a large number of clusters, it would be better to choose a large ζ. On the other hand, if the data has a small number of clusters, a smaller ζ (but not too small) is preferred since it trains faster. In our experiments, we set $\zeta = 0.6$ unless in Tiny-ImageNet which has 200 classes where we used $\zeta = 0.95$.

Fig. 5.7 illustrates the impact of hyperparameter Γ on the performance C3. For very low values of Γ (i.e., $\Gamma \to 0$), all weights converge to the same value of $w_{ij}^k = \frac{1}{2N}$. Conversely, for very high values of Γ (i.e., $\Gamma \to \infty$), the effect of entropy in Eq. (5.3) is neglected, leading to a trivial solution where our weighting function in Eq. (5.7) selects one negative sample having the highest value of $1 - |z_i^{a\intercal} z_j^k|$ to minimize the first term in Eq. (5.3). In all our experiments for all the datasets, Γ is set to 0.1 though a better performance may be obtained if we fine-tune it per dataset. Overall, our results demonstrate the importance of selecting an appropriate value for Γ to optimize the performance of our proposed method.

5.4 Limitations of C3

In this chapter, we introduce a novel approach, called C3, to enhance the inclusion of true positive pairs while mitigating the impact of false negatives, noisy samples, and anomalous pairs during the training of contrastive clustering algorithm. The C3 methodology demonstrates significant efficacy in achieving better feature representations for clustering tasks, addressing critical challenges in contrastive learning frameworks. However, while C3 exhibits notable advantages, it also has certain limitations that must be acknowledged and addressed for broader applicability.

One prominent limitation of C3 arises when applied to datasets with a large number of clusters, such as Tiny-ImageNet [248]. As detailed in Section 5.3.3, the average number of positive pairs in a typical batch size of 256 for the Tiny-ImageNet dataset is approximately 1.28 per sample. This scarcity of positive pairs within a batch makes it challenging for C3 to effectively capture and leverage informative features among the positive pairs. While increasing the batch size could theoretically address this issue by providing a larger pool of positive pairs, it introduces significant practical challenges. The loss function in C3 relies on pairwise relationships between samples, making it computationally and memory intensive. Scaling the batch size to accommodate datasets like Tiny-ImageNet becomes non-trivial due to the exponential growth in memory requirements and processing overhead. Moreover, careful consideration must be given to the parameter ζ when working with datasets containing a large number of clusters such as Tiny-ImageNet (See Section 5.3.4). If ζ is set to a small number, the model may include less similar or true negative samples as positive pairs, which can dilute the quality of the training signal. This issue can significantly hinder the model's ability to learn robust feature representations, potentially preventing convergence to a good solution.

5.5 Summary

In this Chapter, we introduced C3, a novel algorithm designed for contrastive data clustering. The primary aim of C3 is to leverage the inherent similarity between different data instances to develop more effective representations for clustering tasks. By integrating contrastive learning principles with clustering objectives, our approach ensures that similar instances are brought closer in the representation space while maintaining separability between dissimilar instances. This process results in a more structured and cluster-friendly embedding space, which is critical for improving clustering performance. To validate the effectiveness of our approach, we conducted extensive experiments across five challenging computer vision datasets. These experiments demonstrated that C3 significantly outperforms the current state-of-the-art clustering methods in terms of key performance metrics. The results highlight the robustness and adaptability of our algorithm across diverse datasets with varying characteristics, underscoring its practical utility in real-world applications.

Chapter 6

Forward-Backward Knowledge Distillation for Continual Clustering (FBCC)

The field of Unsupervised Continual Learning (UCL) is evolving, aiming to allow neural networks to learn tasks sequentially without needing explicit labels. This is crucial because, in many scenarios, acquiring labeled data is impractical. A major hurdle in this area is Catastrophic Forgetting (CF), which occurs when a model forgets previously learned tasks as it learns new ones. This issue is particularly challenging in UCL due to the absence of labeled data, making traditional CF mitigation techniques like knowledge distillation and replay buffers less effective due to memory inefficiency and privacy concerns. To address this, this chapter introduces a novel concept called Unsupervised Continual Clustering (UCC) and proposes a method named Forward-Backward Knowledge Distillation for unsupervised Continual Clustering (FBCC) to tackle CF within UCC. FBCC employs a unique setup with a "teacher" model that learns and retains knowledge across tasks, and "student" models that specialize in specific tasks. This method involves two main phases: Forward Knowledge Distillation, where the teacher model learns new clusters with help from the student models, and Backward Knowledge Distillation, where a student model replicates the teacher's task-specific knowledge, aiding it in future tasks. This approach stands out by showing improved performance and memory efficiency in task-specific clustering compared to applying traditional clustering algorithms to the latent spaces generated by state-of-the-art UCL methods.

6.1 Notation

We introduce the UCC problem which involves learning a sequence of N tasks. Here, the set of tasks is denoted as $\Omega = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_N\}$, where $\mathcal{D}_t$ corresponds to the dataset from task t ($1 \leq t \leq N$). In the UCC setting, although labeled information is not available, it is known that data samples belonging to different tasks are from distinct classes. In other words, if $\mathcal{Y}_i$ and $\mathcal{Y}_j$ respectively represent sets of class labels from task i and task j ($i \neq j$), then $\mathcal{Y}_i \cap \mathcal{Y}_j = \varnothing$. This scenario is similar to unsupervised class-incremental setting [48, 58]. Following the clustering literature where the number of clusters is predetermined, we assume that the number of clusters present in every task is known in the UCC set-up. The number of cluster for task t is denoted by λ_t.

Given the recent success of contrastive clustering methods, we propose training FBCC's networks using losses inspired by contrastive learning. In line with other contrastive learning methodologies, we employ two sets of augmentations a, b on each sample within the dataset $\mathcal{D}_t = \{x_1, ..., x_{|\mathcal{D}_t|}\}$ at task t, yielding $\mathcal{D}_{ta} = \{x_{1a}, ..., x_{|\mathcal{D}_t|a}\}$ and $\mathcal{D}_{tb} = \{x_{1b}, ..., x_{|\mathcal{D}_t|b}\}$ respectively.

As depicted in Figure 6.1, our approach involves training a deep teacher encoder denoted as $T(.)$. The primary function of $T(.)$ is to derive a suitable clustering-friendly latent representation for the current task while also retaining knowledge from the previous tasks. The latent representations of $T(.)$ for x_{ik} is denoted as $h_{ik}^T = T(x_{ik})$. Hereafter, we use $k \in \{a, b\}$ to represent different augmentations.

In Section 6.5.5, we demonstrate that considering all previous tasks in training the current teacher is not essential. We propose training up to M light-weight student encoders, where M is a hyperparameter with $1 < M \leq N$, to alleviate catastrophic forgetting. Specifically, when $t < M$, we train t students, and when $t \geq M$, we retain the last M students while removing all others from memory. Henceforth, we denote the number of students for task t as $l_t = \mathbb{1}\{t <$

$M\}t + \mathbb{1}\{t \geq M\}M$, where $\mathbb{1}\{.\}$ is an indicator function. The r-th student encoder is denoted as $S_r(.)$, where $1 \leq r \leq l_t$. The latent representations of the r-th student encoder for x_{ik} is denoted as $h_{ik}^{S_r} = S_r(x_{ik})$. Hereafter, we utilize the notation r to denote indices ranging from 1 to l_t, i.e. $1 \leq r \leq l_t$.

To reduce dimensionality and define our contrastive loss, we employ l_t instance-level projectors. The r-th instance-level projector is denoted by $I_r(.)$. For $1 \leq r \leq l_t - 1$, the r-th instance-level projector aims to map $h_{ik}^{S_r}$ to obtain $z_{ik}^{S_r} = I_r(h_{ik}^{S_r})$. We propose to share the l_t-th instance-level projector between the l_t-th student and the teacher . Hence, $I_{l_t}(.)$ is responsible for creating $z_{ik}^{S_{l_t}} = I_{l_t}(h_{ik}^{S_{l_t}})$ and $z_{ik}^{T} = I_{l_t}(h_{ik}^{T})$.

In the following sections, we delve into our forward-backward knowledge distillation approach for continual clustering (FBCC). Firstly, we introduce forward distillation, where our teacher is trained to learn the new task t while leveraging the knowledge of previous tasks present in the previously trained student networks, i.e. students from 1 to $l_t - 1$; this is to retain the memory of previous tasks. In the forward mode, all parameters of the students are frozen, and our focus lies solely on training the teacher encoder. Conversely, in the backward mode, we propose a novel approach for training the l_t-th student, enabling it to grasp the latent representation of the teacher encoder for task t. In the backward mode, all parameters of students ranging from 1 to l_{t-1}, as well as the teacher encoder, are frozen, and our attention is directed towards updating the parameters of student l_t.

6.2 Forward Knowledge Distillation

Training Teacher on the Current Task: Since we lack access to labeled information for dataset $\mathcal{D}_t$, inspired by [21, 23], we train our teacher encoder on samples x_{ia} and x_{ib}, which are two augmentations of the same sample $x_i \in \mathcal{D}_t$, in order to maximize the similarity between z_{ia}^T and z_{ib}^T, while minimizing the similarity of z_{ia}^T and z_{ib}^T with the remaining $2|\mathcal{B}| - 2$ samples in the batch $\mathcal{B}$, where $|\mathcal{B}|$ denotes the batch size.

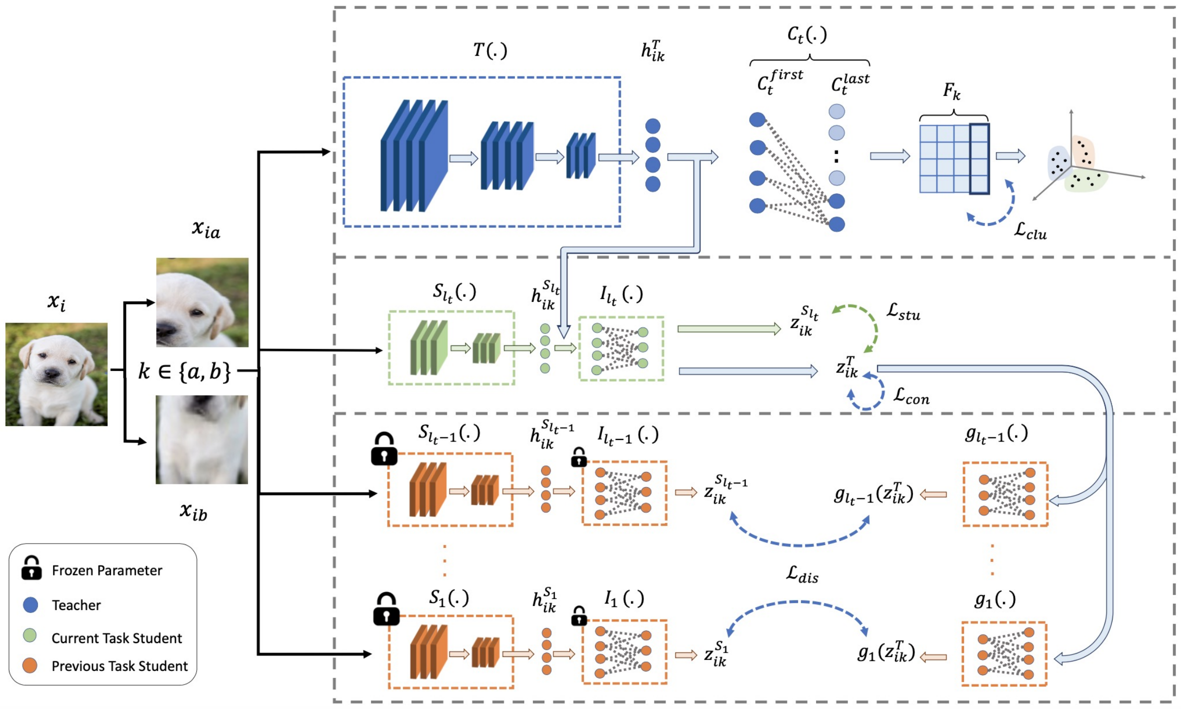


Figure 6.1: Overview of FBCC framework for task t.

Moreover, to further segregate representations of the current task from those of previous tasks, given the absence of datasets from previous tasks, we propose ensuring that augmentations of the current task t are distinctly distant from prototypes learned from previous tasks ranging from the first task up to task $t-1$. We discuss the way of defining prototypes later in this section. We present the set of prototypes from the first task up to task $t-1$ as $\mathcal{P}_{t-1}$. This approach facilitates the preservation of task-specific information and aids in reducing interference between tasks, thereby enhancing the model's performance on sequential learning tasks. We propose the following contrastive loss function (i.e. $\mathcal{L}_{con}$) for training our teacher encoder at task t.

$$\mathcal{L}_{con} = \frac{1}{2|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} (\ell_{ia}^{con} + \ell_{ib}^{con}) \tag{6.1}$$

$$\ell_{ia}^{con} = -\log\left(\frac{\exp(sim(z_{ia}^T, z_{ib}^T))}{\sum_{j\in\mathcal{B}}^{j\neq i} \sum_{k\in\{a,b\}} \exp(sim(z_{ia}^T, z_{jk}^T)) + \sum_{z_p \in \mathcal{P}_{t-1}} \exp(sim(z_{ia}^T, z_p))}\right) \tag{6.2}$$

In this study, $sim(.)$ denotes the cosine similarity between two vectors. Analogous to ℓ_{ia}^{con}, we define ℓ_{ib}^{con}, which assesses the similarity between z_{ib}^{T} and other samples in the batch as well as the prototypes set.

Knowledge Distillation from Students to Teacher: In task t, since we do not have access to $\mathcal{D}_1, ..., \mathcal{D}_{t-1}$, we propose to utilize $l_t - 1$ student networks trained on the last $l_t - 1$ tasks to aid our network in not forgetting the previous tasks. Our objective is to ensure that z_{ik}^{T} contains at least as much information as (and ideally more than) all $z_{ik}^{S_r}$, where $1 \leq r \leq l_t - 1$. Instead of enforcing similarity between z_{ik}^{T} and $z_{ik}^{S_r}$, which could discourage the new model from learning new concepts, we draw inspiration from [58] and propose to map z_{ik}^{T} using fully connected predictor networks $g_r(.)$ to the previous task learned by the r-th student (e.g. $z_{ik}^{S_r}$), while freezing the parameters of the r-th student and r-th instance-level projector. Given the well-studied effectiveness of utilizing contrastive loss between the output of the predictor and the latent representation of the previous task for the current dataset in [58], we define our loss as follows:

$$\mathcal{L}_{dis} = \frac{1}{2|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} (\ell_{ia}^{dis} + \ell_{ib}^{dis}), \tag{6.3}$$

$$\ell_{ia}^{dis} = -\frac{1}{l_t - 1} \sum_{1 \leq r < l_t} \log\left(\frac{\exp(sim(g_r(z_{ia}^{T}), \Delta(z_{ia}^{S_r})))}{\sum_{j \in \mathcal{B}}^{j \neq i} \sum_{k \in \{a,b\}} \exp(sim(g_r(z_{ia}^{T}), \Delta(z_{jk}^{S_r})))}\right), \tag{6.4}$$

where $\Delta(.)$ denotes the detaching operation, in which we detach vectors from a network and do not have a backward path to this network from our loss. With this approach, our teacher network aims to imitate the behavior of students, which serve as estimations of the previous teacher. The primary distinction between our proposed distillation framework and [58] lies in our approach to addressing the catastrophic forgetting issue. We train multiple light-weight students, each capable of serving as a reliable estimation of our teacher network, allowing us to remember more than one previous task while storing a lower number of parameters in memory. In contrast, [58] relies solely on the large-scale deep network learned in the previous task, potentially leading to forgetting of the initial tasks when confronted with numerous tasks. Moreover, in Section 6.5.4, we elucidate the significance of retaining memories of more than one previous task in mitigating the catastrophic forgetting issue.

Clustering Samples of the Current Task: Inspired by [23], for clustering samples belonging to $\mathcal{D}_t$, we propose to train a cluster-level projector network for task t, denoted by $C_t(\cdot)$. This network consists of a 2-layer fully connected network followed by a softmax function, which maps the latent representation of the teacher encoder to a suitable space designed for the clustering task. The first layer and the last layer of $C_t(\cdot)$ are denoted as C_t^{first} and C_t^{last}, respectively, i.e. $C_t = C_t^{\text{last}}(C_t^{\text{first}})$. At the beginning of task t, where $2 \leq t \leq N$, we initialize C_t^{first} with C_{t-1}^{first}. This initialization is intended to retain the information from previous tasks during this stage. The output of C_t^{first} for sample x_{ik} is denoted by $\hat{h}_{ik} = C_t^{\text{first}}(T(x_{ik}))$. Also, we propose to store C_{t-1}^{last} in memory for use during the test phase. Within C_t^{last}, we allocate λ_t neurons to transform $\hat{h}_{ik}$ into a specialized space designed for the clustering of data samples within the current task. We initialize C_t^{last} with random values. For instance, if we assume 2 clusters per task, in Figure 6.1, for the new task, we add two new neurons, shown in dark blue, to create C_t^{last}. Also, we store neurons of previous tasks shown in light blue in memory.

In every task, for a batch of data $\mathcal{B}$, we create two augmentations of the batch to obtain $\mathcal{B}_k$, where $k \in \{a, b\}$. We then pass these two augmented batches to the teacher network and the cluster-level projector to obtain $F_k = C_t(T(\mathcal{B}_k))$, where $F_k = [f_{1k}|f_{2k}|...|f_{\lambda_t k}] \in \mathbb{R}^{|\mathcal{B}| \times \lambda_t}$, and $f_{jk} \in \mathbb{R}^{|\mathcal{B}|}$ represents the probability vector for assigning samples from $\mathcal{B}_k$ to cluster j. Inspired by [23], we apply contrastive loss on the features of F_k (e.g., f_{jk}) instead of applying the contrastive loss between samples. The motivation stems from the derivation of F_a and F_b from two augmentations of the same batch. Therefore, similar clusters represented in F_a and F_b (e.g., f_{ia} and f_{ib}) are expected to possess matching probability assignments and ideally be situated far apart from dissimilar clusters. This strategy is designed to promote distinct and well-separated clusters, thereby improving the overall quality of the clustering process. The loss is defined as follow:

$$\mathcal{L}_{clu} = \frac{1}{2\lambda_t}\sum_{i=1}^{\lambda_t}(\ell_{ia}^{clu} + \ell_{ib}^{clu}) - H(F), \tag{6.5}$$

$$\ell_{ia}^{clu} = -\log(\frac{\exp(sim(f_{ia}, f_{ib}))}{\sum_{j=1\, j\neq i}^{\lambda_t}\sum_{k\in\{a,b\}}\exp(sim(f_{ia}, f_{jk})}), \tag{6.6}$$

where $H(F) = \sum_{k\in\{a,b\}}\sum_{j=1}^{\lambda_t} -\mathcal{Q}(f_{jk})\log(\mathcal{Q}(f_{jk}))$ is the entropy of cluster assignments probabilities, where $\mathcal{Q}(f_{jk}) = ||f_{jk}||_1/||F_k||_1$ and $||.||_1$ denotes the ℓ_1 norm. We maximize the entropy to avoid the trivial solution of converging all assignments to one cluster.

Updating Prototype Set: At the end of training of task t, we propose to define λ_t new prototypes that are representatives of task t. These prototypes, denoted as p_v for $1 \leq v \leq \lambda_t$, are intended for use in (6.1) where we generate representations of future tasks to be distinct from the representations learned during the current task. p_v is designed to maintain the same distance with samples belonging to the v-th cluster. Given c_{ia} and c_{ib} as the cluster assignments of x_{ia} and x_{ib} (i.e. $c_{ia} = \text{argmax}[C_t(T(x_{ia})]$ and $c_{ib} = \text{argmax}[C_t(T(x_{ib})])$, we can formulate the following equation for determining p_v.

$$p_v = \frac{\sum_{x_i\in\mathcal{B}} \mathbb{1}\{c_{ia} = v \text{ and } c_{ib} = v\}(z_{ia}^T + z_{ib}^T)}{2\sum_{x_i\in\mathcal{B}} \mathbb{1}\{c_{ia} = v \text{ and } c_{ib} = v\}} \tag{6.7}$$

To enhance the quality of prototypes for the clusters of task t, the prototype of the v-th cluster is the center of the “reliable” augmented samples in the z space. We consider a sample as “reliable” if both of its augmentations are assigned to the same cluster. Once new prototypes have been identified for task t, we incorporate them into the existing set of prototypes obtained from preceding tasks $\mathcal{P}_{t-1}$ to constitute the updated prototype set $\mathcal{P}_t$.

6.3 Backward Knowledge Distillation

In task t, we propose training a light-weight student encoder with significantly fewer parameters compared to our teacher encoder, with the aim of replicating the behavior of the teacher encoder on task t. Our objective is to store this student model in memory for use in knowledge distillation for future tasks. For each task, we limit the memory storage to at most M student encoder networks, ensuring that the total number of parameters across these networks is less than that of our teacher network. This approach enhances memory efficiency compared to methods such as [58], which necessitate storing their large-scale network from the current task in memory for knowledge distillation in subsequent tasks. In task t, we propose to train the l_t-th student network with the following loss function, while keeping the parameters of the teacher encoder and all other student encoders frozen.

$$\mathcal{L}_{stu} = \frac{1}{2|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} (\ell_{ia}^{stu} + \ell_{ib}^{stu}) \tag{6.8}$$

$$\ell_{ia}^{stu} = \frac{1}{|h_{ia}^{S_{l_t}}|} ||h_{ia}^{S_{l_t}} - \Delta(h_{ia}^T)||_2^2 - \log\left(\frac{\exp(sim(z_{ia}^{S_{l_t}}, \Delta(z_{ia}^T)))}{\sum_{j \in \mathcal{B}}^{j \neq i} \sum_{k \in \{a,b\}} \exp(sim(z_{ia}^{S_{l_t}}, \Delta(z_{jk}^T)))}\right) \tag{6.9}$$

Where $||.||_2$ represents the ℓ_2 norm and $|h_{ia}^{S_{l_t}}|$ shows the number of elements in $h_{ia}^{S_{l_t}}$. With this loss function, we aim to instruct our student network in two critical aspects: 1- Our student must grasp the representations produced by the teacher network irrespective of other samples. This is achieved through the first component of our loss. 2- Our student network must follow the same structural relationships established by the teacher encoder within a batch; we propose to enforce such behavior through defining a contrastive loss between the student and teacher network outputs, as is shown in the second term of the loss defined in (6.9). These two components of the loss function enable the l_t-th student to learn both the output and the relationships between samples produced by the teacher encoder.

6.4 Overall Training Scheme

For the batch $\mathcal{B}$ comprising data samples from task t, we initially fix the parameters of all students and conduct forward distillation to minimize the combined losses of contrastive, distillation, and clustering (i.e., $\mathcal{L}_{con} + \mathcal{L}_{dis} + \mathcal{L}_{clu}$). Subsequently, we proceed with backward distillation, wherein we freeze the parameters of the teacher and unfreeze the parameters of the l_t-th student, minimizing $\mathcal{L}_{stu}$ for the same batch $\mathcal{B}$. Figure 6.1 shows the overall training scheme for task t.

Assigning Samples to Clusters: For the UCC setting, after completing training on the last task N, to find the final cluster assignments for sample x_i, we propose to use the trained teacher encoder (i.e., T), the first layer of the cluster projector (i.e., C_N^{first}), which is shared among all tasks, and the last layers of the cluster projector, which are task-specific and stored in memory (i.e., $C_1^{\text{last}}, ..., C_N^{\text{last}}$). To assign cluster label c_i to data sample x_i, we propose to obtain the latent representation of data $\hat{h}_i = C_N^{\text{first}}(T(x_i))$, then map $\hat{h}_i$ to different clustering spaces using the last layers trained for each task and pick the index of the maximum value, i.e., $c_i = \text{argmax}[C_1^{\text{last}}(\hat{h}_i), ..., C_N^{\text{last}}(\hat{h}_i)]$.

6.5 Experiments and Results

In the previous sections of this chapter, we introduced the FBCC framework to mitigate catastrophic forgetting (CF) in unsupervised continual clustering (UCC). By employing a teacher–student setup, FBCC reduces the number of parameters compared to similar methods (e.g., [58]) while effectively learning new tasks without forgetting previous ones. FBCC operates in two phases: forward KD, during which the teacher model learns new clusters with support from student models, and backward KD, in which a student model replicates the teacher's task-specific knowledge to aid in future tasks. In this section, we conduct comprehensive experiments to illustrate the effectiveness of our proposed FBCC. We assess our model's performance on three challenging computer vision benchmark datasets: CIFAR-10 [230] (with 10 classes and 5 tasks), CIFAR-100 [230] (with 100 classes and 10 tasks), and Tiny-ImageNet [243] (with 200 classes and 10 tasks). To train our model, we concatenate the train and test sets of the datasets, a common practice in clustering research (e.g., [17, 23]). Also, we run each experiment 5 times to measure mean and standard deviation.

6.5.1 Evaluation Metrics

We evaluate the performance of our continual clustering model (FBCC) using two key metrics: average clustering accuracy ($\overline{\text{ACC}}$) and average forgetting ($\overline{\text{F}}$), where ACC [232] is a widely used metric for assessing clustering performance and average forgetting is a common metric used to measure how much information the model has forgotten about previous tasks. $\overline{\text{ACC}}$ and $\overline{\text{F}}$ are defined as follows:

$$\overline{\text{ACC}} = \frac{1}{\text{N}} \sum_{t=1}^{\text{N}} \text{ACC}_{t,\text{N}} \tag{6.10}$$

$$\overline{\text{F}} = \frac{1}{\text{N}-1} \sum_{i=1}^{\text{N}-1} \max_{t \in \{1,..,\text{N}-1\}} (\text{ACC}_{i,t} - \text{ACC}_{i,\text{N}}), \tag{6.11}$$

where $\text{ACC}_{i,j}$ is ACC of task i at the end of training of task j. $\overline{\text{ACC}}$ ranges from 0 to 1, with higher values indicating better performance, while $\overline{\text{F}}$ typically ranges from 0 to 1, but in this case, lower values are preferable.

6.5.2 Implementation Details

We utilize ResNet-18 [1] as our teacher network, which comprises approximately 11.5 million (11.5m) parameters. As for our student networks, we employ SqueezeNet 1.1, containing around 1.2m parameters. To align the output dimension of SqueezeNet with that of ResNet-18, we append a single-layer fully connected network without an activation function at the end of SqueezeNet. The maximum number of students (i.e M) is determined as $\lceil \frac{\text{N}}{2} \rceil$; for example, for CIFAR-10, M is set to 3. Both instance projectors and predictors are implemented as 2-layer fully connected neural networks, with dimensions $d \to 512 \to 128$, where $d = 512$ for instance projectors and $d = 128$ for predictors. The cluster projector is designed as a 2-layer fully connected neural network with dimensions $512 \to 512 \to \lambda_t$, where λ_t denotes the number of clusters in the current task. It's worth re-emphasizing that the first layer of the cluster-projector is shared between tasks, and we store the last layer in memory for predicting cluster assignments. In all experiments the batch size is 256.

6.5.3 Comparison Results

Table 6.1: FBCC Performance Comparison in terms of $\overline{\text{ACC}}(\%)$ and $\overline{\text{F}}(\%)$.

Algorithms	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	$\overline{\text{ACC}}$ ($\uparrow$)	$\overline{\text{F}}$ ($\downarrow$)	$\overline{\text{ACC}}$ ($\uparrow$)	$\overline{\text{F}}$ ($\downarrow$)	$\overline{\text{ACC}}$ ($\uparrow$)	$\overline{\text{F}}$ ($\downarrow$)
CC (offline)	79.00	-	42.90	-	14.00	-
Co^{2}L (SCL)	28.35	14.05	19.88	10.30	8.69	4.95
OCD-Net (SCL)	40.41	7.03	18.06	7.46	7.97	5.31
CCL	36.56	6.21	19.59	8.51	8.21	4.68
STAM	39.61	5.15	25.34	6.25	9.21	4.26
LUMP	56.43	12.76	19.53	6.16	10.53	2.51
CaSSLe	40.56	3.28	36.67	3.92	17.45	2.69
POCON	57.20	3.59	35.29	4.28	16.25	3.54
FBCC	**75.28 $\pm$ 0.81**	**2.29 $\pm$ 0.29**	**38.73 $\pm$ 0.64**	**3.62 $\pm$ 0.21**	**18.36 $\pm$ 0.32**	**1.95 $\pm$ 0.27**

To the best of our knowledge, no existing continual learning algorithm is designed explicitly for the clustering task. Therefore, in this section, we compare our proposed FBCC algorithm with state-of-the-art UCL algorithms such as CCL [260], STAM [261], LUMP [65], CaSSLe [58]. Also, we compare our FBCC with two state-of-the art SCL method Co^{2}L [198] and OCD-Net [59] on three benchmark datasets. Note that Co^{2}L and OCD-Net are supervised methods that make use of data labels during their training phase. We adopt a common approach followed in the field, as outlined in [69], by applying the spectral clustering algorithm to the latent representations learned by other algorithms. This approach allows us to evaluate and report $\overline{\text{ACC}}$ and $\overline{\text{F}}$ for these methods. Comparison results are presented in Table 6.1.

Moreover, we compare our FBCC with a baseline CC [23] algorithm. CC possesses the flexibility to define its loss function using any pair of samples from distinct clusters, rendering it more potent compared to FBCC, which, in each step, only has access to partial clusters. Consequently, we employ CC as a proxy upper bound for assessing the performance of our algorithm. Note that the lower performance of CC on the Tiny-ImageNet compared to FBCC can be associated with the fact that CC is a memory-hungry algorithm, and running it for more than 256 samples per batch is practically impossible [23, 43] while the given 256 samples might not be enough for defining relationship between samples when dealing with numerous clusters like the case of Tiny-ImageNet. Yet, CC serves as a proper upper bound for datasets with a small number of classes, such as CIFAR-10 and CIFAR-100.

As illustrated in Table 6.1, FBCC demonstrates notable superiority over alternative algorithms concerning both $\overline{\text{ACC}}$ and $\overline{\text{F}}$. Notably, FBCC surpasses Co^{2}L and OCD-Net, which learns latent representations of data in a supervised manner, and the state-of-the-art UCL algorithm, CaSSLe, while employing fewer parameters. Specifically, FBCC utilizes 15.1m parameters for CIFAR-10 and 17.5m for CIFAR-100 and Tiny-ImageNet, whereas CaSSLe employs 23m parameters across all datasets. The advantage of FBCC stems from two key factors: Firstly, FBCC is tailored for clustering tasks, concurrently learning data sample representations and performing clustering, whereas other UCL algorithms primarily focus on refining data representation. Secondly, FBCC adeptly retains knowledge from multiple previous tasks by leveraging multiple, but a limited number of, specialized student models to mimic the representations acquired by the teacher model for specific tasks.

6.5.4 Ablation Study

Effectiveness of Prototypes in Forward Knowledge Distillation: To demonstrate the effectiveness of including prototypes learned from the previous task (i.e., $\mathcal{P}_{t-1}$) during training in the forward distillation phase, we propose to exclude prototypes and focus solely on the contrastive loss, i.e., we remove the second term in the denominator of (6.2). All other model configurations remain unchanged. This setup is labeled as FBCC w/o Pro in Table 6.2.

Upon comparing the results obtained from FBCC and FBCC w/o Pro, it is evident that the inclusion of prototypes in forward distillation leads to improved $\overline{\text{ACC}}$ and $\overline{\text{F}}$ across various tasks. This enhancement is attributed to our model's ability to effectively distinguish between data from

Table 6.2: Ablation study of FBCC in terms of $\overline{\text{ACC}}(\%)$ and $\overline{\text{F}}(\%)$. The best result for continual learning algorithms in each column is highlighted in bold. We report the average performance over 5 experiments with the same initialization for each network.

Algorithms	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	$\overline{\text{ACC}}$ (↑)	$\overline{\text{F}}$ (↓)	$\overline{\text{ACC}}$ (↑)	$\overline{\text{F}}$ (↓)	$\overline{\text{ACC}}$ (↑)	$\overline{\text{F}}$ (↓)
FBCC w/o Pro	74.81	2.49	37.73	4.18	17.82	2.45
FBCC w/o KD	67.39	9.01	32.58	13.25	14.35	6.62
FBCC$_{CaSSLe}$	70.81	4.54	35.29	6.58	15.36	2.98
FBCC	**75.28**	**2.29**	**38.73**	**3.62**	**18.36**	**1.95**

the current task and prototypes, which serve as representatives of previous tasks.

Effectiveness of Students in Forward Knowledge Distillation: In Table 6.2, we present a comparison of our proposed method with two alternative configurations in terms of $\overline{\text{ACC}}$ and $\overline{\text{F}}$. In one of these configurations, denoted as FBCC w/o KD, we exclude the knowledge distillation loss from students to the teacher (i.e., $\mathcal{L}_{dis}$) when updating the parameters of the teacher encoder. In the second configuration, inspired by [58], instead of training multiple students, we employ a strategy where we utilize a previously trained teacher model to mitigate catastrophic forgetting. We freeze the parameters of this copied network, and the knowledge distillation loss is defined in [58]. This configuration is labeled as FBCC$_{CaSSLe}$ in Table 6.2.

If we compare FBCC w/o KD with FBCC, we observe approximately a 5.88% improvement in terms of $\overline{\text{ACC}}$ and 7.05% improvement in terms of $\overline{\text{F}}$ across all datasets. This improvement is primarily attributed to the effectiveness of knowledge distillation from students to the teacher using $\mathcal{L}_{dis}$ in retaining knowledge from previous tasks.

Moreover, upon comparing results obtained from FBCC + CaSSLe with those from FBCC, we can conclude that the effectiveness of having multiple students lies in retaining knowledge from more than one previous task. It is worth noting that the number of parameters for FBCC on CIFAR-10, CIFAR-100, and Tiny-ImageNet are 15.1m, 17.5m, and 17.5m, respectively, while the number of parameters for FBCC + CaSSLe for all datasets is 23m. Our FBCC achieves better results in terms of 3.58% improvement in $\overline{\text{ACC}}$ and exhibits 2.03% improvement in terms of $\overline{\text{F}}$ across all datasets despite having fewer parameters.

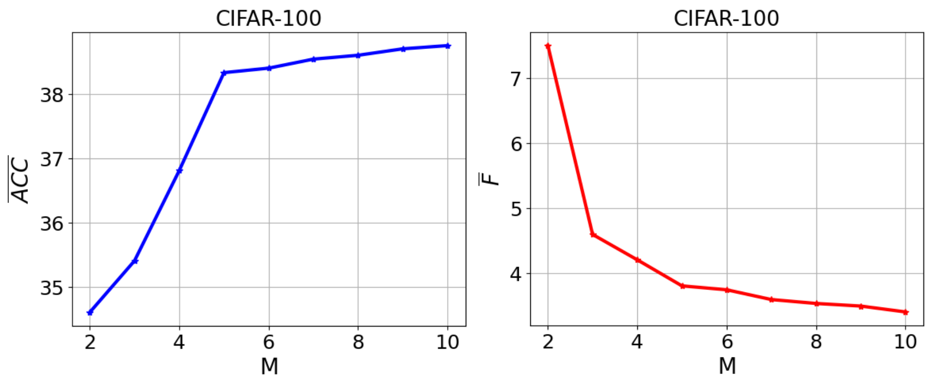


Figure 6.2: Average ACC and Average Forgetting of different values of M.

6.5.5 Effect of Number of Students in Forward Knowledge Distillation:

In this section, we investigate the effect of the number of students (i.e., M) on the performance of FBCC in terms of $\overline{\text{ACC}}$ and $\overline{\text{F}}$. We vary M from 2 to 10 for the CIFAR-100 dataset and report the results in terms of $\overline{\text{ACC}}$ and $\overline{\text{F}}$. The findings are illustrated in Fig. 6.2. As shown in this figure, we observe a significant improvement in performance when changing M from 2 to 5. This improvement is attributed to our model's ability to effectively remember previous tasks. However, when changing M from 6 to 10, we do not observe much improvement. This is because forcing our model to remember numerous previous tasks limits its ability to learn new tasks effectively.

6.5.6 Effectiveness of Backward Distillation

In this section, we delve into the effectiveness of $\mathcal{L}_{stu}$ in transferring knowledge from teacher to student on CIFAR-100 dataset. To achieve this, we define an average over the difference between ACC of the teacher and the ACC of the student. This is defined as follows:

$$\hat{\text{ACC}} = \frac{1}{N} \sum_{t \in \{1,\ldots,N\}} (\text{ACC}_{t,t}^{T} - \text{ACC}_{t,t}^{S_{l_t}}) \tag{6.12}$$

where $\text{ACC}_{t,t}^{T}$ and $\text{ACC}_{t,t}^{S_{l_t}}$ represent the ACC of the teacher and the student on the task t after completing training on the task t, respectively. To obtain $\text{ACC}_{t,t}^{S_{l_t}}$ for dataset $\mathcal{D}_t$, after completing training on $\mathcal{D}_t$, we take the output of the student, i.e. $h_{\mathcal{D}_t}^{S_{l_t}}$, and feed it into the cluster projector learned during forward distillation to obtain cluster assignments for the dataset, i.e. $c_{\mathcal{D}_t} = \text{argmax}[C_t(h_{\mathcal{D}_t}^{S_{l_t}})]$. Subsequently, we compare these assignments with the true cluster assignments to compute the $\text{ACC}_{t,t}^{S_{l_t}}$.

Moreover, we consider three architectures for the student, namely MobileNetV3 Small [262], ShuffleNetV2 (0.5x) [263], and SqueezeNet 1.1 [264]. Inspired by [265], to select the best model for our student, we define a distillation score (DS) that takes into account the size and accuracy of the student relative to those of the teacher network. This score helps us identify the optimal model for the student. The formula for DS is defined as follows:

$$DS = \alpha(\frac{\#Param_S}{\#Param_T}) + (1-\alpha)(1 - \frac{1}{N}\sum_{t=1}^{N}\frac{\text{ACC}_{t,t}^{S_{l_t}}}{\text{ACC}_{t,t}^{T}}), \tag{6.13}$$

where $\#\text{Param}_S$ and $\#\text{Param}_T$ represent the number of parameters of the student and teacher network, respectively, and $\alpha \in [0, 1]$ is a hyperparameter that highlights the importance of the first ratio over the second one. Lower DS values indicate better models. In our experiments, α is set to 0.5 and the teacher network is ResNet-18 with 11.5 million parameters.

Table 6.3: Comparison of Different Student Architectures in Terms of number of parameters, $\hat{\text{ACC}}(\%)$ and DS. The Best Result in Each Column is Highlighted in Bold

Students	CIFAR-100		
	$\#Param_S$ (↓)	$\hat{\text{ACC}}$ (↓)	DS (↓)
MobileNetV3 Small	2.5m	**1.05**	0.120
ShuffleNetV2 (0.5x)	1.3m	1.91	0.081
SqueezeNet 1.1	**1.2m**	1.68	**0.075**

Table 6.3 shows the comparison of different student architecture in terms of number of parameters, $\hat{\text{ACC}}(\%)$, and DS. By comparing the $\hat{\text{ACC}}$ of students with different architectures, we can infer the effectiveness of $\mathcal{L}_{stu}$ in knowledge distillation from the teacher to the students. For instance, the ACC of SqueezeNet 1.1 with 1.2 million parameters is 1.68% less than the ACC of ResNet-18 with 11.5 million parameters on average across different tasks on the CIFAR-100 dataset. Based on DS reported in Table 6.3, we choose SqueezeNet 1.1 as our student network for

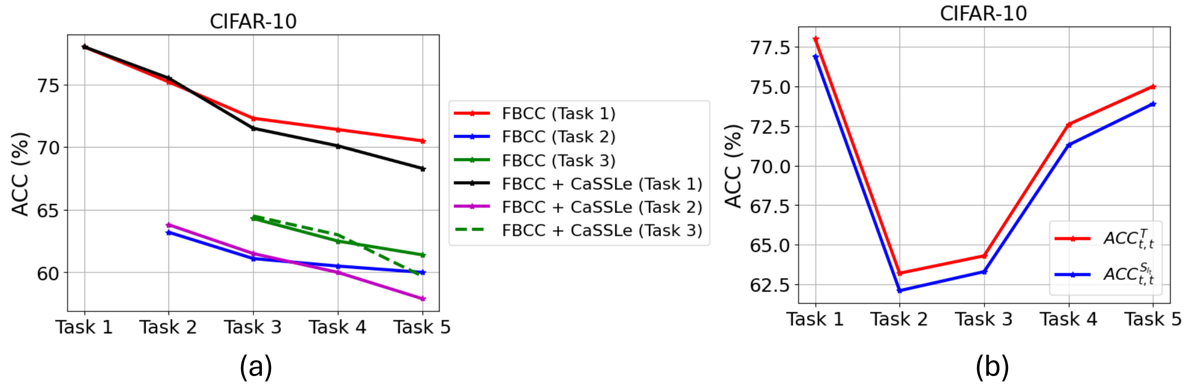


Figure 6.3: Experiments on imbalanced data.

the CIFAR-100 dataset, as it has the lowest DS among the other architectures. We observe similar pattern for the other datasets as well.

6.5.7 Imbalanced Dataset

In this section, we delve into assessing the efficacy of our proposed FBCC in tackling learning tasks characterized by highly imbalanced sample distributions. To accomplish this, we adopt a strategy wherein we selectively sample data from task t within the CIFAR-10 dataset. Instances from the first task are incorporated into the training set with a likelihood of 0.1, while instances from the final task are included with a likelihood of 1. Instances from intermediate tasks are chosen proportionally, following a linear progression. The inherent challenge posed by imbalanced data lies in the scenario where our model is trained on a limited number of instances from the current task, yet it encounters increasingly more samples from subsequent tasks. This imbalance heightens the risk of CF wherein the model's performance on the current task deteriorates as it learns new tasks, potentially leading to performance degradation in future tasks.

In Figure 6.3-(a), we present a comparative analysis between our proposed FBCC approach and FBCC + CaSSLe. The figure illustrates ACC achieved on the first three tasks $(1 \leq t \leq 3)$ after completing training on each task. As depicted in the figure, transitioning from task t to task $t + 1$ reveals that the performance of FBCC + CaSSLe for task t surpasses that of FBCC. This discrepancy arises due to the utilization of a teacher network explicitly trained on task t within the FBCC + CaSSLe framework. In contrast, FBCC employs a student network with notably fewer parameters for task retention. However, following task $t + 2$, a reversal in performance is observed. FBCC exhibits superior performance on task t compared to FBCC + CaSSLe. This shift can be attributed to FBCC's utilization of multiple specialized student networks, thereby enhancing its capacity to retain knowledge from previous tasks effectively. For example, this characteristic becomes particularly evident when examining the performance on task 1. At the conclusion of training on task 2, FBCC + CaSSLe exhibits superior performance compared to FBCC. However, after completing task 3, the trend reverses, with FBCC surpassing FBCC + CaSSLe in terms of ACC.

Furthermore, to demonstrate the effectiveness of training students to mimic the behavior of the teacher network on imbalanced data, we plot $\mathrm{ACC}_{t,t}^{\mathrm{T}}$ and $\mathrm{ACC}_{t,t}^{\mathrm{S}_{l_t}}$, as discussed in Section 6.5.6, for imbalanced data in Figure 6.3-(b). For our experiments, we utilize SqueezeNet 1.1 [264]. As depicted in the figure, our student network adeptly follows the teacher network in generating quality representations for the each task on imbalanced datasets.

6.5.8 Effectiveness of FBCC in Heterogeneous Tasks

In this section, we delve into the impact of heterogeneous tasks, where the number of clusters varies across tasks. The primary challenge posed by such heterogeneity for a continual learner lies in addressing catastrophic forgetting (CF), wherein the model must retain knowledge from tasks with a high number of clusters while adapting to subsequent tasks. To assess our model's ability to handle such scenarios, we delineate two cases using the CIFAR-100 dataset, aiming to evaluate its performance and compare it with the state-of-the-art UCL algorithm CaSSLe [58]. In **Case 1**, we define the number of clusters as 50-10-10-10-10-10 for tasks 1 through 5, respectively, from left to right. Here, our model encounters half of the total clusters in the initial task before facing a consistent number of clusters across subsequent tasks. In **Case 2** (50-30-10-5-5), we present a more challenging setup where our model encounters a higher number of clusters initially, followed by tasks with progressively fewer clusters. This case emphasizes the significance of ensuring that the model retains knowledge from tasks with a greater number of clusters while adapting to those with fewer clusters. Comparison results are presented in Table 6.4. In this section, we employ three students for conducting experiments. As depicted in Table 6.4, the performance of FBCC surpasses that of CaSSLe. This superiority can be attributed to the fact that most forgetting occurs after task 1 and then task 2 for both Case 1 and Case 2. Our model exhibits the capability to remember previous tasks more effectively due to the utilization of multiple well-trained students. In contrast, CaSSLe relies on a single teacher for retaining knowledge from each preceding task, leading to a gradual forgetting phenomenon. This effect is particularly pronounced for tasks 1 and 2. Additionally, it is worth noting that the number of parameters in our model for this experiment is 15.1 million, whereas the number of parameters in CaSSLe is 23 million. This significant difference in parameter count underscores the efficiency of our approach in achieving competitive performance with fewer parameters.

Table 6.4: FBCC and CaSSLe performance on heterogeneous tasks scenario in terms of $\overline{\text{ACC}}(\%)$ and $\overline{\text{F}}(\%)$. The best result for continual learning algorithms in each column is highlighted in bold.

Algorithms	Case 1		Case 2	
	$\overline{\text{ACC}}$ (↑)	$\overline{\text{F}}$ (↓)	$\overline{\text{ACC}}$ (↑)	$\overline{\text{F}}$ (↓)
CaSSLe	48.74	7.89	45.61	8.51
FBCC	**51.33 ± 0.26**	**4.33 ± 0.20**	**48.83 ± 0.49**	**4.61 ± 0.19**

6.6 Limitations of FBCC

In this chapter, we introduce the FBCC framework, a novel approach designed to tackle key challenges within the field of UCC. FBCC demonstrates its effectiveness in addressing several critical issues faced by continual learning systems, particularly in mitigating CF and maintaining robust performance across a sequence of tasks. However, despite its strengths, the framework is not without limitations, which we detail in this section. A significant limitation of FBCC stems from its reliance on cluster-level contrastive learning to define its clustering loss. This method aims to distinguish between clusters by maximizing inter-cluster contrast (see Section 6.2). While this approach is highly effective for tasks with multiple clusters, it faces a fundamental issue when a task consists of only a single cluster. In such cases, FBCC cannot compute the clustering loss because contrastive learning requires the presence of at least two clusters for meaningful comparisons. Moreover, as discussed in Section 6.5.6, the process of KD from teacher models to student models is inherently imperfect, often leading to a performance drop during Forward KD. For example, as shown in Table 6.3, experiments on the CIFAR-100 dataset reveal an average ACC drop of 1.68% when transitioning from ResNet-18 to SqueezeNet 1.1 across different tasks. While it is acknowledged that smaller student networks may not achieve the same level of performance as their larger teacher counterparts for a specific task, FBCC distinguishes itself from algorithms like [58] by emphasizing the retention of knowledge across multiple tasks. This advantage is particularly evident when dealing with imbalanced tasks, as highlighted in Section 6.5.7.

6.7 Summary

UCL has emerged as a promising approach for sequential learning tasks where explicit labels are not available. However, a critical challenge in this domain lies in the lack of effective unsupervised continual clustering methods. This limitation is further compounded by the phenomenon of CF, where models lose previously acquired knowledge when exposed to new data. Current strategies to address CF, such as knowledge distillation and replay buffers, provide partial solutions but come with their own set of limitations. To address this, we propose FBCC, which uses a continual learner (the "teacher") and multiple student models to effectively combat CF. Through Forward and Backward Knowledge Distillation, FBCC allows the teacher to learn new clusters while retaining past knowledge. We validate the effectiveness of FBCC through a series of experiments across various datasets and dynamic environments. Our results demonstrate that FBCC not only outperforms existing CF mitigation strategies but also sets a novel benchmark for unsupervised continual clustering.

Chapter 7

Conclusions and Discussion

In this thesis, we have presented several innovative deep learning-based clustering frameworks that address prevailing challenges in the field, such as crisp assignment issues, the neglect of pairwise relationships, and the uniform treatment of clusters. Our Deep Multi-Representation Learning (DML) framework introduces multiple optimized latent spaces tailored to cluster data effectively, significantly improving performance on imbalanced datasets. Furthermore, the Deep Clustering with Self-Supervision using Pairwise Data Similarities (DCSS) model leverages hypersphere formations and enhanced pairwise data considerations, advancing the state of clustering beyond conventional methods. Additionally, the Cross-instance guided Contrastive Clustering (C3) method enhances clustering accuracy by focusing on cross-instance relationships, effectively reducing false-negative rates and improving positive pair identification. Lastly, our exploration into Unsupervised Continual Clustering (UCC) with the Forward-Backward Knowledge Distillation (FBCC) approach innovatively tackles Catastrophic Forgetting in a clustering context, paving the way for more robust and memory-efficient unsupervised learning applications.

Collectively, these methodologies not only surpass existing state-of-the-art clustering approaches but also open new avenues for research in complex and dynamic data environments. Future work may expand upon these frameworks to include more adaptive and scalable solutions, potentially integrating these techniques with real-world applications across various domains to further validate and enhance their practicality and effectiveness.

All in all, the primary challenge with DML lies in its reliance on multiple expert models to cluster dataset samples. Despite our efforts to minimize the total number of expert models, datasets with a large number of clusters tend to require more networks compared to those with fewer clusters. Therefore, DML proves to be more effective and applicable for datasets with a lower number of clusters, provided there are many samples within each cluster to mitigate the risk of overfitting. On the other hand, DCSS and C3 algorithms are better suited for more complex datasets. This is because the number of parameters in their network does not depend on the number of clusters. Additionally, they utilize the pairwise relationships between samples to assign them to clusters, making them a more scalable and efficient approach for handling intricate data structures. In this thesis, we aim to address the clustering problem under the assumption that the number of clusters (e.g., K) is known. When K is unknown, our algorithm's performance can be significantly hindered. For instance, DML relies on K to determine the number of cluster-specific autoencoders, DCSS requires K to specify the number of neurons at the end of the MNet, and C3 depends on K to define the cluster-level projector. In scenarios where K is not provided, an alternative approach is to estimate K beforehand using techniques such as the elbow method [266]. The elbow method evaluates the variance explained by different numbers of clusters and identifies an optimal K based on where the rate of improvement decreases.

7.1 Future Works

7.1.1 Improving our image clustering frameworks using LLMs

Traditional image clustering methods often focus on single objects, which becomes problematic when an object naturally spans multiple clusters. However, recent developments in large language models (LLMs) and vision-language models (VLMs) offer promising ways around this issue. LLMs have excelled at handling and generating natural language, and their capabilities have been extended to image data as well. Meanwhile, VLMs—or multimodal models—combine visual and textual information to tackle complex tasks requiring an understanding of both modalities. A key breakthrough in these models is their ability to learn shared representations of images and text, typically by first pretraining on extensive datasets of paired image-text samples and then fine-tuning for specific tasks. This joint representation allows the model to use text-based context to enhance its interpretation of images, and vice versa, resulting in more accurate and context-aware outputs in applications like content generation, interactive AI, and cross-modal retrieval. To address the single-object clustering challenge, we propose using a pretrained VLM to derive rich information from each image in a dataset by creating high-dimensional feature vectors based on text embeddings. Incorporating these vectors and evaluating the relationships between samples allows us to assign data points to multiple clusters, thus boosting both accuracy and robustness. By leveraging the advanced capabilities of VLMs, this approach moves beyond the constraints of traditional clustering methods and enables more sophisticated, precise clustering solutions.

7.1.2 Efficient Knowledge Distillation for LLMs

In Chapter 6, we introduced a knowledge distillation strategy between teacher and student models for continual clustering of image datasets. This method effectively lowers model complexity while preserving performance. Building on this foundation, our approach presents exciting possibilities for future work, particularly in reducing the size of language models. Since language models typically consume significant computational and memory resources, applying our distillation technique can lead to more compact architectures that still achieve high accuracy. Such a reduction in model size is especially advantageous for real-time settings, where both rapid inference and efficient memory usage are critical. A prime example is autonomous driving, where vast quantities of data must be processed instantly to make accurate decisions. Incorporating our distillation-based techniques into lightweight language models could substantially enhance the performance of autonomous systems by offering fast and reliable language processing without overstepping the strict resource limits these systems must observe.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[2] S. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[3] J. C. Bezdek, R. Ehrlich, and W. Full, “Fcm: The fuzzy c-means clustering algorithm,” *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing Systems*, 2002, pp. 849–856.

[5] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, “Deep clustering: Discriminative embeddings for segmentation and separation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 31–35.

[6] X. Hu, Q. Tan, and N. Liu, “Deep representation learning for social network analysis,” *Frontiers in Big Data*, vol. 2, pp. 2–18, 2019.

[7] M. Wang and W. Deng, “Deep face recognition with clustering based domain adaptation,” *Neurocomputing*, vol. 393, pp. 1–14, 2020.

[8] W. Zhu, C.-Y. Wang, K.-L. Tseng, S.-H. Lai, and B. Wang, “Local-adaptive face recognition via graph-based meta-clustering and regularized adaptation,” in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 20 301–20 310.

[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[10] Y. Kong and Y. Fu, “Human action recognition and prediction: A survey,” *International Journal of Computer Vision*, vol. 130, no. 5, pp. 1366–1401, 2022.

[11] S. Ryu, H. Choi, H. Lee, and H. Kim, “Convolutional autoencoder based feature extraction and clustering for customer load analysis,” *IEEE Transactions on Power Systems*, vol. 35, no. 2, pp. 1048–1060, 2019.

[12] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, “Auto-encoder based data clustering,” in *Iberoamerican Congress on Pattern Recognition*. Springer, 2013, pp. 117–124.

[13] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[14] J. Haseeb, M. Mansoori, Y. Hirose, H. Al-Sahaf, and I. Welch, “Autoencoder-based feature construction for iot attacks clustering,” *Future Generation Computer Systems*, vol. 127, pp. 487–502, 2022.

[15] M. Sadeghi and N. Armanfard, “Idecf: Improved deep embedding clustering with deep fuzzy supervision,” in *IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 1009–1013.

[16] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, “Learning deep representations for graph clustering,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[17] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International Conference on Machine Learning*, 2016, pp. 478–487.

[18] X. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with local structure preservation.” in *International Joint Conference on Artificial Intelligence(IJCAI)*, 2017, pp. 1753–1759.

[19] M. M. Fard, T. Thonet, and E. Gaussier, “Deep k-means: Jointly clustering with k-means and learning representations,” *Pattern Recognition Letters*, vol. 138, pp. 185–192, 2020.

[20] Z. Feng, C. Xu, and D. Tao, “Self-supervised representation learning by rotation feature decoupling,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 364–10 374.

[21] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 1597–1607.

[22] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[23] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, “Contrastive clustering,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8547–8555.

[24] Z. Dang, C. Deng, X. Yang, and H. Huang, “Doubly contrastive deep clustering,” *arXiv preprint arXiv:2103.05484*, 2021.

[25] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” in *International Conference on Machine Learning*, 2017, pp. 3861–3870.

[26] Y. Opochinsky, S. E. Chazan, S. Gannot, and J. Goldberger, “K-autoencoders deep clustering,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4037–4041.

[27] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[28] K. Tian, S. Zhou, and J. Guan, “Deepcluster: A general clustering framework based on deep learning,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 809–825.

[29] J. Pan, Y. Qian, F. Li, and Q. Guo, “Image deep clustering based on local-topology embedding,” *Pattern Recognition Letters*, vol. 151, pp. 88–94, 2021.

[30] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, “Deep adaptive image clustering,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5879–5887.

[31] J. Wu, K. Long, F. Wang, C. Qian, C. Li, Z. Lin, and H. Zha, “Deep comprehensive correlation mining for image clustering,” in *International Conference on Computer Vision (ICCV)*, 2019.

[32] J. Chang, Y. Guo, L. Wang, G. Meng, S. Xiang, and C. Pan, “Deep discriminative clustering analysis,” *arXiv preprint arXiv:1905.01681*, 2019.

[33] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, “Scan: Learning to classify images without labels,” in *Proceedings of the European Conference on Computer Vision*, 2020.

[34] M. Kaya and H. Ş. Bilge, “Deep metric learning: A survey,” *Symmetry*, vol. 11, no. 9, pp. 1066–1093, 2019.

[35] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, “Deep metric learning with angular loss,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2593–2601.

[36] E. Xing, M. Jordan, S. J. Russell, and A. Ng, “Distance metric learning with application to clustering with side-information,” *Advances in Neural Information Processing Systems*, vol. 15, pp. 521–528, 2002.

[37] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[38] Y. Xie, J. Zhang, Z. Liao, J. Verjans, C. Shen, and Y. Xia, “Pairwise relation learning for semi-supervised gland segmentation,” in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*. Springer, 2020, pp. 417–427.

[39] H. Hojjati, T. K. K. Ho, and N. Armanfard, “Self-supervised anomaly detection: A survey and outlook,” *ArXiv*, vol. abs/2205.05173, 2022.

[40] E. Xie, J. Ding, W. Wang, X. Zhan, H. Xu, P. Sun, Z. Li, and P. Luo, “Detco: Unsupervised contrastive learning for object detection,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8372–8381.

[41] W. Harchaoui, P.-A. Mattei, and C. Bouveyron, “Deep adversarial gaussian mixture auto-encoder for clustering,” in *ICLR 2017 Workshop Proposals*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:69963066

[42] M. Sadeghi and N. Armanfard, “Deep multirepresentation learning for data clustering,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 15 675–15 686, 2024.

[43] M. Sadeghi, H. Hojjati, and N. Armanfard, “C3: Cross-instance guided contrastive clustering,” *The 34th British Machine Vision Conference (BMVC)*, 2023.

[44] K. Shaheen, M. A. Hanif, O. Hasan, and M. Shafique, “Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks,” *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, p. 9, 2022.

[45] A. Singh, M. B. Gurbuz, S. S. Gantha, and P. Jasti, “Class-incremental continual learning for general purpose healthcare models,” *arXiv preprint arXiv:2311.04301*, 2023.

[46] Y. Ouyang, J. Shi, H. Wei, and H. Gao, “Incremental learning for personalized recommender systems,” *arXiv preprint arXiv:2108.13299*, 2021.

[47] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Dıaz-Rodrıguez, “Continual learning for robotics,” *arXiv preprint arXiv:1907.00182*, 2019.

[48] L. Wang, X. Zhang, H. Su, and J. Zhu, “A comprehensive survey of continual learning: Theory, method and application,” *arXiv preprint arXiv:2302.00487*, 2023.

[49] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[50] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, “Progress & compress: A scalable framework for continual learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4528–4537.

[51] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[52] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, “On tiny episodic memories in continual learning,” *arXiv preprint arXiv:1902.10486*, 2019.

[53] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, “Learning to learn without forgetting by maximizing transfer and minimizing interference,” *arXiv preprint arXiv:1810.11910*, 2018.

[54] L. Caccia, E. Belilovsky, M. Caccia, and J. Pineau, “Online learned continual compression with adaptive quantization modules,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 1240–1250.

[55] X. Liu, C. Wu, M. Menta, L. Herranz, B. Raducanu, A. D. Bagdanov, S. Jui, and J. v. de Weijer, “Generative feature replay for class-incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 226–227.

[56] R. Gao and W. Liu, “Ddgr: Continual learning with deep diffusion-based generative replay,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 10 744–10 763.

[57] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[58] E. Fini, V. G. T. Da Costa, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal, “Self-supervised models are continual learners,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9621–9630.

[59] J. Li, Z. Ji, G. Wang, Q. Wang, and F. Gao, “Learning from students: Online contrastive distillation network for general continual learning,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2022, pp. 3215–3221.

[60] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, “Dark experience for general continual learning: a strong, simple baseline,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 920–15 930, 2020.

[61] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3589–3599.

[62] F. Feng, R. H. Chan, X. Shi, Y. Zhang, and Q. She, "Challenges in task incremental learning for assistive robotics," *IEEE Access*, vol. 8, pp. 3434–3441, 2019.

[63] L. Wang, K. Yang, C. Li, L. Hong, Z. Li, and J. Zhu, "Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5383–5392.

[64] A. Lechat, S. Herbin, and F. Jurie, "Semi-supervised class incremental learning," in *The 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 383–10 389.

[65] D. Madaan, J. Yoon, Y. Li, Y. Liu, and S. J. Hwang, "Representational continuity for unsupervised continual learning," *arXiv preprint arXiv:2110.06976*, 2021.

[66] J. Liu, K. Wu, Q. Nie, Y. Chen, B.-B. Gao, Y. Liu, J. Wang, C. Wang, and F. Zheng, "Unsupervised continual anomaly detection with contrastively-learned prompt," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 4, 2024, pp. 3639–3647.

[67] T.-D. Truong, P. Helton, A. Moustafa, J. D. Cothren, and K. Luu, "Conda: Continual unsupervised domain adaptation learning in visual perception for self-driving cars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5642–5650.

[68] J. He and X. Yu, "BrainUICL: An unsupervised individual continual learning framework for EEG applications," in *Submitted to The Thirteenth International Conference on Learning Representations*, 2024.

[69] X. Yu, Y. Guo, S. Gao, and T. Rosing, "Scale: Online self-supervised lifelong learning without prior knowledge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2483–2494.

[70] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-incremental learning: survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2022.

[71] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[72] S. Chen and W. Guo, "Auto-encoders in deep learning—a review with new perspectives," *Mathematics*, vol. 11, no. 8, pp. 1777–1831, 2023.

[73] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[74] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *Wireless Telecommunications Symposium (WTS)*. IEEE, 2018, pp. 1–5.

[75] H. Cui and L. Zdeborová, "High-dimensional asymptotics of denoising autoencoders," *Advances in Neural Information Processing Systems*, vol. 36, pp. 11 850–11 890, 2023.

[76] K. Shinde, V. Itier, J. Mennesson, D. Vasiukov, and M. Shakoor, "Dimensionality reduction through convolutional autoencoders for fracture patterns prediction," *Applied Mathematical Modelling*, vol. 114, pp. 94–113, 2023.

[77] G. Parmar, D. Li, K. Lee, and Z. Tu, "Dual contradistinctive generative autoencoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 823–832.

[78] Y. Liang and W. Liang, "Reswcae: Biometric pattern image denoising using residual wavelet-conditioned autoencoder," *arXiv preprint arXiv:2307.12255*, 2023.

[79] C. Zhang and L. Xue, "Autoencoder with emotion embedding for speech emotion recognition," *IEEE Access*, vol. 9, pp. 51 231–51 241, 2021.

[80] D. Ferreira, S. Silva, A. Abelha, and J. Machado, “Recommendation system using autoencoders,” *Applied Sciences*, vol. 10, no. 16, p. 5510, 2020.

[81] D. Pratella, S. Ait-El-Mkadem Saadi, S. Bannwarth, V. Paquis-Fluckinger, and S. Bottini, “A survey of autoencoder algorithms to pave the diagnosis of rare diseases,” *International Journal of Molecular Sciences*, vol. 22, no. 19, p. 10891, 2021.

[82] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d’Enza, A. Markos, and E. Tuzhilina, “Principal component analysis,” *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[83] F. Yang, L. Herranz, J. Van De Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozerov, “Variable rate deep image compression with modulated autoencoder,” *IEEE Signal Processing Letters*, vol. 27, pp. 331–335, 2020.

[84] V. Alves de Oliveira, M. Chabert, T. Oberlin, C. Poulliat, M. Bruno, C. Latry, M. Carlavan, S. Henrot, F. Falzon, and R. Camarero, “Reduced-complexity end-to-end variational autoencoder for on board satellite image compression,” *Remote Sensing*, vol. 13, no. 3, p. 447, 2021.

[85] Q. T. Phan, Y. K. Wu, and Q. D. Phan, “A hybrid wind power forecasting model with xgboost, data preprocessing considering different nwps,” *Applied Sciences*, vol. 11, no. 3, p. 1100, 2021.

[86] U. Maduranga, K. Wijegunarathna, S. Weerasinghe, I. Perera, and A. Wickramarachchi, “Dimensionality reduction for cluster identification in metagenomics using autoencoders,” in *20th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, 2020, pp. 113–118.

[87] M. Ranzato, C. Poultney, S. Chopra, and Y. Cun, “Efficient learning of sparse representations with an energy-based model,” *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[88] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, no. 2, p. 28, 2024.

[89] J. Ngiam, Z. Chen, S. Bhaskar, P. Koh, and A. Ng, "Sparse filtering," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[90] A. Makhzani and B. Frey, "K-sparse autoencoders," *arXiv preprint arXiv:1312.5663*, 2013.

[91] C. Li, W. Zhang, G. Peng, and S. Liu, "Bearing fault diagnosis using fully-connected winner-take-all autoencoder," *IEEE Access*, vol. 6, pp. 6103–6115, 2018.

[92] J. Liu, S. Wang, and W. Yang, "Sparse autoencoder for social image understanding," *Neurocomputing*, vol. 369, pp. 122–133, 2019.

[93] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *Humaine Association Conference on Affective Computing and Intelligent Interaction*. IEEE, 2013, pp. 511–516.

[94] R. Huben, H. Cunningham, L. R. Smith, A. Ewart, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," in *The Twelfth International Conference on Learning Representations*, 2023.

[95] G. Szlobodnyik, "Extracting activity patterns from altering biological networks: A sparse autoencoder approach," in *IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, 2019, pp. 555–558.

[96] S. H. Kabil and H. Bourlard, "From undercomplete to sparse overcomplete autoencoders to improve lf-mmi speech recognition," *Interspeech*, pp. 1061–1065, 2022.

[97] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine learning*, 2008, pp. 1096–1103.

[98] L. Gondara, “Medical image denoising using convolutional denoising autoencoders,” in *IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 241–246.

[99] S. Agarwal, A. Agarwal, and M. Deshmukh, “Denoising images with varying noises using autoencoders,” in *Computer Vision and Image Processing*. Springer, 2020, pp. 3–14.

[100] D. Xiao, C. Qin, H. Yu, Y. Huang, C. Liu, and J. Zhang, “Unsupervised machine fault diagnosis for noisy domain adaptation using marginal denoising autoencoder based on acoustic signals,” *Measurement*, vol. 176, p. 109186, 2021.

[101] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[102] C. Ou, H. Zhu, Y. A. Shardt, L. Ye, X. Yuan, Y. Wang, C. Yang, and W. Gui, “Missing-data imputation with position-encoding denoising auto-encoders for industrial processes,” *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–11, 2024.

[103] P. Liang, W. Shi, and X. Zhang, “Remote sensing image classification based on stacked denoising autoencoder,” *Remote Sensing*, vol. 10, no. 1, p. 16, 2017.

[104] W. El-Shafai, S. A. El-Nabi, E.-S. M. El-Rabaie, A. M. Ali, N. F. Soliman, A. D. Algarni, A. El-Samie, and E. Fathi, “Efficient deep-learning-based autoencoder denoising approach for medical image diagnosis.” *Computers, Materials & Continua*, vol. 70, no. 3, 2022.

[105] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[106] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework.” vol. 3, 2017.

[107] C. K. Sonderby, T. Raiko, L. Maaloe, S. K. Sonderby, and O. Winther, "Ladder variational autoencoders," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[108] W. Harvey, S. Naderiparizi, and F. Wood, "Conditional image generation by conditioning variational auto-encoders," *arXiv preprint arXiv:2102.12037*, 2021.

[109] D. Wang, Y. Yan, R. Qiu, Y. Zhu, K. Guan, A. Margenot, and H. Tong, "Networked time series imputation via position-aware graph enhanced variational autoencoders," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2256–2268.

[110] T. Sudak and S. Tschiatschek, "Posterior consistency for missing data in variational autoencoders," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 508–524.

[111] D. M. S. Bhatti and B. J. Choi, "Enhancing iot healthcare with federated learning and variational autoencoder," *Sensors*, vol. 24, no. 11, p. 3632, 2024.

[112] J. Yan, M. Ma, and Z. Yu, "bmvae: a variational autoencoder method for clustering single-cell mutation data," *Bioinformatics*, vol. 39, no. 1, p. 790, 2023.

[113] E. Sevgen, J. Moller, A. Lange, J. Parker, S. Quigley, J. Mayer, P. Srivastava, S. Gayatri, D. Hosfield, M. Korshunova *et al.*, "Prot-vae: protein transformer variational autoencoder for functional protein design," *bioRxiv*, 2023.

[114] S. Cao, D. Joshi, L. Gui, and Y.-X. Wang, "Hassod: Hierarchical adaptive self-supervised object detection," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[115] S.-C. Huang, A. Pareek, M. Jensen, M. P. Lungren, S. Yeung, and A. S. Chaudhari, "Self-supervised learning for medical image classification: a systematic review and implementation guidelines," *NPJ Digital Medicine*, vol. 6, no. 1, p. 74, 2023.

[116] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[117] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[118] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[119] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 335–355, 2023.

[120] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision*. Springer, 2016, pp. 69–84.

[121] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[122] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 310–12 320.

[123] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," *arXiv preprint arXiv:2105.04906*, 2021.

[124] A. Bardes, J. Ponce, and Y.LeCun, "Vicregl: Self-supervised learning of local visual features," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8799–8810, 2022.

[125] W. G. C. Bandara, C. M. De Melo, and V. M. Patel, "Guarding barlow twins against overfitting with mixed samples," *arXiv preprint arXiv:2312.02151*, 2023.

[126] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[127] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9650–9660.

[128] B. Han, W. Huang, Z. Chen, and Y. Qian, “Improving dino-based self-supervised speaker verification with progressive cluster-aware training,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. IEEE, 2023, pp. 1–5.

[129] Y. Chen, H. Zhou, and Z. C. Lipton, “Moco-transfer: Investigating out-of-distribution contrastive learning for limited-data domains,” *arXiv preprint arXiv:2311.09401*, 2023.

[130] C.-Y. Chuang, J. Robinson, Y.-C. Lin, A. Torralba, and S. Jegelka, “Debiased contrastive learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 8765–8775, 2020.

[131] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[132] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” in *16th European Conference on Computer Vision*. Springer, 2020, pp. 776–794.

[133] I. Misra and L. v. d. Maaten, “Self-supervised learning of pretext-invariant representations,” in *Proceedings of the IEEE/CVF Conference on Computer Cision and Pattern Recognition*, 2020, pp. 6707–6717.

[134] J. C. Bezdek, R. Ehrlich, and W. Full, “Fcm: The fuzzy c-means clustering algorithm,” *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[135] D. Geary, “Mixture models: Inference and applications to clustering,” *Journal of the Royal Statistical Society*, vol. 152, no. 1, pp. 126–127, 1989.

[136] I. E. Kaya, A. Ç. Pehlivanlı, E. G. Sekizkardeş, and T. Ibrikci, “Pca based clustering for brain tumor segmentation of t1w mri images,” *Computer Methods and Programs in Biomedicine*, vol. 140, pp. 19–28, 2017.

[137] K. Tang, Z. Su, W. Jiang, J. Zhang, X. Sun, and X. Luo, “Robust subspace learning-based low-rank representation for manifold clustering,” *Neural Computing and Applications*, vol. 31, no. 11, pp. 7921–7933, 2019.

[138] B. A. Kelkar and S. F. Rodd, “Subspace clustering—a survey,” in *Data Management, Analytics and Innovation*. Springer, 2019, pp. 209–220.

[139] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data: a review,” *Acm SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[140] S. Fortunato, “Community detection in graphs,” *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[141] Y. Han and M. Filippone, “Mini-batch spectral clustering,” in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3888–3895.

[142] M. El Gheche, G. Chierchia, and P. Frossard, “Stochastic gradient descent for spectral embedding with implicit orthogonality constraint,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3567–3571.

[143] R. Toosi, M. Sadeghi, H. B. Yazdi, and M. A. Akhaee, “Fast and accurate spectral clustering via augmented lagrangian,” *Journal of Computational Science*, vol. 64, p. 101860, 2022.

[144] J. Yang, D. Parikh, and D. Batra, “Joint unsupervised learning of deep representations and image clusters,” in *IEEE/ CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5147–5156.

[145] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, “Deep adaptive image clustering,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5879–5887.

[146] T. V. Souza and C. Zanchettin, “Improving deep image clustering with spatial transformer layers,” in *The 28th International Conference Artificial Neural Networks*. Springer, 2019, pp. 641–654.

[147] J. Liang, Y. Cui, Q. Wang, T. Geng, W. Wang, and D. Liu, "Clusterfomer: clustering as a universal visual learner," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[148] I. M. Metaxas, G. Tzimiropoulos, and I. Patras, "Divclust: Controlling diversity in deep clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3418–3428.

[149] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *22nd International Conference on Pattern Recognition*, 2014, pp. 1532–1537.

[150] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *28th AAAI Conference on Artificial Intelligence*, 2014.

[151] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[152] M. C. Nascimento and A. C. De Carvalho, "Spectral methods for graph clustering–a survey," *European Journal of Operational Research*, vol. 211, no. 2, pp. 221–231, 2011.

[153] N. Mrabah, M. Bouguessa, and R. Ksantini, "Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 4, pp. 1603–1617, 2020.

[154] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5736–5745.

[155] Q. An, J. Wu, T. Huang, L. Huang, H. Luo, and F. Ye, "Clustering network based on pre-clustering center codebook," in *18th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 2023, pp. 471–476.

[156] W. Wu, W. Wang, X. Jia, and X. Feng, "Transformer autoencoder for k-means efficient clustering," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108612, 2024.

[157] W. Doo and H. Kim, “Simultaneous deep clustering and feature selection via k-concrete autoencoder,” *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[158] S. Hosseini and Z. A. Varzaneh, “Deep text clustering using stacked autoencoder,” *Multimedia Tools and Applications*, vol. 81, no. 8, pp. 10 861–10 881, 2022.

[159] F. Nan, R. Ding, R. Nallapati, and B. Xiang, “Topic modeling with wasserstein autoencoders,” *arXiv preprint arXiv:1907.12374*, 2019.

[160] K.-L. Lim, X. Jiang, and C. Yi, “Deep clustering with variational autoencoder,” *IEEE Signal Processing Letters*, vol. 27, pp. 231–235, 2020.

[161] Y. Xie, X. Wang, D. Jiang, and R. Xu, “High-performance community detection in social networks using a deep transitive autoencoder,” *Information Sciences*, vol. 493, pp. 75–90, 2019.

[162] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

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

[164] J. Guo, W. Fan, M. Amayri, and N. Bouguila, “Deep clustering analysis via variational autoencoder with gamma mixture latent embeddings,” *Neural Networks*, p. 106979, 2024.

[165] D. A. B. Oliveira and L. E. C. La Rosa, “Improving variational autoencoders reconstruction using prototypes,” *Research Square Preprints*, 2023. [Online]. Available: https://doi.org/10.21203/rs.3.rs-2777723/v1

[166] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, “A survey of clustering with deep learning: From the perspective of network architecture,” *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[167] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2180–2188.

[168] Y. Yu and W.-J. Zhou, "Mixture of gans for clustering," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018, pp. 3047–3053.

[169] Z. Yang, J. Wen, and C. Davatzikos, "Smile-gans: Semi-supervised clustering via gans for dissecting brain disease heterogeneity from medical images," *arXiv preprint arXiv:2006.15255*, 2020.

[170] S. R. Dustakar, L. K. Rao, and B. Vipparthi, "An automated medical image segmentation framework using deep learning and variational autoencoders with conditional neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 8, pp. 568–578, 2023.

[171] N. Purandhar, S. Ayyasamy, and P. Siva Kumar, "Classification of clustered health care data analysis using generative adversarial networks (gan)," *Soft Computing*, vol. 26, no. 12, pp. 5511–5521, 2022.

[172] W. Khan, S. Abidin, M. Arif, M. Ishrat, M. Haleem, A. A. Shaikh, N. A. Farooqui, and S. M. Faisal, "Anomalous node detection in attributed social networks using dual variational autoencoder with generative adversarial networks," *Data Science and Management*, vol. 7, no. 2, pp. 89–98, 2024.

[173] Y. Qu, S. Yu, W. Zhou, and Y. Tian, "Gan-driven personalized spatial-temporal private data sharing in cyber-physical social systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2576–2586, 2020.

[174] A. A. Pol, V. Berger, C. Germain, G. Cerminara, and M. Pierini, "Anomaly detection with conditional variational autoencoders," in *18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2019, pp. 1651–1657.

[175] S. Liang, Z. Pan, w. liu, J. Yin, and M. de Rijke, “A survey on variational autoencoders in recommender systems,” *ACM Computing Surveys*, 2024.

[176] X. Ji, J. F. Henriques, and A. Vedaldi, “Invariant information clustering for unsupervised image classification and segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9865–9874.

[177] J. Wu, K. Long, F. Wang, C. Qian, C. Li, Z. Lin, and H. Zha, “Deep comprehensive correlation mining for image clustering,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8150–8159.

[178] J. Huang, S. Gong, and X. Zhu, “Deep semantic clustering by partition confidence maximisation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8849–8858.

[179] G. Shiran and D. Weinshall, “Multi-modal deep clustering: Unsupervised partitioning of images,” in *The 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4728–4735.

[180] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, “Scan: Learning to classify images without labels,” in *European Conference on Computer Vision*. Springer, 2020, pp. 268–285.

[181] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9912–9924, 2020.

[182] X. Deng, D. Huang, D.-H. Chen, C.-D. Wang, and J.-H. Lai, “Strongly augmented contrastive clustering,” *Pattern Recognition*, vol. 139, p. 109470, 2023.

[183] Q. Qian, “Stable cluster discrimination for deep clustering,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 16 645–16 654.

[184] V. Sharma, M. Tapaswi, M. S. Sarfraz, and R. Stiefelhagen, "Video face clustering with self-supervised representation learning," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 2, pp. 145–157, 2019.

[185] E. Ahn, D. Feng, and J. Kim, "A spatial guided self-supervised clustering network for medical image segmentation," in *The 24th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2021).* Springer, 2021, pp. 379–388.

[186] H. Shi and C. Wang, "Self-supervised document clustering based on bert with data augment," *arXiv preprint arXiv:2011.08523*, 2020.

[187] C. Rios-Martinez, N. Bhattacharya, A. P. Amini, L. Crawford, and K. K. Yang, "Deep self-supervised learning for biosynthetic gene cluster detection and product classification," *PLOS Computational Biology*, vol. 19, no. 5, p. e1011162, 2023.

[188] H. Hu, J. P. Bindu, and J. Laskin, "Self-supervised clustering of mass spectrometry imaging data using contrastive learning," *Chemical Science*, vol. 13, no. 1, pp. 90–98, 2022.

[189] Y. Zhang, X. Zhou, Q. Meng, F. Zhu, Y. Xu, Z. Shen, and L. Cui, "Multi-modal food recommendation using clustering and self-supervised learning," *arXiv preprint arXiv:2406.18962*, 2024.

[190] H. Ritter, A. Botev, and D. Barber, "Online structured laplace approximations for overcoming catastrophic forgetting," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[191] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *2018 24th International Conference on Pattern Recognition (ICPR).* IEEE, 2018, pp. 2262–2268.

[192] M. K. Titsias, J. Schwarz, A. G. d. G. Matthews, R. Pascanu, and Y. W. Teh, "Functional regularisation for continual learning with gaussian processes," *arXiv preprint arXiv:1901.11356*, 2019.

[193] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[194] D. Park, S. Hong, B. Han, and K. M. Lee, “Continual learning by asymmetric loss approximation with single-side overestimation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3335–3344.

[195] L. Wang, X. Zhang, K. Yang, L. Yu, C. Li, L. Hong, S. Zhang, Z. Li, Y. Zhong, and J. Zhu, “Memory replay with data compression for continual learning,” *arXiv preprint arXiv:2202.06592*, 2022.

[196] S. Ebrahimi, S. Petryk, A. Gokul, W. Gan, J. E. Gonzalez, M. Rohrbach, and T. Darrell, “Remembering for the right reasons: Explanations reduce catastrophic forgetting,” *Applied AI Letters*, vol. 2, no. 4, p. e44, 2021.

[197] V. Khan, S. Cygert, B. Twardowski, and T. Trzciński, “Looking through the past: better knowledge retention for generative replay in continual learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3496–3500.

[198] H. Cha, J. Lee, and J. Shin, “Co2l: Contrastive continual learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9516–9525.

[199] Q. Pham, C. Liu, and S. Hoi, “Dualnet: Continual learning, fast and slow,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 131–16 144, 2021.

[200] A. Gomez-Villa, B. Twardowski, K. Wang, and J. van de Weijer, “Plasticity-optimized complementary networks for unsupervised continual learning,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 1690–1700.

[201] A. Mallya, D. Davis, and S. Lazebnik, “Piggyback: Adapting a single network to multiple tasks by learning to mask weights,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82.

[202] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4548–4557.

[203] A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[204] H. Jin and E. Kim, “Helpful or harmful: Inter-task association in continual learning,” in *European Conference on Computer Vision*. Springer, 2022, pp. 519–535.

[205] Q. Gao, X. Shan, Y. Zhang, and F. Zhou, “Enhancing knowledge transfer for task incremental learning with data-free subnetwork,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 471–68 484, 2023.

[206] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.

[207] F. Du, Y. Yang, Z. Zhao, and Z. Zeng, “Efficient perturbation inference and expandable network for continual learning,” *Neural Networks*, vol. 159, pp. 97–106, 2023.

[208] Z. Hu, Y. Li, J. Lyu, D. Gao, and N. Vasconcelos, “Dense network expansion for class incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 858–11 867.

[209] P. Yun, Y. Liu, and M. Liu, “In defense of knowledge distillation for task incremental learning and its application in 3d object detection,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2012–2019, 2021.

[210] Y. Fu, H. Cao, X. Chen, and J. Ding, "Task-incremental broad learning system for multi-component intelligent fault diagnosis of machinery," *Knowledge-Based Systems*, vol. 246, p. 108730, 2022.

[211] M. Liu, S. Bian, B. Zhou, and P. Lukowicz, "ikan: Global incremental learning with kan for human activity recognition across heterogeneous datasets," in *Proceedings of the 2024 ACM International Symposium on Wearable Computers*, 2024, pp. 89–95.

[212] G. Kim, C. Xiao, T. Konishi, Z. Ke, and B. Liu, "A theoretical study on solving continual learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5065–5079, 2022.

[213] G. Kim, C. Xiao, T. Konishi, and B. Liu, "Learnability and algorithm for continual learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 16 877–16 896.

[214] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, "Few-shot class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 183–12 192.

[215] W. Pian, S. Mo, Y. Guo, and Y. Tian, "Audio-visual class-incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7799–7811.

[216] H. Zhao, H. Wang, Y. Fu, F. Wu, and X. Li, "Memory-efficient class-incremental learning for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5966–5977, 2021.

[217] N. Dong, Y. Zhang, M. Ding, and Y. Bai, "Class-incremental object detection," *Pattern Recognition*, vol. 139, p. 109488, 2023.

[218] Y. Zhao, Y. Zheng, B. Yu, Z. Tian, D. Lee, J. Sun, H. Yu, Y. Li, and N. L. Zhang, "Semi-supervised lifelong language learning," *arXiv preprint arXiv:2211.13050*, 2022.

[219] L. Kou, D. Zhao, H. Han, X. Xu, S. Gong, and L. Wang, “Sscl-transmd: Semi-supervised continual learning transformer for malicious software detection,” *Applied Sciences*, vol. 13, no. 22, p. 12255, 2023.

[220] Y. Fan, Y. Wang, P. Zhu, and Q. Hu, “Dynamic sub-graph distillation for robust semi-supervised continual learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 11 927–11 935.

[221] L. Caccia and J. Pineau, “Special: Self-supervised pretraining for continual learning,” in *International Workshop on Continual Semi-Supervised Learning*. Springer, 2021, pp. 91–103.

[222] Z. Kang, E. Fini, M. Nabi, E. Ricci, and K. Alahari, “A soft nearest-neighbor framework for continual semi-supervised learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 868–11 877.

[223] M. Boschini, P. Buzzega, L. Bonicelli, A. Porrello, and S. Calderara, “Continual semi-supervised learning through contrastive interpolation consistency,” *Pattern Recognition Letters*, vol. 162, pp. 9–14, 2022.

[224] P. Skierś and K. Deja, “Joint diffusion models in continual learning,” *arXiv preprint arXiv:2411.08224*, 2024.

[225] J. He and F. Zhu, “Unsupervised continual learning via pseudo labels,” in *International Workshop on Continual Semi-Supervised Learning*. Springer, 2021, pp. 15–32.

[226] A. M. N. Taufique, C. S. Jahan, and A. Savakis, “Unsupervised continual learning for gradually varying domains,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3740–3750.

[227] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

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

[229] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[230] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744

[231] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[232] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2761–2773, 2010.

[233] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003, pp. 267–273.

[234] L. Yao and G.-F. Lu, "Multi-view clustering indicator learning with scaled similarity," *Pattern Analysis and Applications*, vol. 26, no. 3, pp. 1395–1406, 2023.

[235] W. He, Z. Zhang, Y. Chen, and J. Wen, "Structured anchor-inferred graph learning for universal incomplete multi-view clustering," *World Wide Web*, vol. 26, no. 1, pp. 375–399, 2023.

[236] T. Liu, J. Zhu, J. Zhou, Y. Zhu, and X. Zhu, "Initialization-similarity clustering algorithm," *Multimedia Tools and Applications*, vol. 78, pp. 33 279–33 296, 2019.

[237] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *25th AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, 2011, pp. 313–318.

[238] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization," in *Twenty-first International Joint Conference on Artificial Intelligence*, vol. 9, 2009, pp. 1010–1015.

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

[240] M. Sadeghi and N. Armanfard, "Deep successive subspace learning for data clustering," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[241] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[242] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[243] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[244] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv: 1412.6980*, 2014.

[245] J. Cai, J. Fan, W. Guo, S. Wang, Y. Zhang, and Z. Zhang, "Efficient deep embedded subspace clustering," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 21–30.

[246] Y. Tao, K. Takagi, and K. Nakata, "Clustering-friendly representation learning via instance discrimination and feature decorrelation," *International Conference on Learning Representations (ICLR)*, 2021.

[247] T. W. Tsai, C. Li, and J. Zhu, "Mice: Mixture of contrastive experts for unsupervised image clustering," in *International Conference on Learning Representations*, 2020.

[248] Y. Le and X. S. Yang, "Tiny imagenet visual recognition challenge," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:16664790

[249] S. Zhou, H. Xu, Z. Zheng, J. Chen, Z. Li, J. Bu, J. Wu, X. Wang, W. Zhu, and M. Ester, "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," *ArXiv*, vol. abs/2206.07579, 2022.

[250] L. Zelnik-manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2004/file/40173ea48d9567f1f393b20c855bb40b-Paper.pdf

[251] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," *Pattern Recognit.*, vol. 10, pp. 105–112, 1978.

[252] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, p. 1010–1015.

[253] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper/2006/file/5da713a690c067105aeb2fae32403405-Paper.pdf

[254] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. 110, pp. 3371–3408, 2010. [Online]. Available: http://jmlr.org/papers/v11/vincent10a.html

[255] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[256] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2528–2535.

[257] P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers, “Associative deep clustering: Training a classification network with no labels,” in *Pattern Recognition*. Springer International Publishing, 2019, pp. 18–32.

[258] X. Ji, J. F. Henriques, and A. Vedaldi, “Invariant information clustering for unsupervised image classification and segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.

[259] C. Niu, J. Zhang, G. Wang, and J. Liang, “Gatcluster: Self-supervised gaussian-attention network for image clustering,” in *European Conference on Computer Vision (ECCV)*, 2020.

[260] Z. Lin, Y. Wang, and H. Lin, “Continual contrastive learning for image classification,” in *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[261] J. Smith, C. Taylor, S. Baer, and C. Dovrolis, “Unsupervised progressive learning and the stam architecture,” *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2979–2987, 2021.

[262] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for mobilenetv3,” *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:146808333

[263] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[264] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[265] A. Alkhulaifi, F. Alsahli, and I. Ahmad, “Knowledge distillation in deep learning and its applications,” *PeerJ Computer Science*, vol. 7, p. e474, 2021.

[266] P. Bholowalia and A. Kumar, “Ebk-means: A clustering technique based on elbow method and k-means in wsn,” *International Journal of Computer Applications*, vol. 105, no. 9, 2014.