DETECTING 2D AND 3D CLUSTERS THROUGH FEATURE EXTRACTIONS IN DEEP CONVOLUTIONAL NEURAL NETWORKS

DEVIN SCHAFTHUIZEN
Bachelor of Science, University of Lethbridge, 2024

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

DETECTING 2D AND 3D CLUSTERS THROUGH FEATURE EXTRACTIONS IN DEEP CONVOLUTIONAL NEURAL NETWORKS

DEVIN SCHAFTHUIZEN

Date of Defence: April 16, 2026

Dr. John Zhang Thesis Supervisor	Assoc. Professor	Ph.D.
Dr. Wendy Osborn Thesis Examination Committee Member	Assoc. Professor	Ph.D.
Dr. Yllias Chali Thesis Examination Committee Member	Professor	Ph.D.
Dr. Andrew Fiori Chair Thesis Examination	Assoc. Professor	Ph.D.

Dedication

This thesis is dedicated to the three

Most important groups of people in my life

My parents: Edward and Robin

My siblings: Brett and Steph

And my wife: Ami-Marie

You're everything

Abstract

Clustering data is a complex and computationally expensive task, and in some approaches, it requires multiple passes over the same data to ensure globally optimal results. Clustering is an unsupervised learning method, meaning the distinction between points is made solely from the dataset being explored, with no prior knowledge or examples from previously clustered data. Using previously analyzed work to guide future analysis is supervised learning, which is typically applied to other problem categories, such as classification.

Our research investigates the application of supervised learning techniques, specifically Convolutional Neural Networks (CNN), to perform clustering on projections of datasets in 2D and 3D visual representations using graphs and voxelization representations of data. These CNN models are designed for categorical output and can be used to guide the training process and leverage previously clustered data to learn representations in new, previously unseen datasets. However, that categorical output can only represent the number of clusters present. To extend this approach further, we explore extracting information from the CNN's processing layers to analyze the activation maps between the convolutional layers using our proposed SilhouetteGen algorithm to delineate cluster shapes and locations within the original input space. In later models, our algorithm also replaces the CNN's categorical output after training is complete to remove any restrictions on the prediction range.

Various benchmark datasets and cluster quality metrics are used to assess the feasibility of this approach relative to a widely used and well-researched clustering method. The primary goal of this analysis is to demonstrate the feasibility of deep CNN feature extraction for detecting cluster information in distance and density-based clustering problems without requiring individual point-wise distance calculations.

Acknowledgments

First and foremost, I extend my gratitude to my supervisor, Dr. John Z. Zhang. Your support and encouragement throughout this research and all my time studying at the University of Lethbridge have been invaluable, and it has been a pleasure to work with you.

I would like to thank the members of my thesis examination Committee. Dr. Wendy Osborn and Dr. Yllias Chali for their feedback on the final examination of this research and its proposal. I would also like to extend my thanks to Dr. Andrew Fiori, for his work chairing the examination committee.

I am also grateful to the ITS departments at both the University of Lethbridge and Lethbridge Polytechnic, who have facilitated and encouraged my studies and provided opportunities to work with and learn from their talented teams. This professional growth is also a valuable asset that I do not take for granted.

Lastly, I would like to thank my family, to whom this work is dedicated, for their love, support and encouragement in all aspects of life, including my research.

Contents

Dedication iii

Abstract iv

Acknowledgments v

List of Tables viii

List of Figures ix

List of Algorithms x

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Research Objectives and Contributions . . . 3
- 1.3 Overview of Methodology . . . 4
- 1.4 Thesis Outline . . . 5

2 Background 6
- 2.1 Clustering . . . 6
  - 2.1.1 Distance-Based Clustering . . . 8
  - 2.1.2 Density-Based Clustering . . . 12
  - 2.1.3 Semi-Supervised Clustering . . . 14
  - 2.1.4 Quantum Clustering . . . 15
  - 2.1.5 Spectral Clustering . . . 17
  - 2.1.6 Cluster Quality . . . 18
- 2.2 Recent Work in Clustering . . . 20
- 2.3 Neural-based Machine Learning . . . 23
  - 2.3.1 Artificial Neural Networks . . . 24
  - 2.3.2 Convolutional Neural Networks . . . 26
  - 2.3.3 Pretrained Networks . . . 27
  - 2.3.4 Image Segmentation . . . 27
- 2.4 Related Works . . . 29
  - 2.4.1 Neural-based Clustering . . . 29

3 Classifying 2D Distance-Based Clusters 31
- 3.1 Introduction . . . 31
- 3.2 Problem Statement . . . 32

3.3 Our Approach . . . 33
3.3.1 Architecture . . . 34
3.3.2 Extracting Feature Maps . . . 35
3.4 Experiments . . . 37
3.4.1 Software and Libraries . . . 37
3.4.2 Data Collection . . . 38
3.4.3 Training . . . 38
3.5 Results and Discussion . . . 39
3.6 Summary . . . 41

4 Classifying 2D Density-Based Clusters 42
4.1 Introduction . . . 43
4.2 Problem Statement . . . 44
4.3 Our Approach . . . 45
4.3.1 Architecture Changes . . . 46
4.4 Experiments . . . 48
4.4.1 Datasets . . . 48
4.4.2 Evaluation . . . 49
4.5 Results and Discussions . . . 49
4.5.1 Density Benchmark Dataset . . . 51
4.6 Summary . . . 55

5 Classifying 3D Distance- and Density-Based Clusters 57
5.1 Introduction . . . 57
5.2 Problem Statement . . . 58
5.3 Our Approach . . . 59
5.3.1 Architecture . . . 61
5.4 Experiments . . . 62
5.4.1 Datasets . . . 63
5.4.2 Training . . . 63
5.5 Results and Discussions . . . 64
5.6 Summary . . . 69

6 Conclusion 71
6.1 Results and Discussions . . . 71
6.2 Summary of Contributions . . . 72
6.3 Future Directions . . . 74

Bibliography 76

A Appendix 83

List of Tables

3.1 Generated Silhouettes on Image Example in Figure 3.2(a). 40

4.1 Nested Rings Results From Figure 4.3. 50
4.2 Quality Comparisons on Figure 4.1 and Figure 4.3. 51
4.3 Quality Comparisons on Figures 4.4, 4.5, 4.6, 4.7, and 4.8. 55

5.1 Quality Comparisons on Figures 5.2, 5.3, 5.5, 5.6, and 5.7. 68

A.1 Model Summary of our proposed CNN (Chapter 3). 83
A.2 Model Summary of the trimmed CNN (Chapter 4). 83
A.3 Model Summary of our proposed CNN - Conv3D (Chapter 5). 85
A.4 Model Summary of the trimmed CNN - Conv3D (Chapter 5). 85

List of Figures

2.1 Example of Cluster Types (Adapted from [21]). 7
2.2 Example of classical K-means vs Kernel K-means (Adapted from [21]). . . 11
2.3 Density-reachability and density-connectivity in DBSCAN. (Adapted from [21]). 13
2.4 Architecture of a single neuron (Adapted from [2]). 24
2.5 Example Artificial Neural Network (Adapted from [69]). 25
2.6 Example Convolutional Neural Network (Adapted from [21]). 26
2.7 Example of segmentation types (Adapted from [14]). 28

3.1 The Architecture of our CNN-based Model. 34
3.2 Image Augmentation and Cluster Silhouettes. 39
3.3 Non-Spherical Cluster Silhouettes Produced. 40

4.1 Demonstration of Non-convex Clusters Issues. 46
4.2 Trimmed architecture of our 2D CNN-based Model. 48
4.3 Demonstration of Improved Clusters Nested Rings. 50
4.4 Our Method Compared to DBSCAN - Target. 52
4.5 Our Method Compared to DBSCAN - Wingnut. 52
4.6 Our Method Compared to DBSCAN - TwoDiamonds. 53
4.7 Our Method Compared to DBSCAN - Sparse Clusters. 53
4.8 Our Method Compared to DBSCAN - Varying Density. 54

5.1 Trimmed Architecture of our 3D CNN-based Model. 62
5.2 Our Method Compared to DBSCAN - Hepta. 64
5.3 Our Method Compared to DBSCAN - Atom. 65
5.4 Activation map representation at various depths on the Atom dataset. 66
5.5 Our Method Compared to DBSCAN - Tetra. 66
5.6 Our Method Compared to DBSCAN - Chainlink. 67
5.7 Our Method Compared to DBSCAN - Chainlink Ext. 68

A.1 Full architecture of our 3D CNN-based Model used for training (Chapter 5). 87
A.2 Activation map representation on the Tetra dataset (Chapter 5). 87
A.3 Activation map representation on the Hepta dataset (Chapter 5). 87
A.4 Activation map representation on the Chainlink dataset (Chapter 5). 87
A.5 Activation map representation on the Chainlink Ext dataset (Chapter 5). . . 87

List of Algorithms

3.1 SilhouetteGen 35
3.2 Generate Convex Hulls 36

4.1 SilhouetteGen (Density) 47

5.1 Point Cloud to Voxel Grid 60
5.2 SilhouetteGen (3D) 61

A.1 Flood Fill Using Stack (Chapter 3) 84
A.2 Flood Fill 3D Using Stack in (Chapter 5) 86

Chapter 1

Introduction

Computers have evolved beyond being merely helpful research tools and are now a necessary infrastructure for collecting, interpreting, and visualizing data, as well as for running models that evaluate data, used across industrial, academic, governmental and research activities alike. As the volume and complexity of the available data have increased, computational methods have become essential for extracting useful information from large datasets [44]. Two key concepts that support this process are *machine learning* (ML) and *data mining*. Machine learning focuses on developing algorithms that can learn patterns from data, while data mining involves applying these computational techniques to large collections of data to discover meaningful structures and relationships [21]. These computational models help us identify trends and patterns in data that are not immediately apparent from the raw data points alone. Identifying patterns in data is always beneficial, as it enables us to tailor our actions and put researchers and analysts in an advantageous position in activities such as advertising, business deployment, and product stocking.

Clustering is the task of organizing similar objects into groups (clusters) using machine learning algorithms [68]. The task of detecting clusters within a dataset has traditionally been *unsupervised*, i.e., it handles data points that *do not* have predefined (group) *labels* and groups them based only on their similarities [24]. A clustering algorithm must draw conclusions from only the data to be explored. However, the exploratory nature of traditional clustering methods introduces limitations, not only due to the algorithm's computational complexity but also because some approaches are highly sensitive to input parameters [39],

and some require the number of clusters as input [34]. For instance, in distance-based clustering, we must specify the number of clusters to discover, whereas in density-based clustering, we must specify the radius and the threshold for the number of points in a cluster. Sensitive input parameters pose a dilemma: while we seek groupings based solely on the data, we must specify these values to do so, which could require prerequisite knowledge of the dataset and how groups are represented. The details of these algorithms and their restrictions are discussed further in Section 2.1 and throughout our work as we compare both distance- and density-based clustering problems.

1.1 Motivation

Humans and computers are naturally suited to different categories of problems. Computers can calculate large mathematical equations with ease, but the use of computational algorithms makes other tasks, such as object recognition in an image, difficult for computers [2]. *Artificial neural networks* (ANN) are a type of data processing model that aims to simulate the information processing of a human brain on a computer to predict complex data patterns [53]. Furthermore, a specific type of ANN, *Convolutional Neural Networks* (CNN), was developed, inspired by the human visual system, and is considered the state-of-the-art approach to object detection and other tasks, combining feature extraction and classification abilities [54]. This research proposes a novel approach to clustering that leverages advancements in data processing with CNNs. It aims to address some limitations of traditional clustering algorithms by applying a neural-based processing strategy and approaching data clustering from a visualization perspective.

Two essential ideas motivate the use of visualization tools for clustering. Firstly, the idea that humans, even in early childhood, learn to distinguish between different objects or animals, such as cats or dogs, and continuously improve subconscious cluster schemes [21]. This suggests that past information can be used to predict future objects. This is the fundamental idea of supervised learning, which learns to distinguish objects, or clusters

in this case, by example rather than rediscovering these representations on a case-by-case basis. The second motivation is that humans can visually judge the quality of clusters in up to three dimensions [21]. That is, we can inspect visual representations of low-dimensional datasets and quickly identify trends or groups within them. The goal here is to remove human intervention from that determination and offload that cognitive task to a neural network.

Research in traditional centroid-based unsupervised data clustering techniques investigates additions and optimizations to existing models to mitigate the risk of poor initial centroid placement and address the problem's complexity [5]. However, these popular unsupervised clustering approaches still have limitations that can affect the usefulness of the results, which may be unavoidable given their unsupervised nature.

1.2 Research Objectives and Contributions

This research aims to directly address some common limitations of existing unsupervised approaches to data clustering. The goal is to investigate and assess the feasibility and potential of applying supervised learning techniques to address this classically unsupervised problem. The use of supervised machine learning models for clustering has been previously explored. These initial efforts addressed some limitations of classical unsupervised approaches, such as the requirement for a user-defined number of clusters [51] by having an output range to determine the number of clusters from a predefined output. However, they introduced other limitations and consistency issues [25]. These models are further discussed in Section 2.4.1. The significance of this research lies in understanding how these models can be improved through recent advances in machine learning, particularly by applying more advanced neural network architectures and examining how the rich feature-extraction capabilities of CNNs can be applied to clustering.

Our contributions from this research aim to develop a generalized form of transferable knowledge by creating a neural network whose weights can be saved and exported for

reuse. When neural networks learn generalizable trends in datasets, they can be applied to new datasets, allowing previous trends to guide decisions and predictions on new data [2]. The goal is to have a feature-extraction process pull information from a network that does not require retraining when processing new datasets, thereby using previously observed information to detect and identify clusters in future datasets.

Specific Research Objectives (RO) to support our contributions are as follows:

- RO1: Determine whether CNNs can be used for clustering problems through visualizing datasets. Specifically, if we can extract information from the hidden layers of a CNN to determine cluster information from a visual representation of data, rather than making point-wise distance comparisons.
- RO2: Determine whether we can avoid common limitations of existing clustering algorithms, such as in some cases needing to specify the number of clusters to create, poor centroid initialization and parameter sensitivity.
- RO3: Determine the flexibility of a CNN-based approach in detecting clusters of arbitrary and complex shapes.
- RO4: Determine if a CNN-based clustering approach faces the same restrictions that limited the usefulness of previous neural-based approaches to clustering tabular data.

1.3 Overview of Methodology

This research adopts a phased experimental methodology consisting of iterative planning, implementation, and evaluation stages. During each phase, we focus on developing and assessing our proposed SilhouetteGen clustering algorithm, with specific goals we adapt to solve, thereby enabling incremental refinement of the model and experimental design. Synthetic datasets are used during training to provide controlled environments in which specific cluster characteristics, such as cluster density, separation, and noise, can be controlled. These datasets enable a targeted evaluation of the model's behaviour under

known conditions. When available, established benchmark datasets from existing literature are also incorporated to validate our approach and facilitate comparisons with one of the most cited and widely used density-based clustering algorithms in the scientific community, DBSCAN [20]. This phased methodology enables insights from each stage to inform subsequent experimental decisions, thereby supporting iterative model improvement while maintaining flexibility to address challenges that may arise during development.

1.4 Thesis Outline

This thesis comprises six chapters, each with the following structure and content.

Chapter 2 introduces the background concepts on which this study is based. This chapter also reviews current approaches to data clustering and recent publications on a range of clustering types.

Chapter 3 is a summary of a refereed conference paper on our initial prototype of this approach [58]. This chapter examines clustering data containing simple spherical-shaped patterns in two-dimensional space using a visual representation of the dataset to determine the number of clusters, their positions, and outline their shapes.

Chapter 4 extends the capabilities of our initial model by adapting to more complex data patterns, which are typically associated with density-based clustering approaches. By evaluating the previous approach, we simplify and refine our processes to make them more adaptable to clusters of arbitrary shapes. When examining our model, we also make comparisons to the popular density-based algorithm DBSCAN.

Chapter 5 reinvents our approach using a new input and data-encoding technique to perform clustering with CNNs in volumetric three-dimensional space, rather than in two-dimensional images. This allows us to further extend our neural-based approach to existing CNN architectures and, again, increase the complexity of the data that can be interpreted.

Finally, in Chapter 6, we conclude the research by summarizing our findings and offering recommendations for future research on this topic.

Chapter 2

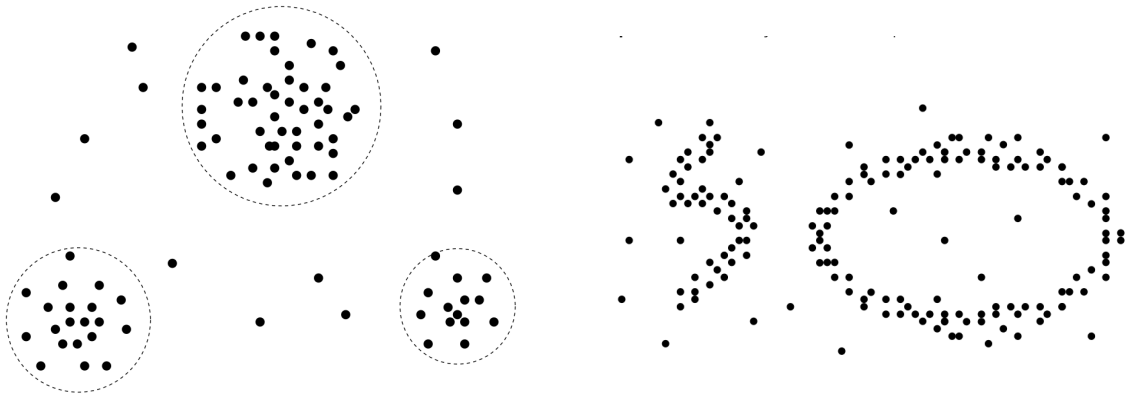
Background

Machine learning forms the technical foundation of data mining, enabling the extraction of meaningful information from raw data [68]. By identifying patterns, structures, and relationships within collected data, ML supports informed decision-making across a wide range of domains. The choice of learning approach depends on both the nature of the available data and the specific question being asked [2, 21]. For instance, regression techniques are used to predict continuous outcomes, such as price or stock pricing changes, whereas classification techniques assign a predefined category to new observations, such as identifying spam emails or fraudulent transactions [21].

Our work focuses on clustering unlabelled data. Accordingly, this chapter reviews traditional clustering methods, followed by neural-based processing and deep learning techniques, which serve as the primary tools for addressing this problem. This chapter also includes a brief overview of image segmentation, a computer vision strategy for partitioning images into meaningful regions for identification and labelling [4]. Although segmentation is typically applied to photographs, its underlying principles of grouping similar features and boundary detection are similar to the objectives of clustering and provide useful conceptual insights for our work.

2.1 Clustering

Clustering data is the process of partitioning data points or objects within a dataset into different groups (clusters) by quantitatively comparing the characteristics of the data points



(a) Distance-Based Example (b) Density-Based Example

Figure 2.1: Example of Cluster Types (Adapted from [21]).

so that the points within the same group have high similarities compared to data points from another cluster [21]. Clustering has been used to address analysis problems in various domains such as biology, medical science, sales, marketing, manufacturing, security, privacy protection, and the financial sector, to name a few [24, 41, 43, 57, 62]. The two main categories of clustering algorithms are hierarchical and partition-based. Hierarchical clustering can be performed using a top-down or bottom-up approach to generate hierarchical structures, producing nested clusters that may overlap. A hierarchical approach produces a dendrogram that represents the nested patterns and similarities in the dataset at different levels. In contrast, partition-based clustering algorithms find non-overlapping splits and group points simultaneously as the dataset is explored [21, 25]. The most popular and commonly used algorithms for the partitioning of datasets into clusters are K-means and K-medoids [5, 21, 24, 43, 47]. While these are classical approaches that have been in use for decades, additional comparisons and optimizations for these approaches have been presented in recent work on clustering [5, 24].

Beyond hierarchical and partition-based methods, clustering research has expanded to include sub-categories such as density-based, semi-supervised, and hybrid approaches designed to address complex data characteristics. Illustrations of distance-based clusters and density examples are shown in Figure 2.1. While hierarchical clustering provides a mul-

tilevel view of data structure, our work is primarily related to partition-based and density-based methods.

2.1.1 Distance-Based Clustering

K-means

K-means is a distance-based data clustering algorithm that partitions datasets into *K* groups, where *K* is a user-defined parameter. Lloyd proposed this algorithm [37], which works by initializing the centre points of *K* clusters, typically selected on a uniform random distribution of the dataset, and then iterating over two steps. In an assignment step, each data point is assigned to the nearest cluster centre to minimize intra-cluster variance. The nearest centre is determined by calculating the distance from each point to all *K* cluster centre points. Then, in an update step, the cluster centres are recalculated by averaging the points assigned to each cluster. These steps are repeated until a convergence criterion is met, meaning that cluster centres stabilize or no significant updates occur [24, 25]. *K*-means has been credited for its flexibility, efficiency, and ease of implementation and is a widely used algorithm with several publications reporting improvements and practical applications [5, 24].

K-medoids

The *K*-medoids, or Partitioning Around Medoids (PAM), is a variant of the *K*-means algorithm and is less sensitive to outliers in the dataset. It was proposed by Kaufman and Housseeuw [24] and is another centroid-based algorithm, as each cluster is characterized by a centrally located object [28]. It requires a user-defined parameter *K*, indicating the number of partitions the algorithm should create in the dataset, and begins by selecting random data points as centres (called medoids). The remaining data points are then assigned to the medoids based on distance calculations to create the groupings, which are updated after all points have been assigned in two distinct steps, BUILD and SWAP. These steps are functionally equivalent to the assignment and update steps of the *K*-means algorithm.

However, the objective function in the update (or BUILD) step aims to minimize the sum of all distances within a group to its medoid [28, 48]. As in the K-means algorithm, this process will repeat and continue to update the medoids until the algorithm stabilizes.

While both algorithms share a similar iterative approach, K-medoids differs from K-means in that cluster centres are restricted to actual data points, whereas that is only the case for the initial centroid selection in K-means. By continuing to use data points as possible centres, K-medoids offers greater robustness to outliers at the cost of increased computational complexity [28].

Improved Initialization

One trend to overcome the challenges of these centroid-based models is to use a heuristic to determine a better initial selection of the centroid [21]. Improving the initial placement of the K centroids reduces the number of iterations required by the algorithm, accelerating convergence and improving computational efficiency. This motivated the development of new approaches, such as K-means++ [5] which selects the first centroid uniformly at random and chooses each subsequent centroid from the remaining data points with probability proportional to the squared distance from the nearest already-selected centroid, promoting well-separated initial centroids. A more recent example that aims to overcome the same issue is the enhanced K-means algorithm [27] that improves the placement of the initial centroids by splitting the dataset using a percentile method, ensuring that the groups start roughly equal sized to avoid the dominance of any one dense region that causes other centroids to become trapped in a local minimum.

Fuzzy Clustering

Another type of centroid-based clustering algorithm related to K-means is fuzzy c-means (FCM), introduced by Bezdek [10]. Unlike K-means, which assigns each data point exclusively to a single cluster, FCM produces fuzzy partitions in which each data point is associated with all clusters through degrees of membership ranging between 0 and 1.

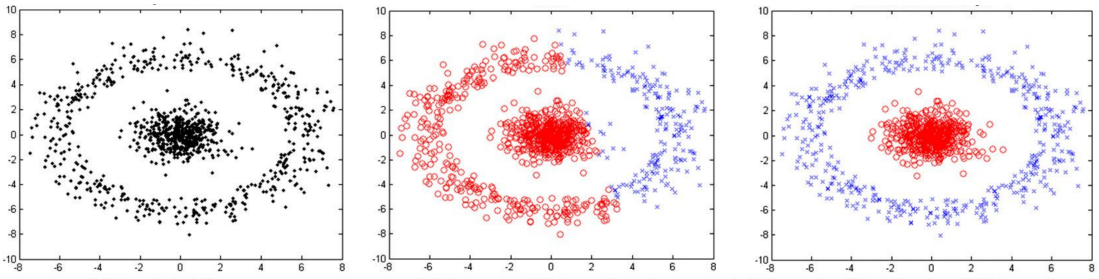
This approach, commonly referred to as soft clustering, is particularly useful in applications where cluster boundaries are not well defined. Over time, extensions of FCM have been proposed to address its sensitivity to noise and outliers, a common occurrence in real-world data. These include fuzzy-possibilistic and other hybrid approaches that relax strict membership constraints and make FCM more robust [45]. While such extensions enhance clustering flexibility and noise tolerance, they also introduce additional parameters that strongly influence clustering outcomes. As a result, research has focused on parameter selection, stability, and interpretability in generalized fuzzy c-means approaches, highlighting the sensitivity of some of these approaches [46].

Scalability

Another avenue of research on clustering focuses on the scalability of clustering methods, ensuring that these approaches remain effective even with large datasets. Two broad strategies for handling large-scale and high-dimensional data are feature and dimensionality reduction, as well as subspace clustering [39]. Feature reduction is important because it reduces the number of attributes required to accurately group points, through the use of feature selection, feature extraction, and projection techniques. These preprocessing steps help eliminate redundant or irrelevant information, reduce noise, and improve both computational efficiency and clustering quality. For example, studies in high-dimensional single-cell and biomedical data have shown that applying principal component analysis (PCA) or random projection before clustering can preserve or even improve cluster separability while significantly lowering computation cost and memory usage [1, 57]. Subspace clustering, in contrast, draws small random samples from the original dataset to determine how cluster centroids are updated. An example is the mini-batch K-means algorithm, which uses small, randomly sampled subsets of the dataset. During each iteration, data points in the mini-batch are assigned to the nearest centroid, which is then updated incrementally. By operating on subsets of the data, this algorithm reduces computational cost and mem-

ory usage while producing an approximate solution comparable to standard K-means for large-scale datasets [39, 52].

Kernel K-means



(a) Original dataset (b) Results from k-means (c) Kernel k-means

Figure 2.2: Example of classical K-means vs Kernel K-means (Adapted from [21]).

In many real-world applications, clusters are not linearly separable, and applying the standard K-means algorithm may produce poor results because clusters are defined solely by distances to centroids in the original feature space. Kernel K-means extends the traditional algorithm to address this limitation by enabling the discovery of non-linearly separable clusters [21, 50]. The key idea behind kernel K-means is to map data points from the original input space into a higher-dimensional feature space, where clusters that are inseparable in the original space may become separable. Rather than computing this transformation explicitly, the algorithm uses a kernel function to compute similarities between points in the transformed space, a technique commonly referred to as the kernel trick [17, 71].

Common kernel functions include the Gaussian Radial Basis Function (RBF) kernel and polynomial kernels. This approach allows kernel K-means to correctly identify complex cluster structures, such as concentric rings or nested groups, that standard K-means fails to separate. For example, a dataset containing a dense inner cluster surrounded by an outer ring cannot be partitioned correctly using centroid-based distances in the original space, but becomes separable after kernel transformation, as illustrated in 2.2. Despite its flexibility, kernel K-means introduces additional challenges to overcome, such as selecting

an appropriate kernel function and determining its parameters, which is critical to performance. Highly complex datasets may also require multiple kernel approaches to adequately separate the data [36, 71].

2.1.2 Density-Based Clustering

In addition to partitioning and hierarchical clustering approaches, density-based clustering methods identify clusters as contiguous regions of high point density separated by regions of low density [20]. Unlike centroid-based methods, density-based approaches can discover clusters of arbitrary shapes and are robust to noise. These methods assume that clusters correspond to dense regions in the data space, whereas less densely populated regions correspond to cluster boundaries or outliers in the data [21]. A prominent algorithm in this category is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), proposed by Ester et al. [13]. Density-based methods are particularly effective for spatial data and applications where noise detection is important.

DBSCAN

DBSCAN defines clusters based on the density of data points within a specified neighbourhood. Two key parameters determine this density, ε which represents the radius defining a point's neighbourhood and *MinPts*, which is the minimum number of points required to form a dense region [13]. Based on these parameters, DBSCAN classifies points into three types:

- Core points: points with at least *MinPts* neighbours within ε.
- Border points: points within ε of a core point but with fewer than *MinPts* neighbours.
- Noise points: points that are neither core points nor border points.

Clusters are formed by connecting the core points and their reachable neighbours. A major advantage of DBSCAN is its ability to detect arbitrary shape clusters while simultaneously

identifying noise and outliers [21]. However, its performance can be sensitive to parameter selection and may struggle with datasets containing clusters of varying densities [66].

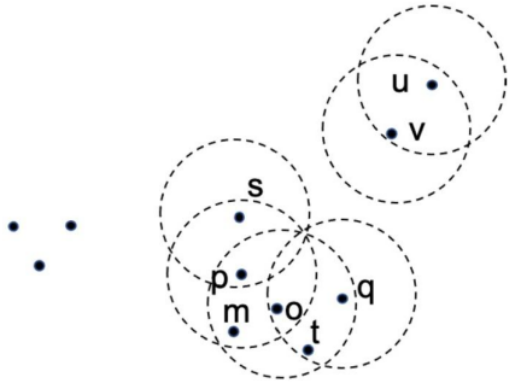


Figure 2.3: Density-reachability and density-connectivity in DBSCAN. (Adapted from [21]).

An example DBSCAN searching and labelling points is shown in Figure 2.3 and is adapted from Han [21]. In this representation, the dotted circles represent the range ε around each point, being the area in which the MinPts (or minimum neighbours) need to be discovered in order to be core points. If the MinPts value is configured to three, then points *p*, *m*, *o*, and *t* are labelled as core points, since they have enough neighbouring points within the specified ε range. The points *s* and *q* do not satisfy the MinPts requirements, but are within the ε range of core points, and are therefore labelled as border points. The points *u* and *v* do not meet the MinPts value and are not within ε of a core point, and are then labelled as noise. This process would then continue to other points in the dataset, such as the three yet to be defined points on the left in the example in Figure 2.3, until all points in the dataset have been identified and labelled as either core points, border points or noise points.

Grid-based Models

Grid-based clustering methods are a variation of density-based clustering that partitions the data space into a finite number of cells that form a grid structure. Instead of operating on individual data points, clustering is performed on grid cells, thereby significantly reduc-

ing computational complexity. One well-known grid-based algorithm is STING (Statistical Information Grid) proposed by Wang et al. [67], which divides the spatial area into hierarchical rectangular cells. Statistical information (e.g., mean, variance, density) is stored for each cell, and clustering decisions are made by evaluating these summaries [47]. Because operations are performed on cells rather than individual data points, grid-based methods are highly scalable and efficient for large datasets [63]. The main advantage of grid-based approaches is their fast processing time, which is typically independent of the number of data points. However, grid-based methods are sensitive to grid resolution. If the grid is too large, important cluster structures may be lost; too fine and the computational efficiency will decrease [21].

2.1.3 Semi-Supervised Clustering

Semi-supervised learning is a machine learning approach that leverages a small amount of labelled data alongside a large amount of unlabelled data to guide the learning process. Unlike supervised learning, the objective is not to predict labels for new instances, but to improve the quality and interpretability of clusters by incorporating prior knowledge about relationships among data points. This approach is particularly valuable in real-world settings where unlabelled data is abundant and inexpensive to collect, while obtaining labelled data is costly and time-consuming [7, 40]. As a result, datasets often contain far more unlabelled observations than labelled ones, motivating methods that can effectively use both. In semi-supervised clustering, prior knowledge is typically introduced through either seeded points or pairwise constraints. Seeded points have known cluster assignments that do not update, providing reference points that guide the updated steps from deviating from known positions [7]. While pairwise constraints, on the other hand, indicate whether two observations should belong to the same cluster (must-link) or to different clusters (cannot-link) [33]. By incorporating these forms of supervision, clustering algorithms can avoid implausible groupings and produce results that better reflect domain knowledge.

Constrained K-means is one such approach that uses labelled information to improve clustering by modifying the assignment step by enforcing pairwise constraints, ensuring that must-link and cannot-link constraints are followed while assigning points to clusters. Labelled points are first associated with their known clusters, after which unlabelled points are assigned while following specified constraints. In contrast, seeded K-means uses labelled data during centroid initialization, selecting initial cluster centres based on known examples. This strategy is useful for determining centroid initialization, which reduces the likelihood of convergence to poor local minima [7, 9]. Even limited supervision can therefore improve clustering stability and alignment with real-world categories. Semi-supervised learning has also been extended to density-based approaches such as DBSCAN. Semi-Supervised DBSCAN (SSDBSCAN), proposed by Lelis and Sander [33], incorporates labelled points while retaining DBSCAN's core concepts of density reachability and connectivity. SSDBSCAN uses labelled points as anchors that define cluster identities, and unlabelled points may be assigned to a cluster only if they are reachable from labelled points through dense regions, allowing clusters to expand through regions of sufficient density. At the same time, conflicting labels can prevent the merging of neighbouring dense regions, improving cluster separation.

These approaches extend unsupervised clustering approaches by incorporating limited supervision through a small number of samples. The integration of labelled and unlabelled data provides several advantages over purely unsupervised methods. By incorporating prior knowledge, semi-supervised clustering can improve accuracy and help distinguish adjacent dense regions that unsupervised methods might merge, with only a small amount of labelled data [7, 40].

2.1.4 Quantum Clustering

Machine learning approaches that are partially or fully transformed from classical algorithms into quantum algorithms in order to improve computational performance are referred

to as quantized [6]. Clustering has become a topic of interest because clustering algorithms, including K-means, require repeated distance calculations and iterative optimization, which can be computationally expensive for large or high-dimensional datasets. The quantization of classical K-means has led to several proposed algorithms, including Q-means and δK-means [29, 62]. Quantized clustering methods use a hybrid quantum–classical framework, where classical preprocessing prepares and normalizes the data, which is then encoded into quantum states. Quantum algorithms accelerate the computationally intensive step of determining distance estimation in the nearest-centroid search, while classical processing updates cluster assignments and centroids. This hybrid design reflects current hardware limitations, in which some tasks are still performed with classical computation while exploiting potential quantum advantages.

The Q-means algorithm is a quantization of the classical k-means approach that accelerates distance computations and cluster assignment. In classical K-means, distances between data points and centroids are computed explicitly. In Q-means, data points are encoded as quantum states, allowing distances to be estimated using quantum inner-product evaluation. A common technique for this is a quantum circuit that estimates the similarity between two quantum states known as a SWAP test. By preparing quantum states corresponding to a data point and a centroid, the SWAP test can estimate their inner product, which can then be used to derive Euclidean distance. This enables fast distance estimation for high-dimensional data [29]. Another quantum advantage arises in the cluster assignment step. Searching for the nearest centroid among K candidates is a core operation in k-means. Quantum search algorithms, such as Grover's search [19], can reduce the complexity of this step [6].

The δk-means algorithm extends quantum clustering by introducing a tolerance parameter that allows approximate distance evaluation. Where distances only need to fall within a specified threshold δ. This reduces quantum resource requirements and improves robustness to noise [62]. By trading exactness for efficiency, δK-means aims to make quantum

clustering more practical on noisy intermediate-scale quantum (NISQ) hardware.

Quantum clustering methods offer several potential advantages. They may provide computational speedups for distance calculations and search operations, particularly in high-dimensional spaces. Some studies report improved convergence and clustering accuracy in specific applications, such as intrusion detection in Internet of Things (IoT) networks [62]. However, the need to enhance the hybrid quantum-classical pipeline to reduce quantum-resource overhead and improve interpretability is recognized [62].

2.1.5 Spectral Clustering

As the number of dimensions in a dataset increases, the amount of data required to maintain a similar density grows exponentially. This phenomenon is known as the *curse of dimensionality* [35]. Feature selection and ranking can be used to reduce the number of dimensions in a dataset needed for clustering, but this may not always be an option or reduce the required feature space enough for meaningful changes. Transforming multidimensional data into a lower-dimensional representation can improve separability. Spectral clustering is a method for transforming the original dataset of objects into a similarity graph, where each point is a node and edges connect its close neighbours based on similarities in the original dataset. Once points are embedded in a low-dimensional space, clusters can be discovered more easily, and classical clustering algorithms, such as K-means, can be used to extract them [21]. These approaches excel on graph-structured and sparse datasets, or when high dimensionality makes other approaches less reliable.

Not all clustering algorithms belong exclusively to a single category. Some approaches use multiple clustering strategies to address specific problems, such as combining density-based and grid-based techniques to improve scalability while preserving the ability to detect arbitrarily shaped clusters [21]. Such hybrid approaches are increasingly used in large-scale data mining and spatial analysis and can incorporate many of the methods discussed in this chapter.

2.1.6 Cluster Quality

There are several approaches to measuring the quality of the clusters generated by the clustering algorithms, and the choice of metric depends on the characteristics of the data, the clustering algorithm, and the availability of ground-truth labels [39]. In general, evaluation methods are classified as intrinsic or extrinsic.

Intrinsic Methods

Intrinsic methods assess clustering quality by evaluating their compactness and separation using only the data points and their assigned groups, without requiring true class labels. The Davies–Bouldin Index (DBI) [12] was an early metric introduced to represent cluster-separation. DBI is widely used due to its simplicity and computational efficiency. However, it relies on distances to cluster centroids, making it less suitable for clustering methods that do not define centroids or that contain subcluster structures [39, 52]. Another intrinsic metric is the silhouette coefficient, defined as:

$$Silhouette = \frac{b_i - a_i}{max(a_i, b_i)}, \tag{2.1}$$

where a_i is the average distance between a point and other points in the same cluster, and b_i is the average distance between that point and points in the nearest neighbouring cluster. The silhouette coefficient evaluates both intra-cluster cohesion and inter-cluster separation; its results range from -1 to 1, where higher values indicate well-defined, clearly separated clusters [21].

Extrinsic Methods

Extrinsic methods evaluate clustering results by comparing them against known ground-truth labels. These methods are applicable when the true class membership of data points is available. Metrics such as homogeneity and completeness measure whether clusters contain only members of a single class and whether all members of a class are assigned to the same

cluster, respectively [21]. Two such extrinsic metrics are *precision*, which represents the proportion of points assigned to the same cluster that belong to the same true class, and *recall*, which represents the proportion of points from the same true class that are correctly grouped together. Precision and recall are calculated with the following formulas:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositives}, \tag{2.2}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegatives}. \tag{2.3}$$

A True Positive occurs when two points that belong to the same ground-truth cluster are also assigned to the same cluster. A False Positive occurs when two points from different ground-truth clusters are incorrectly placed in the same cluster. A False Negative occurs when two points that belong to the same ground-truth cluster are incorrectly separated into different predicted clusters. In clustering, outliers are assigned their own cluster identifier, typically -1. Meaning points falsely labelled as outliers will affect these measures. Another situation is a True Negative, which occurs when two points that belong to different ground-truth clusters are assigned to different clusters. These four statistics can also be presented as a confusion matrix, but this representation is not commonly used in clustering problems.

Precision and recall can be combined into a *F1-score* or (F-measure). That looks to balance the two metrics. The F-measure is calculated as:

$$FMeasure = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{2.4}$$

Another extrinsic metric is Cluster purity, which measures the extent to which all points of the same class are grouped into the same clusters; it is also sometimes referred to as completeness. For each cluster, the most frequent class label is identified, and the overall

purity is computed as:

$$Purity = \frac{1}{N}\sum_{j=1}^{k} max j(count(C_i \cap G_j)), \tag{2.5}$$

where N is the total number of data points, k is the number of clusters, C_i represents cluster i and G_j represents the true class labels. Higher purity values indicate more homogeneous clusters [39].

There are many other metrics that can be used to assess how well a dataset has been clustered. Some evaluation measures are specialized for particular clustering methods. For example, fuzzy clustering metrics evaluate the degree of membership of groups rather than distinctive assignments [30], and are not directly applicable to other clustering approaches that assign only a single group to data points.

For the evaluation of work, we will use extrinsic methods and include comparisons of precision, recall, F1-score and cluster purity. These are the most robust methods and are commonly used metrics when ground-truth labels are available.

2.2 Recent Work in Clustering

Given the various types of clustering methods, there is no single dominant direction in current research. Various categories are being studied and applied to a wide range of domains. Some focus on improving the underlying model, such as improved seeding for centroid-based models, and others on how to adapt datasets for analysis. These are a small sample of the recent works.

In 2022, Omar *et al.* [50] proposed an approach to improve the clustering performance of the K-means algorithm for non-linear data. In the first step, the hidden layer of a Radial Basis Function (RBF) network transforms low-dimensional input data into a higher-dimensional feature space. In the second step, standard K-means is applied to the transformed data. The model was then evaluated using simulated datasets that are not lin-

early separable and demonstrates improved clustering accuracy and effectiveness compared to standard Kernel K-Means approaches and, in some cases, outperforms DBSCAN. The approach is further validated on an image segmentation task [50].

In 2023, Miraftabzadeh *et al.* [43] preformed a topical review in IEEE Access examining K-means role in modern power systems. This publication notes a wide range of applications of clustering in power systems, including load forecasting, fault detection, power quality analysis and system security assessments. Additional clustering methods were also reviewed, including K-medoids, Time-series K-means, HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), and SOMs, among others. The review also notes the dominance of K-means and other centroid-based approaches within power system research, noting $1,929$ publications on K-means application in power system up until 2022 [43].

Salloum *et al.* [57] conducted a study of clustering medial transcriptions in 2024. This study converted textual data into a vector representation and applied PCA to reduce the high-dimensional dataset to a two-dimensional space. K-means was then used to group the data into five clusters. The author suggests that this approach could be used to automatically categorize medical documentation in a way that mirrors clinical evaluation [57].

Katyal and Sharma proposed Enhanced K-means in 2024 [27]. This research aims to improve the conventional K-means algorithm by introducing a new centroid seeding protocol and an improved update step. The new method initializes centroids systematically using a percentile method to ensure an even distribution at the beginning of the algorithm. Then, during each update iteration, an array stores the minimum distances from each object to its centroid. After the centroids have been repositioned, a point is only recalculated to other centroids if the distance to its previous centroid increases. The goal is to reduce the number of iterations and avoid issues caused by poor initialization [27].

Also in 2024, Pardede *et al.* [52] discussed the implementation of K-means, K-Medoids and Mini batch K-means. The work investigates the use of clustering techniques to identify

and group areas prone to natural disasters in Indonesia. Looking into issues such as earthquakes, floods, drought, forest fires, tsunamis, volcanic eruptions and landslides. Their data was sourced from the Indonesian Central Statistics Agency. The research evaluated clustering performance approaches using the DBI, concluding that all three could be used for this use case, but k-means showed the best DBI value [52].

Also in 2023, Long *et al.* [38] combined Q-means with spectral clustering to create the Quantum Regularized Spectral Clustering QRSC algorithm. This approach uses a set of matrices to regularize the input, aiming to avoid overfitting in sparse graphs. The data is then projected into a new low-dimensional space before processing it with the Q-means algorithm. This study used simulated benchmark datasets to confirm the effectiveness of their proposed approach and a simulated quantum backend, which is common in quantum research. Although full implementation on quantum hardware is not yet possible, the authors suggest that the framework establishes a theoretical foundation for future quantum clustering algorithms [38].

In 2025 Srinivasan *et al.* [62] proposed Quantum k-means (QK-means). Their approach uses Qiskit, a simulated quantum backend, along with classical computing to encode the required information in the quantum framework. Their method is applied to IoT devices and network security. The quantum circuit is used for distance computation. The model performs similarly to K-means in low-dimensional tests, but excels as the number of dimensions increases, and shows increased stability as the volume of data increases [62].

In 2025 proposed by Tyagi *et al.* [66] introduced a DBSCAN improvement called Feature Selection-DBSCAN (FS-DBSCAN), designed to improve clustering efficiency and accuracy compared to traditional DBSCAN. Their approach integrates feature selection with a dynamically learned weighted Euclidean distance metric, which is said to reduce irrelevant dimensions; however, the principles of density determinations in DBSCAN remain unchanged [66].

In 2026, Yasin *et al.* [72] proposed DBSCAN-Leak, an extension of DBSCAN for real-

time water leak detection. DBSCAN-Leak introduces a dynamic neighbourhood radius and a decision threshold for grouping points. The data points are partitioned by day and hour, allowing the algorithm to compare each observation with its relevant historical values. A point is classified as a leak only if its density exceeds a threshold. That threshold can be automatically adjusted based on the context history of previous observations. Unlike regular clustering, the output of this model is the outliers that exceed the threshold, which alert to a leak. Points within the regular threshold are of no interest. DBSCAN-Leak was tested on a large university campus smart water network and outperformed classical DBSCAN, achieving 100% precision, 98% accuracy, 80% recall, and an F1-score of 89% [72].

2.3 Neural-based Machine Learning

Traditional clustering approaches, such as K-means and K-medoids, separate data without any prior knowledge of how the data should be structured and group data points solely based on observed characteristics, typically using distance-based heuristics. This is known as unsupervised learning, where the number and meaning of labels are unknown. The structures in the data are discovered directly as the data are explored, and labels can then be assigned post hoc based on the identified groupings [25]. As discussed in Section 2.1.3, semi-supervised approaches extend this approach by incorporating a limited amount of labelled data to guide the grouping of unlabelled observations.

In contrast, supervised learning leverages labelled data and aims to learn a mapping from input features to known output classes. A model is trained on labelled examples and can then predict labels for previously unseen data [25]. *Artificial Neural Networks* (ANNs) are a prominent type of supervised learning models inspired by biological neural systems [53]. Although simplified, they capture key ideas from neuroscience, representing neurons as computational units and synapses as weighted connections that manipulate the data being processed.

2.3.1 Artificial Neural Networks

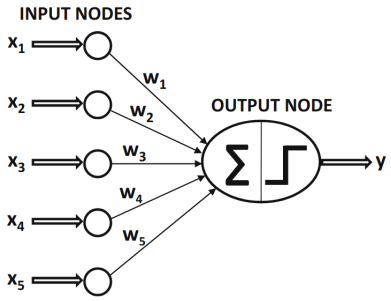


Figure 2.4: Architecture of a single neuron (Adapted from [2]).

At the core of an ANN is the artificial neuron, or node, which receives inputs either from raw data or from neurons in a previous layer. Each input is multiplied by an associated weight, and the weighted inputs are summed to produce a net input. An activation function is then applied to this value to determine the neuron's output. This process is illustrated in Figure 2.4. The output of a neuron can be expressed as:

$$z = \sum_{i=1}^{n} w_i x_i + b, \tag{2.6}$$

where x_i are inputs, w_i are weights, and b is an optional bias term. The activation function $\phi(z)$ produces the final output.

Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU), each with different output ranges, making them suitable for specific tasks. For example, sigmoid functions are often used in binary classification, while ReLU is widely used in hidden layers due to its computational efficiency and ability to mitigate the vanishing gradient issues [2].

Perceptron Model

The perceptron is one of the earliest neural models developed by Rosenblatt [55] and represents a single artificial neuron used for binary classification. It computes a weighted sum of inputs and applies a threshold activation function to produce a class label. While

simple, the perceptron established the foundation for learning weight parameters from data. However, single-layer perceptrons can only separate linearly separable data, limiting their applicability to more complex problems.

Multilayer and Deep Neural Networks

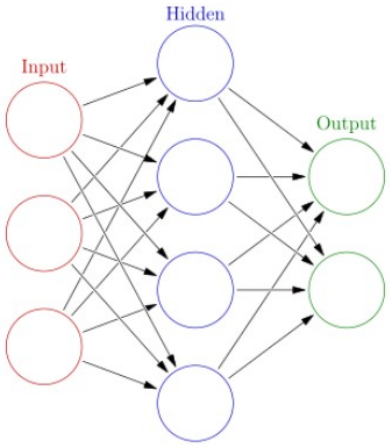


Figure 2.5: Example Artificial Neural Network (Adapted from [69]).

To overcome the limitations of single-layer models, multilayer neural networks introduce one or more hidden layers between the input and output layers, as shown in Figure 2.5. These hidden layers enable the network to learn more complex feature representations, allowing it to model nonlinear relationships in the data [2].

In classification tasks, the output layer contains one node per class, and the class associated with the highest activation is selected as the predicted label. Softmax activation is commonly used in the output layer for multi-class classification, as it converts outputs into a probability distribution over classes [2]. Training is performed using supervised learning, where the network's predictions are compared to true labels, and the weights are iteratively adjusted to minimize prediction error [53, 69, 51]. This process enables neural networks to learn complex patterns.

The introduction of multiple hidden layers leads to deep neural networks, which have demonstrated significant success in tasks such as image recognition, natural language processing, and anomaly detection. Depth improves the network's ability to extract high-level

features, making deep architectures substantially more powerful than early single-layer models [2].

2.3.2 Convolutional Neural Networks

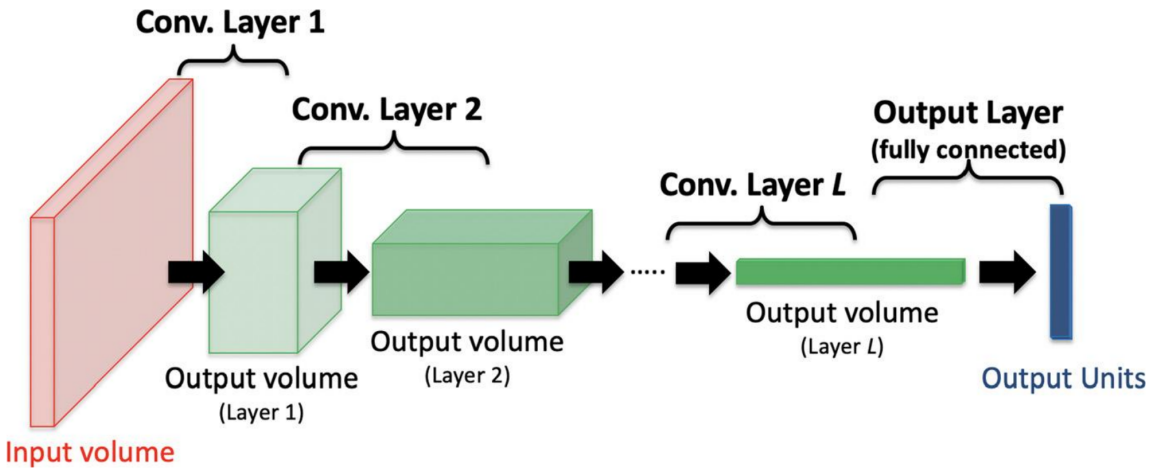


Figure 2.6: Example Convolutional Neural Network (Adapted from [21]).

Convolutional Neural Networks (CNNs) are a specialized class of ANNs designed to process structured grid-like data such as images. Early work by Fukushima introduced the Neocognitron, a hierarchical model capable of learning spatial features [15]. Modern CNNs extend this idea by learning these representations through stacked convolutional layers, enabling robust feature extraction from high-dimensional inputs [2]. Unlike traditional ANNs that operate on one-dimensional feature vectors, CNNs accept two or three-dimensional inputs, typically representing image height, width, and colour channels. Convolutional layers apply filters that slide across the input to detect local patterns such as edges, textures, and shapes. The result of each convolution is a feature map that highlights the presence of learned patterns in specific spatial locations [2, 54]. These feature maps are passed through nonlinear activation functions, similar to those in their base ANN models, allowing the network to model complex relationships in the data. Additional layer types are used with convolutional layers to improve performance and stability. Pooling layers reduce spatial resolution while preserving important features, improving computational efficiency. Batch

normalization layers stabilize training by normalizing intermediate activations, allowing deeper architectures to converge more reliably [2, 21]

As the network depth increases, CNNs learn increasingly abstract representations. Early layers detect simple features such as edges, while deeper layers capture high-level semantic structures. Eventually, the feature maps are flattened into a one-dimensional vector and passed to fully connected layers that perform classification. This architecture is illustrated in Figure 2.6.

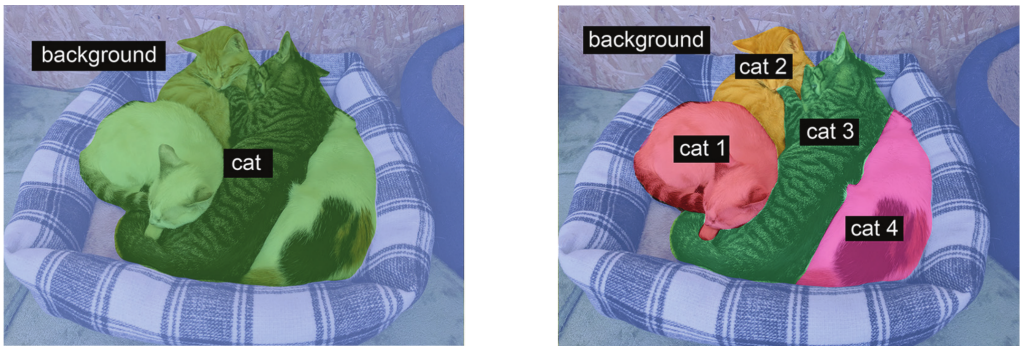
2.3.3 Pretrained Networks

CNN architectures have been topics of study, notably in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [56] between 2010 and 2017. Some popular models to come from these challenges include AlexNet [32], VGG-16 [61], Inception-v3 [64] and SENet [22], to name a few. ILSVRC introduced a classification dataset, and the goal was for researchers to develop the best performing model on that provided dataset. These trained networks would later be shared, allowing their architectures and learned weights to be applied to other tasks. From here, by locking the pretrained layers' weights and replacing the classification output layer, the CNN's feature-extraction capabilities could be applied to new tasks with minimal additional training. This results in faster training times, improved performance when training data are limited, and enables faster deployment in specific domains and comparisons of applications across different models within those domains, such as handwritten recognition [8].

2.3.4 Image Segmentation

Image segmentation is a task in computer vision that partitions an image into meaningful regions, applying labels at the pixel level rather than to the entire image, like in classification tasks. This enables precise localization of objects and boundaries, making segmentation essential for applications such as medical imaging, autonomous driving, remote sensing, and scene understanding [2, 14].

Computer Vision Segmentation Tasks



(a) Semantic Segmentation (b) Instance Segmentation

Figure 2.7: Example of segmentation types (Adapted from [14]).

Image segmentation tasks are categorized into three types: semantic, instance, and panoptic segmentation. Semantic segmentation assigns a class label to every pixel in an image, without distinguishing individual object instances. Such as all pixels associated with cats. Whereas instance segmentation extends this by identifying individual objects of the same class, such as labelling each separate cat as a new group assigned to that label. These differences are represented in Figure 2.3.4. Panoptic segmentation combines both approaches, producing a unified representation that assigns each pixel a class and an instance identifier [14].

Pretrained Segmentation Models

Pretrained segmentation models have become widely used because they can transfer learned representations from large datasets to new tasks, much as CNN models submitted to ILSVRC do. These models reduce training time and improve performance when the amount of labelled data is limited. Segment Anything (SAM) [31] by Kirillov *et al.* is a recent model for image segmentation that supports multiple prompting modalities, including points, bounding boxes, text, and masks. Trained on a large, diverse dataset, SAM can generalize to unseen objects and scenes, making it a versatile tool for both interactive and automated segmentation.

2.4 Related Works

The approaches proposed in this work are relatively novel. However, there are previous works related to our approach of using neural-based machine learning for clustering that employed different architectures.

2.4.1 Neural-based Clustering

Traditional clustering algorithms, such as K-means and K-medoids, group data points based solely on observed characteristics using heuristics such as distance. These are unsupervised learning models, where the number and meaning of labels are unknown. Structures are discovered directly from the data, and labels may be assigned after clusters are formed [25]. Neural networks were initially developed for supervised learning tasks, where labelled data are used to train models to classify unseen observations, based on learned observations [2]. However, researchers also explored their potential for unsupervised learning and clustering. Early neural clustering approaches include Self-Organizing Maps (SOMs) [51], which project high-dimensional data onto lower-dimensional grids while preserving topological relationships. These models demonstrated that neural architectures could learn meaningful representations of data without explicit labels, but had difficulty adapting to changes in the input data, the size of the required output, or the number of clusters to determine. Recent models, such as Robust Embedded Deep K-means Clustering (RED-KC), extend deep clustering by introducing clustering-specific loss functions that improve robustness and cluster compactness [4, 74]. Neural-based clustering aims to combine deep learning feature representation and nonlinear dimensionality reduction with clustering objectives [73, 74]. Deep models can learn feature spaces in which clusters are more separable than in the original input space. This addresses a limitation of traditional clustering methods, which rely on raw features that may not capture the underlying structure of complex data. While kernel methods address nonlinearity through implicit feature mappings, deep clustering learns task-specific representations directly from data. Deep Embedded Cluster-

ing (DEC) [70] is an example of such an ANN framework that learns feature representations and iteratively refines cluster assignments within the learned feature space for each cluster.

Many types of Neural networks have been used to address the problem of clustering, such as Multiplayer Perceptron (MLP), CNN, Deep Belief Networks (DBN), Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE) [4]. These models transform inputs into a latent representation, which is then used for clustering. An example of using deep networks for subspace clustering proposed by Zhang et al. [73] is the Self-Supervised Convolutional Subspace Clustering Network (S^2CSC), which integrates convolutional feature learning with subspace clustering. The model learns a self-expressive representation in which each data point is reconstructed as a combination of points in the same subspace. This approach is particularly effective for high-dimensional data such as images, where clusters may lie in multiple low-dimensional subspaces.

Deep Subspace Clustering Networks (DSC-Net) [26] is another CNN-based approach that employs an auto-encoder with a self-expressive layer to learn relationships among data points. This method performs well on image clustering benchmarks, grouping objects in an image by jointly learning representations and cluster structures. By learning feature representations, they can capture complex, non-linear relationships and improve cluster separability [73, 74].

Chapter 3

Classifying 2D Distance-Based Clusters

In this chapter, we investigate the use of image recognition tools and how they can be applied to the problem of data clustering. For an initial prototype, the goal is to determine the quantities and locations of the clustering using a two-dimensional (2D) representation of the dataset. This will serve as the initial step for our work and will inform future priorities as we explore the feasibility of our proposed approach.

3.1 Introduction

It is noted that finding the exact optimal clustering from data is *NP-hard* [16, 48], even for small values of K [5, 24]. Traditional unsupervised clustering approaches do not utilize previously seen data. Rather, they group points only on the basis of the characteristics observed in the given data, using metrics such as distance. Traditional distance-based approaches that rely on these metrics then impose limitations on the results that can be generated, even in widely used algorithms, such as k-means.

One limitation is the requirement for user input for the parameter K, which specifies the number of clusters to create. Different values of K can yield different results for the same data [24]. Furthermore, these clustering algorithms are sensitive to the initial selection of the K centroids. Poor initialization of the centroids can cause the algorithms to converge to a local minimum rather than the global optimum, due to their greedy nature. This is because a dense region of points, or sufficiently separated data, can prevent the centre from relocating during the update step of the algorithm, preventing it from updating past that

local range [24]. This means that a clustering algorithm requires several runs to determine the best clustering results for any single value of K. This issue is exacerbated by data outliers, which can significantly affect the final clustering results.

Another limitation is imposed by the distance metric that the algorithm uses between points, which can restrict the final shape of clusters around their centroids. In K-means, for example, it is common to use the Euclidean distance metric to measure the distance between points, limiting the possible shapes of the clusters to circles or spheres in higher dimensionality [24].

Additional efforts have investigated different ways to address these problems, such as K-means++ [5], which introduces a heuristic for separating initial centroids, reducing the number of iterations the algorithm requires to converge on a solution. Other works on clustering using alternative approaches, such as the ANN-like models, overcome the issue of having to predefine K clusters to create [25], but introduce other problems related to the stability of the model, consistency of the results produced, and *plasticity*, which refers to the algorithm's ability to adapt to new data. Another drawback of these networks is that they struggle to detect clusters of varying shapes and have a fixed number of outputs, thereby limiting the number of clusters that can be calculated to a predefined range [25]. A range is still an improvement over specifying the exact value of K, but it still creates a restriction on the data that can be processed.

3.2 Problem Statement

In our work, we believe that previously explored and clustered data should be a valuable asset and should be reused for future clustering tasks. In addition, in many application domains, it is reasonable to assume that the number of clusters in a clustering task lies within a finite range rather than an infinite space. For instance, to cluster customers into groups and thereby specify customized marketing strategies, a practical number of clusters could be three to seven. We use this assumption about the appropriate number of clusters

for our initial model, but we plan to explore ways to remove the requirement for a specific range and determine any number of clusters in the following Chapter.

3.3 Our Approach

To overcome the limitations of traditional clustering algorithms, we propose an alternative approach to determining the number and location of clusters. In our initial efforts, we focus on 2D data. Our approach encodes a dataset into 2D images and analyzes them using a CNN model as a categorical image recognition task, rather than comparing each raw data point using pairwise distance calculations. The output of this network is to predict the number of clusters present in an image, which is the class label used to train our model. Therefore, by leveraging previously clustered data, we investigate the feasibility of transforming a clustering task into a classification task.

In addition to using the categorical output of our CNN-based model to determine the number of clusters in an image, we also propose leveraging the information reservoir embedded in our model's convolutional operations to localize potential cluster positions and delineate their shapes.

This proposed approach has several key benefits over traditional clustering algorithms and previous ANN-style approaches. One such improvement is that we do not rely on a centroid to define the cluster location or shape. This eliminates the possibility that outliers or local minima can produce suboptimal results, requiring multiple runs on the same data. Additionally, the size of the CNN output can specify a range of possible cluster counts, so the exact number is not required. Further reducing the number of iterations needed to obtain optimal results. The improvements over ANN-style approaches are twofold. One is the improved plasticity of the input, as it is not reliant on the dataset size but on the resolution of the encoded image. Meaning it is more adaptable to changes in the amount of input data than previous approaches. Another improvement is in the stability and consistency of this approach. Given that the input is a visual representation under predefined settings, we

expect this approach to be deterministic and consistent in its results.

3.3.1 Architecture

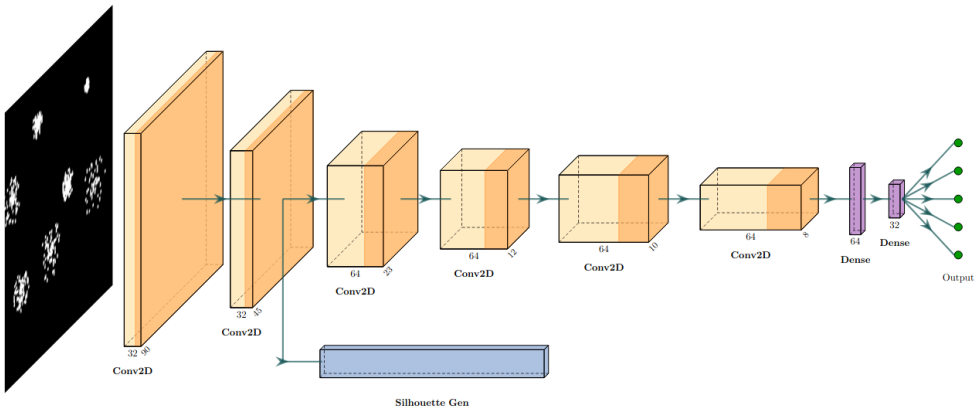


Figure 3.1: The Architecture of our CNN-based Model.

Our proposed CNN-based model, as shown in Figure 3.1, uses a $n \times n$ pixel image as input and is composed of six (6) convolutional layers that feed into three (3) fully connected traditional layers. The first two convolutional layers have a kernel size of four (4), a stride of two (2), an even padding size of one (1), and use 32 filters. The two middle convolutional layers are the same. However, the kernel size is reduced to three (3), with 64 filters. The final two convolutional layers are the same as the previous two layers, with the stride reduced to one (1) and the padding removed. The output of the final convolutional layer is then flattened and passed to a traditional dense layer with a size of 64. This layer is fed into another dense layer of size 32 before being passed to the final output classification layer with C neurons, which outputs the prediction of the corresponding class label (the number of clusters) for an input image from the training data. All hidden layers use a *relu* activation function while the output layer uses a *softmax* activation function[1]. Parameters, such as n and C, will be discussed in Section 3.4.2.

[1]For a comprehensive treatment of convolutional neural networks, we refer the reader to [2], among others.

3.3.2 Extracting Feature Maps

The second component of our proposed CNN model is to generate the silhouettes of clusters from the input data. By reading a hidden layer's activation map within our CNN model, the results from the convolutional operations drawn from an input image can be extracted and inspected. To achieve this, we create a new branch in our processing to extract the action map from our proposed CNN-based model into an algorithm, called *Silhouette-Gen*, which is traditional computation, compared to the neural-based processing of the rest of the network, as shown in Figure 3.1.

Algorithm 3.1: SilhouetteGen

Data: Input image tensor I, trained CNN model M, layer index d, threshold T, rescale factor s
Result: Set of grouped activation regions

activations $\leftarrow M$.predict(I) ; `// Process image with pre-trained model`
layer_activation $\leftarrow$ activations$[0]$; `// Remove batch dimension`
heat-map $\leftarrow$ layer_activation$[:,:,d]$; `// Extract feature map from depth d`
bool-array $\leftarrow$ heat-map $> T$; `// Threshold to create binary mask`
final_array $\leftarrow$ resize(bool-array, s) ; `// Resize activation map`
groupings $\leftarrow [\,]$;
`// Find connected regions using Flood Fill`
for $i \leftarrow 0$ **to** *rows*(*final_array*) $- 1$ **do**
  for $j \leftarrow 0$ **to** *cols*(*final_array*) $- 1$ **do**
    if *final_array*$[i, j] =$ *true* **then**
      group $\leftarrow$ FloodFill(final_array, (i, j)) ; `// Finds connected points`
      append group to groupings
    end
  end
end
return *groupings*;

SilhouetteGen begins by reading the activation map between convolutional layers as a 2D array and interprets it as magnitudes or intensity of data within a specific range, also referred to as a *heatmap*. By setting a threshold value, the heatmap is converted into a Boolean array of all values above the threshold. The value of the threshold can affect how closely the resulting cluster shape is defined. A low value will create a more relaxed shape. In contrast, a high value can aid in separating clusters within close proximity, but may

result in less densely populated clusters being defined separately. The resulting Boolean representation is then rescaled to match the input image size, as the convolutional operations have shrunken the image. Then, a flood-fill or seed algorithm is used to identify all groups of values in the Boolean array. The resulting list of all connected pixel arrays is then returned. This approach is further outlined in Algorithm 3.1. The flood fill function is performed by scanning the 2D heatmap array until a true value is found, then checking all of its neighbours, and returning an array of connected pixel points. For completeness, this algorithm is provided in the Appendix as Algorithm A.1. The final step is to convert the groups into a continuous silhouette around the collection of points contained in each cluster. This is performed in the Generate Convex Hulls function as shown in Algorithm 3.2. In this function, we generate a new convex hull (cluster) around any group that has a size greater than *minSize*. This limit can be adjusted for particularly noisy data as needed, but even very small values are sufficient to prevent individual points from being treated as distinct clusters.

Algorithm 3.2: Generate Convex Hulls

```
Data: groups of points groupings, minimum size of a hull to create minSize
Output: Dictionary of convex hulls
index ← 0;
foreach group in groupings do
    if |group| ≥ minSize then
        hull ← ConvexHull(group);
        area ← hull.volume;
        if area ≥ min_area then
            convex_hulls["hull_"|| index] ← hull;
            index ← index + 1;
        end
    end
end
```

3.4 Experiments

We develop our proposed CNN-based model using *Keras* [2] and *tensorflow* [3]. To evaluate the feasibility of this proposed approach to data clustering, our goal is to examine the number of clusters present in a given image and to tap into a convoluational layer to inspect the extracted features.

The model architecture involves an input layer that reads an image with dimensions $n * n * 3$, where $n = 180$ in our simulations. This is a common input dimension of image processing with CNN's, although other dimensions can be used. The image is then processed by six (6) convolutional layers before being flattened into two fully connected layers, leading to a final output classification layer. The output layer is restricted to five (5) outputs (shown in Figure 3.1) to determine between a preset number of clusters present in an image. In the SilhouetteGen algorithm, we used a threshold value of 0.9, a minSize of 3, and we extracted data from the activation map from the second hidden layer. Note that this extraction point could be changed to a different layer depth. During our simulations, the activation map from the second hidden layer has produced the best results.

3.4.1 Software and Libraries

The creation of the dataset and network utilized various software packages and libraries to facilitate our research and experiments, including:

- Environment Management: *Conda* [4], *Jupyter* [5]
- Machine learning Frameworks: Keras, tensorflow, *Scikit-learn* [6]
- Visualization Libraries: *Matplotlib* [7] [23], *Plotly* [8], *Scikit-image* [9]

[2] https://keras.io/.
[3] https://www.tensorflow.org/.
[4] https://docs.conda.io/.
[5] https://jupyter.org/.
[6] https://scikit-learn.org/.
[7] https://matplotlib.org/.
[8] https://plotly.com/.
[9] https://scikit-image.org/.

3.4.2 Data Collection

A synthetic dataset was created to train our CNN model. The dataset comprises 5,000 images, evenly divided into categories based on the number of clusters per image, ranging from three (3) to seven (7), producing the five ($C = 5$) class labels. Each cluster has a random number of data points within a randomly sized radius around that cluster centre point, with a set minimum and maximum value for those fields. A cluster's centre points can be no closer than twice the maximum size of the radius plus one point. This ensures that no clusters can overlap but allows them to be placed in close proximity. The dataset is divided into 75% for training, 20% for validation, and the remaining 5% for testing. The images are exported using a black background, and the data points are presented as white dots. In addition to analyzing the accuracy of the CNN output classification, the test images are saved with the coordinates of each data point from each cluster and their associated cluster identifiers. The labelled data points enable a more detailed analysis of per-cluster accuracy.

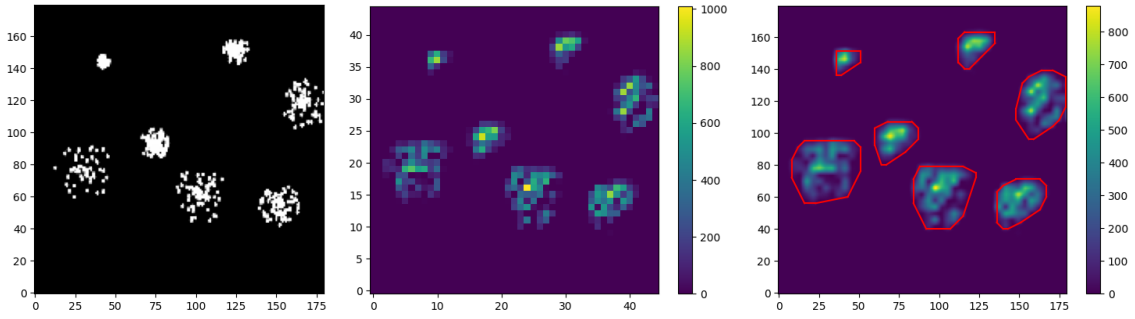
In addition to our synthetic data, we utilize one dataset from Mikhail's *2D Clustering Dataset Collection*, referred to as *file_11* [42]. This will provide an opportunity to examine our approach to non-spherical cluster shapes, a unique situation given the strictly spherical training data.

3.4.3 Training

The three datasets, testing, training, and validation, were configured with a batch size of forty (40). In addition, training was configured to use Keras callback checkpoints and early stopping. This means training can be configured for a large number of epochs (or training cycles), and the highest performing model will be saved. Checkpoints are configured to save only the best model, as determined by its accuracy on the validation set. Early stopping monitors the loss calculated on the validation set, and has a patience configured to five (5), meaning training will terminate if no improvements are made within the last five epochs.

The Keras *fit* function was then used with these configurations and a maximum number of epochs of 50. The callback function then terminated the training after the 27th epoch.

3.5 Results and Discussion



(a) Input Image

(b) Heatmap from an Activation Map

(c) Silhouette overlay on Heatmap

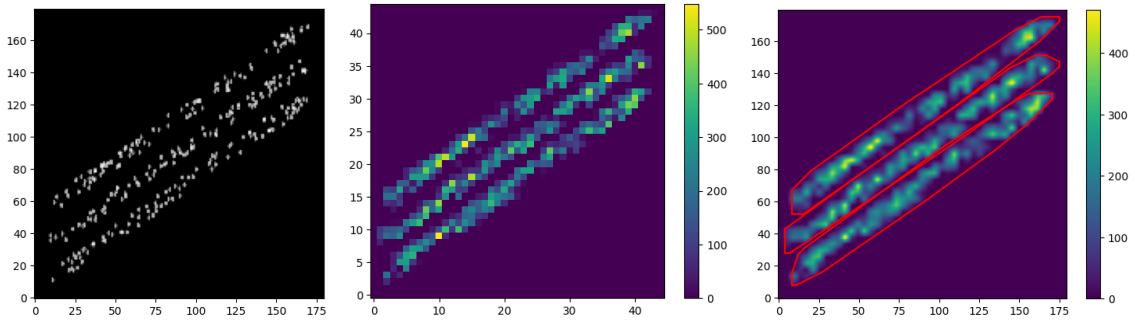
Figure 3.2: Image Augmentation and Cluster Silhouettes.

After training, our proposed model correctly predicts the number of clusters in an image, achieving an accuracy of 95.6% on the previously unseen test dataset. These results were obtained solely from the CNN's output classification and are encouraging initial findings, but only predict the number of clusters within the dataset. To determine the locations or shapes of clusters, the activation map needs to be converted to a heatmap, which is then used to create silhouettes. Figure 3.2 illustrates the various steps of this approach. Figure 3.2(a) shows an example of an image from the test dataset that is given to the network as input. Figure 3.2(b) represents the CNN's layers activation map after being transformed into a heatmap of values that are above a preset threshold. Figure 3.2(c) shows the overlay of the generated silhouettes on top of the previous heatmap. It is easy to see a clear correspondence among the input, the heatmap, and the generated clusters.

The results gathered by testing the placement of the recorded data points against the silhouettes generated from the images shown in Figure 3.2 are presented in Table 3.1. This table lists each cluster by its identifier, the number of points it contains, the number of tested points that are either outside any cluster or in the wrong cluster, and the percentage accuracy

Table 3.1: Generated Silhouettes on Image Example in Figure 3.2(a).

Cluster ID	Points in Cluster	Points Outside Convex Hulls	Points in Incorrect Convex Hull	Accuracy (percentage)
1	61	0	0	100
2	76	0	0	100
3	87	0	0	100
4	88	6	0	93.18
5	80	11	0	86.3
6	95	6	0	93.7
7	81	4	0	95.1



(a) Input Image

(b) Heatmap from an Activation Map

(c) Silhouette overlay on Heatmap

Figure 3.3: Non-Spherical Cluster Silhouettes Produced.

for all points in each cluster. We computed the accuracy of our approach as the proportion of correctly assigned points across all clusters. Of 568 total points in this example, 541 were correctly assigned, resulting in an accuracy of 95.25% across all clusters. An overall accuracy of 93.26% has been observed for testing all individual point placements across all images in our test dataset.

To further test the feature-extraction capabilities of our proposed approach, we experimented with non-spherical data from Mikhail's file_11 dataset [42]. Although our model was trained exclusively on spherical clusters, the SilhouetteGen algorithm accurately generated the contours of rectangular clusters, without requiring retraining with the addition of those shapes in the training data. The results of this are shown in Figure 3.3. These promising results demonstrate the generalizability of extracting information from the net-

work's hidden layers and highlight the flexibility of this approach in handling new data and detecting clusters of varying shapes.

3.6 Summary

The work in this phase of our research introduces a novel approach to identifying and locating clusters within a dataset. The results of our initial simulations on synthetic data are promising. Our proposed model addresses some concerns of distance-based clustering algorithms, as discussed in Section 3.3.1, by eliminating the need to specify an exact number for the parameter K, and removing the risk of initial centroid placements becoming trapped in local minima. Both improvements reduce the number of iterations required to generate globally optimal results on the same data. Additionally, this neural network architecture is flexible in handling datasets of varying sizes because different numbers of data points can be encoded into the same-sized image. making the input more flexible than previous ANN approaches. This was observed in the test data, where the number of clusters and the number of points per cluster varied across images.

Our model can also recognize clusters of different shapes. This can be observed in the examples in Figure 3.2 and Figure 3.3, which allows for more detailed silhouettes of the clusters than those produced by centroid-based algorithms that are restricted to distance functions from their centre point, such as spherical shapes when using Euclidean distance [24]. This demonstrates the potential of our proposed approach for handling clusters of different shapes. We plan to further adapt our approach to handle density-based clustering and more complex shapes, as discussed in the following chapter.

Chapter 4

Classifying 2D Density-Based Clusters

The prototype SilhouetteGen model that we introduced in the previous chapter was trained exclusively on round clusters. By testing non-spherical clusters in parallel lines, as shown in Figure 3.3, the SilhouetteGen algorithm accurately generated the contours of the point groups without retraining the model. These promising results demonstrate the robustness of using information from the network's hidden layers to distinguish between groups in the data. Patterns in real-world datasets can take many forms, including nested groups and curved, non-linear structures. These patterns may contain interior angles greater than 180° and are therefore referred to as non-convex. These non-convex shapes, along with groups nested within larger groups, such as multiple concentric rings, pose a challenge for centroid and distance-based models that rely on a distance function for a centre point to define the cluster boundaries [21, 39]. Instead, density-based models look for groups of densely populated areas, defined by a minimum spacing and a minimum number of points within that area to be considered dense, separated by sparsely populated areas. Points within the less populated region can then be classified as outliers that do not belong to any group.

Since our model's clusters are not defined by a centre point and radius, it is suitable for density-based clustering problems, given that our approach can define shapes that are not constrained by a distance function. Answering the question of how our model can be adapted and improved to detect more complex patterns in data, such as varying densities and non-convex or nested shapes, is the primary focus of this chapter.

4.1 Introduction

Density-based clusters are not confined to a single centroid, unlike many distance-based approaches. Rather, the points are grouped by their proximity to surrounding data points. This makes these approaches more flexible for generating non-spherical clusters and those with an arbitrary or complex shape that may not have a common centre point. Some distance-based approaches, such as kernel K-means, can partially address this limitation by mapping data into a higher-dimensional feature space where complex structures may become separable. However, selecting an appropriate kernel function can be challenging, and even when effective kernels are identified, these approaches remain sensitive to noise and outliers [21]. Another advantage of density-based approaches is the automatic determination of the number of clusters within a dataset. The data points are explored, and their relationships with neighbouring points are used to determine whether they are part of a cluster or an outlier. This determination does not rely on a specified user input for the number of groups to create [39]. Density-based approaches also excel in detecting noise in datasets. Unlike centroid-based models, which can be sensitive to outliers due to poor initialization, density-based models make distance and neighbour comparisons as points are explored, not only to a centroid but to their local surroundings, meaning they can quickly determine whether a point is part of a group or distinct from other values, allowing it to be detected as an outlier [21].

DBSCAN is one of the most cited and widely used clustering algorithms in the scientific community [20]. The algorithm systematically finds all core points, as defined by the input parameters ε and MinPts. Then, it finds any points around the core points, but within the range of ε as border points. Any other points are considered outliers. Although DBSCAN is widely used and offers notable benefits, it has limitations and considerations when used.

One known limitation of DBSCAN is its high sensitivity to its input parameters. Even small changes can lead to large variations in results [39]. These parameters are notoriously difficult to determine in real-world datasets, and sometimes there is no single set of param-

eters that clusters all points [33]. This leads to the second problem: DBSCAN has difficulty identifying clusters of varying densities [33, 39]. This can be attributed to the issue of input parameters; if the dataset contains clusters of varying densities, it can be challenging and in some cases even impossible to determine the parameter values needed to extract real cluster structures from the data across diverse cluster structures [33]. HDBSCAN (Hierarchical DBSCAN) is a hybrid clustering method that combines density-based clustering with hierarchical clustering, enabling the detection of clusters with varying densities. HDBSCAN calculates core distances to the K^{th} nearest neighbour of each point to construct a weighted graph of points. Then a minimum spanning tree is used to determine the hierarchy, where different levels or groups may have different densities [18]. Although this increases its flexibility in determining clusters and reduces the complexity of parameter selection, it has high computational costs [11].

Border points introduce additional limitations, such as difficulties in discerning the boundary between neighbouring clusters with identical densities [39]. This can lead to clusters being merged unnecessarily. Another issue with border points is their sensitivity to the input order [20]. If the same dataset is reversed, the border points between two groups of core points may be assigned to the other cluster, since they are assigned to the cluster that is expanded first. Meaning that the order in which data is processed can affect the results produced. DBSCAN* and HDBSCAN* avoid this issue by abandoning the concept of border points and treating all would be boarder points as noise [20, 59].

4.2 Problem Statement

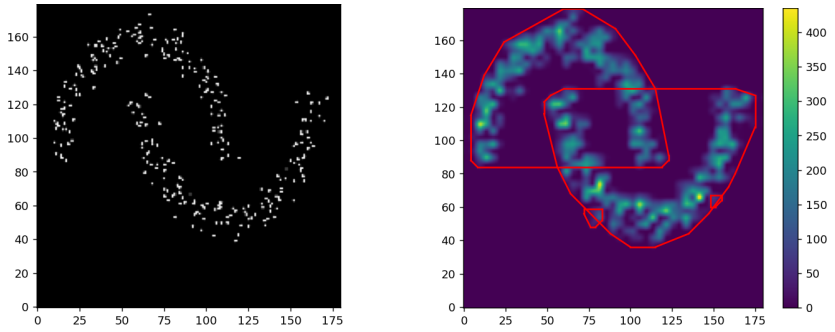
Given these descriptions regarding density-based clustering algorithms and our results from the previous chapter, it is evident that our model leverages some key characteristics of density-based approaches. Primarily, these aspects are the automatic detection of the number of clusters and the detection of clusters of different shapes based on spatial occupancy. This alignment invites further investigation into extending or improving our model

for detecting density-based clusters. To make that comparison fully, we need to improve the robustness of our model in delineating clusters of any arbitrary shape. In addition, update our model so that it does not limit the number of clusters that can be detected, which is currently constrained by the number of categorical output nodes in our CNN.

4.3 Our Approach

To improve the SilhouetteGen model, we first need to identify the weaknesses or representations that it struggles to process. One such weakness of the model in Chapter 3 is its handling of clusters with more complex shapes. Although our initial approach allowed for clusters of different shapes, the convex-hull representation limited the complexity of the shapes it could generate. When visually inspecting the heatmaps produced by the CNN prototype, we observe that the hot spots align with the expected cluster shape. The issue arises in interpreting the heatmap and generating convex hulls, which, by its very name, is not suitable for more complex shapes. Additionally, the original model assumes that all points within the shape's outer boundary belong to that cluster. This means that any nested groups would also belong to the parent group that encompasses them. An illustration of these issues can be seen in Figure 4.1. In this example, the CNN heatmap aligns with our visual intuition of what the cluster should look like; however, the convex hull representation fails to capture the contours of the inner edges of these shapes. This suggests that we do not need to retrain the CNN portion of the model, as it is adapting to this new input and representing the information we want to capture. But we should rethink how we generate cluster information from the activation layers.

The problem we now address is whether we can adjust how we interpret the activation map to enable the model to differentiate between clusters that are nested within one another or have an interlaced structure, so that forming a box to enclose one cluster does not, either partially or completely, enclose another.



(a) Input image (b) Convex Hull Results

Figure 4.1: Demonstration of Non-convex Clusters Issues.

4.3.1 Architecture Changes

In our density-based adaptation, we replaced the Convex Hull generation with a point-cloud representation, in which each cluster is stored as an array of x and y-coordinates for all points associated with that cluster. In our SilhouetteGen algorithm, this is represented by the variable group, which is populated by the Flood Fill function's return value. This group comprised the point cloud of an individual cluster. These individual groups are then appended to the clusters array, which is an array of arrays. The length of the clusters array can then be used to determine the number of clusters in a given dataset.

This approach reduces the computational cost of generating cluster data from the full dataset compared to the previous generation of convex hulls, because we can skip generating them entirely. Instead, we only reinterpret how we used the group points with the SilhouetteGen algorithm. The only other consideration is to move the *minSize* parameter from the convex hull generation into the SilhouetteGen algorithm. This updated approach is shown in Algorithm 4.1.

Our previous model had six convolutional layers followed by three dense layers (including the output classification layer), totalling $412,037$ trainable parameters (weights). For this new approach, we removed all network layers beyond the point at which we read the activation maps. The network's categorical output is redundant after training, as the number of clusters can be determined by counting the distinct groups produced by the Silhouette-

Algorithm 4.1: SilhouetteGen (Density)

Data: Input image tensor I, trained CNN model M, layer index d, threshold T, rescale factor s, minimum size clusters *minSize*
Result: Set of grouped activation regions

activations $\leftarrow M$.predict(I) ; `// Process image with pre-trained model`
layer_activation $\leftarrow$ activations[0] ; `// Remove batch dimension`
heat-map $\leftarrow$ layer_activation$[:,:,d]$; `// Extract feature map from depth d`
bool-array $\leftarrow$ heat-map $> T$; `// Threshold to create binary mask`
final_array $\leftarrow$ resize(bool-array, s) ; `// Resize activation map`
clusters $\leftarrow$ [ ];
`// Find connected regions using Flood Fill`
for $i \leftarrow 0$ **to** *rows*(*final_array*) $- 1$ **do**
  for $j \leftarrow 0$ **to** *cols*(*final_array*) $- 1$ **do**
    if *final_array*$[i, j] =$ *true* **then**
      group $\leftarrow$ FloodFill(final_array, (i, j)); `// Finds connected points`
      if *group* $>$ *minSize* **then**
        append group to clusters
      end
    end
  end
end
return *clusters*;

Gen scan of the activation map. Meaning further processing through the standard network architecture does not yield any new information. Also, removing that output removes the current restriction that limits the number of clusters that can be predicted. This means we can reduce computational cost and time by removing it and improve the model's ability to automatically detect the number of clusters present in a dataset.

After trimming those layers, the new model consists of only two convolutional layers and has a total of $17,984$ parameters, representing a 95.6% reduction in the required weights and, consequently, in the calculations needed to process each dataset. This new architecture is shown in Figure 4.2.

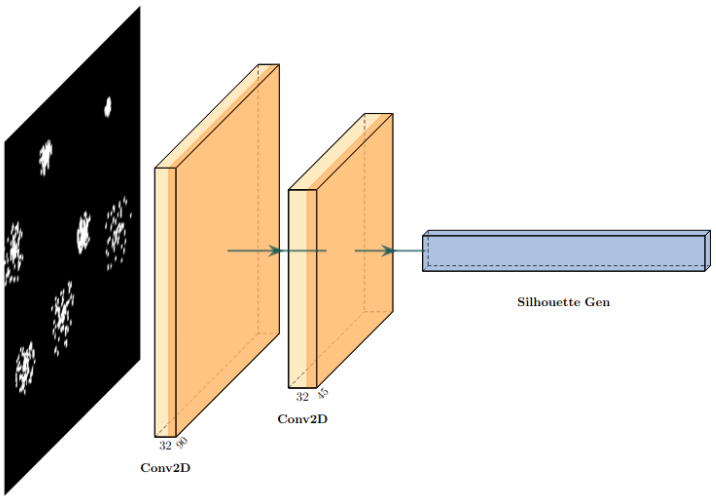


Figure 4.2: Trimmed architecture of our 2D CNN-based Model.

4.4 Experiments

To conduct the following experiments, we modified our existing approach from the previous chapter by updating the SilhouetteGen function and pruning the output and excess hidden layers. Importantly, we did not generate or use any new information to train the network. The two remaining CNN layers use the trained weights from the synthetic spherical datasets used in the previous chapter, and no updates were made to the weights when pruning the network's hidden layers. This demonstrates the robust feature extraction capabilities of CNNs and illustrates how our approach yields a transferable method for clustering new and previously unseen data and patterns.

4.4.1 Datasets

Multiple datasets were used to evaluate different density-based clustering constraints. The first dataset is the 2D Clustering Dataset Collection generated by Samoilov Mikhail [42], the same dataset we used for representing the non-spherical parallel clusters in the previous chapter. This collection of datasets comprises additional 2D datasets designed to analyze clustering approaches. The datasets are also labelled, enabling extrinsic evaluation metrics such as precision, recall, F1-score, and purity. Figures 3.3 and 4.1 are examples of

this dataset, among future subsets of that collection that we will use for density comparisons throughout this phase of our research.

The second dataset is published by Thrun and Ultsch [65]. This is a collation of datasets from the *Fundamental Clustering Problems Suite* (FCPS), which presents various challenges for clustering algorithms, along with descriptions of the content and the challenges a clustering approach would need to overcome to accurately define those clusters. More details on individual datasets will be reviewed as we discuss each example in Section 4.5.

4.4.2 Evaluation

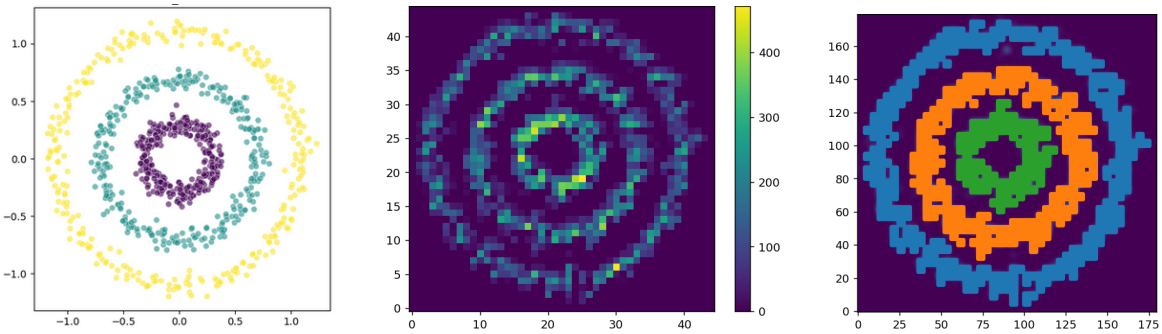
Given that our datasets are labelled, we can use extrinsic quality metrics such as precision, recall, F1-Score, and cluster purity to evaluate the quality of our results. These calculations are discussed in Section 2.1.6, and we use Scikit Learn's *metrics* [10] API for our implementation of these evaluations. Additionally, we can utilize Scikit Learn to import an existing implementation of *DBSCAN* [11]. This will allow us to compare our results with a documented existing approach of a well-known and researched method.

4.5 Results and Discussions

Our updated model was tested on a subset of the datasets used. The datasets were selected to investigate specific challenges related to density-based clustering algorithms and are available in 2D representations, which are suitable for this iteration of our model. This includes nested structures, clusters with short intercluster distances combined with varying intracluster distances, weakly linked clusters (groups that nearly touch), and clusters of varying densities.

Mikhail's datasets included a nested cluster example consisting of three concentric rings. The results of our new point cloud representation of clusters after being processed by the pruned CNN architecture are demonstrated in Figure 4.3. In these results, we can

[10] https://scikit-learn.org/stable/api/sklearn.metrics.
[11] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.



(a) Reference Image

(b) Heatmap of the Activation Layer

(c) New Approach

Figure 4.3: Demonstration of Improved Clusters Nested Rings.

observe a clear distinction between the dataset's reference image and the generated cluster shapes. In addition, we can evaluate the placement of points within the generated groups to assess accuracy. Those results are shown in Table 4.1. We observe that all clusters have accuracies above 90%, but some points are identified as not belonging to any cluster and are therefore considered outliers.

Table 4.1: Nested Rings Results From Figure 4.3.

Cluster ID	Points in Cluster	Points Outside any Cluster	Points in Incorrect Cluster	Accuracy (percentage)
1	300	30	0	90
2	300	21	0	93
3	300	17	0	94.3

Given that our modified approach separates these nested clusters, we can continue our experiments with a more detailed analysis of the placement of individual points and compare our results with DBSCAN. An important first step is to determine that we detected the correct number of clusters, which both approaches did, and then to assess the cluster quality metrics. The results of our model and that of the DBSCAN algorithm for the Nested Rings example in Figure 4.3, as well as the non-convex crescent moon example in Figure 4.1, are in Table 4.2. From these results, we see that DBSCAN outperforms our model in these examples. But our approach ranks highly across all the evaluation metrics.

Table 4.2: Quality Comparisons on Figure 4.1 and Figure 4.3.

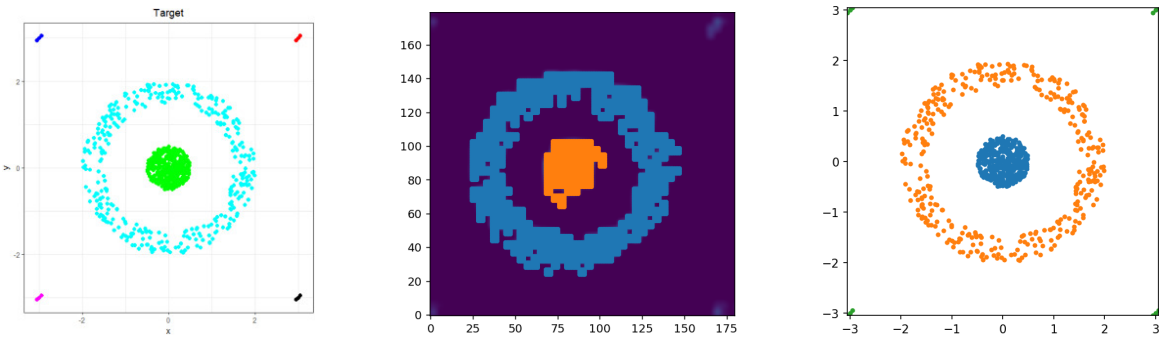
Dataset	Approach	Metric			
		Precision	Recall	F1 Score	Purity
Nested Rings	Our Model	1	0.9244	0.9606	0.9578
	DBSCAN	1	1	1	1
Crescent Moons	Our Model	1	0.9367	0.9672	0.9767
	DBSCAN	1	1	1	1

4.5.1 Density Benchmark Dataset

With our adapted density-based clustering approach, which returns cluster shapes that can be visually verified and achieves precision above 90%, our next step is to compare its performance across datasets from the Thrun and Ultsch benchmark set [65]. Three datasets were used for this evaluation, each posing different challenges for density-based models and composed of 2D data. An additional dataset from Mikhail's collection and one dataset from our synthetic test set were also used. Reviewing these challenges will help us compare our model with a popular approach in areas that are known to be challenging.

The first dataset is named Target and included 770 data points. These comprise two main clusters: one spherical, and the other a ring that surrounds the spherical cluster. In addition, four outlier groups separated from the two main clusters. The challenge with this dataset is the presence of nested shapes within the main clusters and the introduction of noise in the four small groups of points that do not belong to any cluster. Figure 4.4 shows an example image of that dataset, with the results from our model and from DBSCAN. Both approaches determined the correct number of clusters and accurately classified the small groups as outliers. Specific quality metrics for all included datasets will be reviewed at the end of this section.

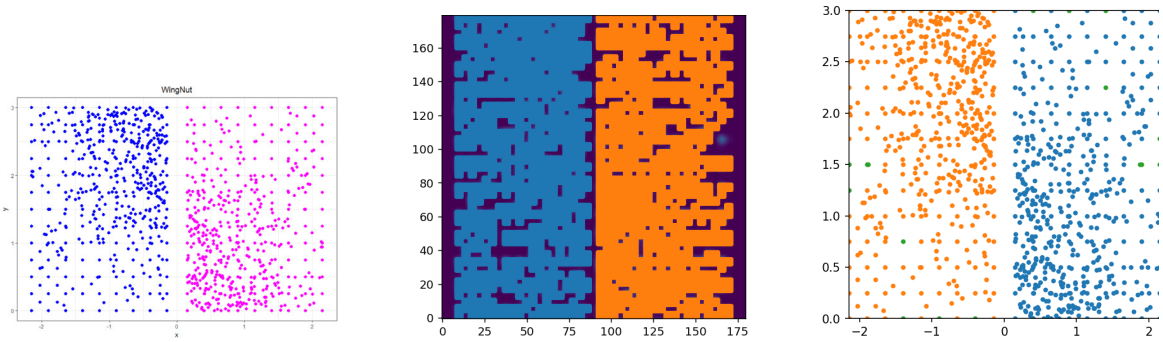
The next dataset is the labelled WingNut, which consists of $1,016$ points between two clusters. This dataset is a collection of random points within a rectangle, with a higher density concentrated toward one corner. That rectangle is then mirrored and shifted, such



(a) Reference Image (b) Our Results (c) DBSCAN Results

Figure 4.4: Our Method Compared to DBSCAN - Target.

that the gap between the subsets is only slightly larger than the gap between points within each rectangle. The specific challenge for clustering algorithms in this dataset is from the small intercluster distances relative to the large intracluster distances near the less densely populated areas of the rectangular cluster. After a few attempts to configure ε and MinPts for DBSCAN and the activation-map threshold in our model, both approaches identified the correct number of clusters, but also assigned a few points as outliers in sparsely populated regions of the two clusters.

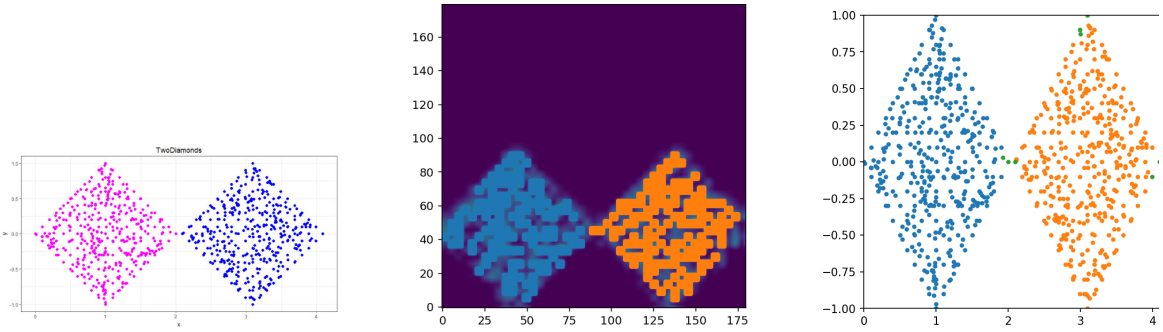


(a) Reference Image (b) Our Results (c) DBSCAN Results

Figure 4.5: Our Method Compared to DBSCAN - Wingnut.

The third dataset from the Thrun and Ultsch benchmark set is TwoDiamonds which has 800 points. This dataset contains two independently drawn diamond shapes with a uniform distribution of points within them. These two diamonds are aligned such that at one point, they nearly touch. The challenge this dataset introduces is to group these two clusters of similar densities that are nearly connected. The connection is sufficiently close that some

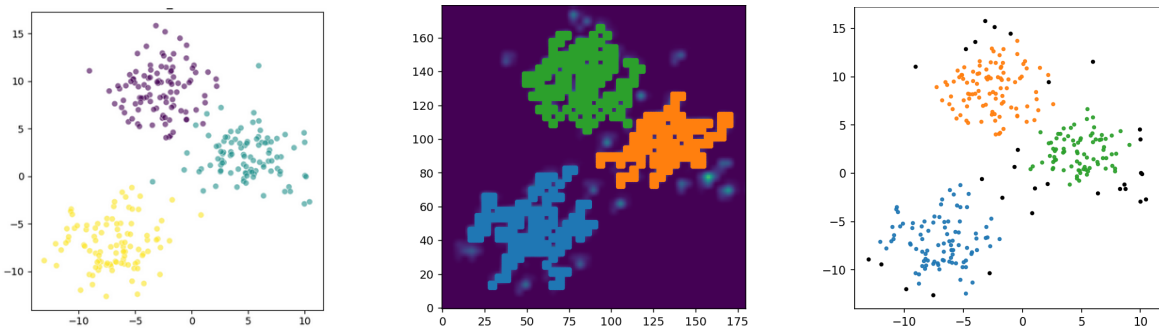
approaches may form weak links between the groups. A reference image of this dataset, along with our results and the results of DBSCAN can be seen in Figure 4.6. As the clusters move closer together, the time required to select the appropriate parameters increases, but both models correctly identified the two clusters, however, at the cost of increased outliers.



(a) Reference Image (b) Our Results (c) DBSCAN Results

Figure 4.6: Our Method Compared to DBSCAN - TwoDiamonds.

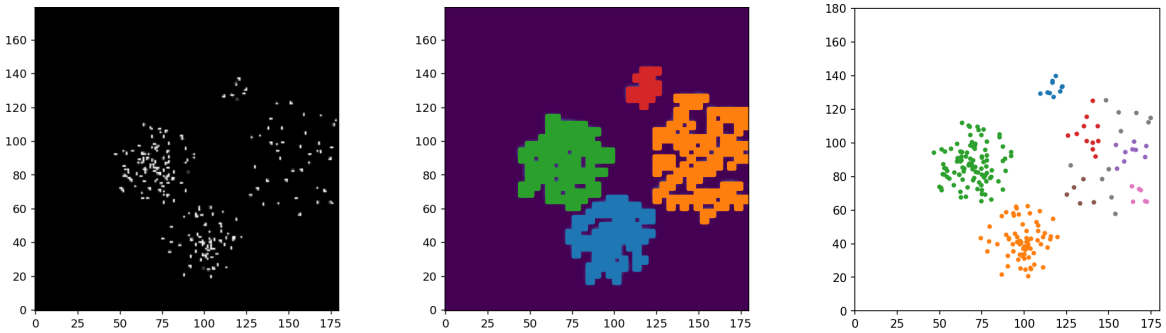
The next image is from Mikhail's [42] and is referred to only as *file_1*. For our purposes, we will refer to this set as Sparse Clusters, as it is less densely populated than our previous datasets. This dataset includes three clusters and 300 total points. As the name suggests, the challenge in this dataset is that the clusters are sparsely populated relative to the distances separating them, with high intracluster distances and some points overlapping near the edges of the clusters. Again, after some tuning, both DBSCAN and our approach determined the correct number of clusters, with both having an increase in falsely labelled outliers. These results can be seen in Figure 4.7.



(a) Reference Image (b) Results of Our Approach (c) Results of DBSCAN

Figure 4.7: Our Method Compared to DBSCAN - Sparse Clusters.

We introduced an additional dataset, Varying Density, to test the limits of clusters of varying densities in close proximity. It comprises four clusters among its 224 total points. The challenge this dataset introduces is distinguishing between dense clusters that are closer in proximity to each other, while also identifying a less densely populated cluster. The parameter selection can make proper separation difficult, which is the distinct challenge this dataset aims to test. It is worth noting that our approach did determine the correct number of clusters, but after several iterations of updating the DBSCAN parameters, it could not determine the correct number of clusters, and labelled many points as noise. The reference image for this dataset and the clustering results are shown in Figure 4.8.



(a) Input image (b) Results of Our Approach (c) Results of DBSCAN

Figure 4.8: Our Method Compared to DBSCAN - Varying Density.

Our approach performs well on the included evaluation metrics; however, DBSCAN outperforms our model in some cases. We present the results for specific cluster quality metrics in Table 4.3. The precision measured across all datasets is quite similar, with both models having a higher stat in one benchmark set. The recall, i.e., how many points were labelled as noise or in the wrong class, was slightly lower, with our model only outperforming DBCAN on our varying-density dataset. Because those two metrics are used to create the F1-Score, a similar pattern emerges. DBSCAN outperformed our approach, but was similar in some scenarios, such as WingNut and Sparse Clusters. For the purity metric, our approach scored lower across the board. The lower score is due to our model often labelling points from each cluster as noise. This noise label is then treated as a cluster,

thereby reducing the overall purity of all clusters. This means that, in scenarios such as Varying Density, where DBSCAN cannot accurately determine the true number of clusters, the purity is higher because it creates more clusters, and those clusters are homogeneous, even if they are not complete. Overall, these results show that our model has a slight advantage in datasets with clusters of varying densities. A known challenge for density-based approaches.

Table 4.3: Quality Comparisons on Figures 4.4, 4.5, 4.6, 4.7, and 4.8.

Dataset	Approach	Metric			
		Precision	Recall	F1 Score	Purity
Target	Our Model	0.989	0.966	0.975	0.981
	DBSCAN	1	1	1	1
WingNut	Our Model	1	0.938	0.968	0.971
	DBSCAN	1	0.984	0.992	0.992
TwoDiamonds	Our Model	1	0.79	0.883	0.903
	DBSCAN	1	0.99	0.995	0.999
Sparse Clusters	Our Model	**0.993**	0.87	0.927	0.923
	DBSCAN	0.99	0.89	0.936	0.943
Varying Density	Our Model	1	**0.946**	0.972	0.973
	DBSCAN	1	0.901	0.932	1

4.6 Summary

Through these experiments, we have demonstrated that our model can adapt to the non-convex, nested, and arbitrarily shaped clusters, a characteristic of density-based clustering problems. As well as identify any number of clusters, even for values that were not possible in the original training data, which in these examples were two clusters.

Our approach can be further refined to improve performance in these benchmark sets. Topics for such improvements include varying the size of data points encoded in the image, a recurrent-network strategy that pulls and bins the points using SilhouetteGen before applying another round of CNN transformations, with the goal of filling some intraclus-

ter voids. Or investigating the use of coloured points to increase the number of attributes that can be processed, such as having colour be a categorical attribute representation. Although we believe these adjustments could improve the model at this stage, these results demonstrate the capabilities of density-based clustering with our CNN feature-extraction approach, which was the primary focus for this stage of our research. A higher priority will be to adapt our input to fully utilize the CNN input shape by using 3D volumetric representations of data, which will be discussed in the following chapter.

Chapter 5

Classifying 3D Distance- and Density-Based Clusters

In the previous chapter, we modified our SilhouetteGen algorithm that reads the heatmap from our CNN to adapt to arbitrary shapes and removed any restrictions on the number of clusters that can be discovered by our approach. Having obtained satisfactory results, we now examine our CNN architecture to determine how to adapt our approach to handle more complex inputs. Specifically, we aim to avoid translating the dataset into an image and instead use a tensor array, which is more suitable for 3D (3D) data. A standard 2D image representation of data can obscure 3D representations because foreground data points restrict the view of background points, necessitating an update to our approach to ensure that we do not lose any detail when projecting data into a third dimension for our model to interpret. Our goal is now to translate our input data into a 3D representation and continue using the FCPS datasets to evaluate our approach against DBSCAN.

5.1 Introduction

CNNs used for image segmentation and classification typically operate on 2D inputs, such as images, similar to the representation used in the previous chapters. However, CNNs can also be extended to operate on other dimensional input data through one-dimensional and 3D convolutional layers.

CNNs used for image segmentation and classification typically take 2D inputs, namely images, similar to our initial approaches. Common approaches to extend this to 3D use

stacks of images, or 'slices', of a volumetric space to construct a 3D representation from multiple images. This is commonly used in medical image segmentation, such as heart-chamber segmentation from CT scans and brain-tumour detection from MRI scans [3, 49].

Another approach to provide these 3D inputs to a CNN is to use a tensor array, a multi-dimensional array of the input data. In Keras, a 3D layer expects a five-dimensional input tensor with shape (batch size, height, width, depth, channels). The first dimension is the batch size, representing the number of samples. The following three dimensions define the 3D spatial structure of the input, and the final dimension stores the input channels or features. These channels may correspond to colour or extracted feature information during processing. Keras includes a built-in Conv3D layer that applies 3D convolutional filters to this tensor representation. Our goal is to determine whether our model and feature-map extraction approach can be reformulated using Conv3D layers and tensor-based inputs. In doing so, increasing the dimensionality of the input representation and enables benchmarking on 3D datasets.

5.2 Problem Statement

As noted previously, humans are capable of visually judging the quality of clusters up to three dimensions [21]. That is, we can quickly inspect data, and make a determination of whether groups exist that structure that data. With this being a primary motivation of this research, to offload those cognitive recognition tasks to a neural-based model, we can then consider how to incorporate that third dimension into our approach, using existing tool sets for visualization. In this chapter, we explore whether our proposed model and feature-map extraction approach can be reformulated using Conv3D layers and tensor-based inputs. By doing so, we aim to increase the dimensionality of the input representation and enable benchmarking on 3D datasets. To continue exploring how CNN architectures can be used for cluster analysis, another consideration is how the data is represented when imported into the model, as when analyzing a dataset, we do not have the available topographical

information available in slices of medical images, and generating several stacks of images for a single dataset may be a burdensome task for the data encoding stage.

5.3 Our Approach

Neural networks define a predetermined input shape; in our last model, this shape was determined by the size of the input image representation of the dataset. Similarly, adapting to a 3D input space, we need to define the shape of our data, independently of the number of points in the dataset. Simply providing the CNN with a tensor array of points is insufficient because the number of points across different datasets is not constant. To achieve a standardized input shape, we have incorporated a modified voxelization approach from Ayushi Sharma [60]. *Voxelization* converts an unordered point cloud into a structured 3D grid representation. Instead of representing data as individual spatial coordinates, the space is discretized into evenly sized volumetric cells, called *voxels*. Each voxel stores information about whether a point from the original point cloud occupies that region of space. This voxel information is then used as our channel dimension.

The algorithm used to voxelize the coordinates into an $n \times n \times n \times n \times 1$ grid is outlined in Algorithm 5.1. Where the maximum value of the dataset is used for the maximum range value r, which is used to normalize the input into a finite grid size. One additional consideration is the input parameter ρ, which allows any true point to expand to a number of neighbours determined by that parameter. If ρ is set to zero, padding is applied because the voxel must span at least one voxel, otherwise if ρ is greater than zero, the voxel for one point will expand that number of neighbouring voxels in all directions. This consideration was included to reduce voxel-grid sparsity, which becomes more pronounced as the representation's dimensionality increases, which can be linked to the curse of dimensionality. This voxel growth strategy will aid in situations where the data density is sparse, but it can negatively affect the results if the input is too high, as it will cause clusters in close proximity to merge. Additionally, we have converted our threshold parameter into a percentile of

Algorithm 5.1: Point Cloud to Voxel Grid

Input: Point cloud P with coordinates (x, y, z), grid size n, maximum range r, point radius ρ

Output: Voxel grid V of shape $n \times n \times n \times 1$

$V \leftarrow np.zeros(n, n, n)$; // initialize voxel grid $V \leftarrow 0^{n \times n \times n}$

$x, y, z \leftarrow points.T$; // Extract coordinate vectors from P

// Normalize coordinates to grid indices

$x_{temp} \leftarrow \text{clip}\left(\frac{x}{r}(n-1), 0, n-1\right)$

$y_{temp} \leftarrow \text{clip}\left(\frac{y}{r}(n-1), 0, n-1\right)$

$z_{temp} \leftarrow \text{clip}\left(\frac{z}{r}(n-1), 0, n-1\right)$

if ρ *is* 0 **then**

  $padding \leftarrow 1$; // Voxel must span at least one space

else

  $padding \leftarrow 0$

// Convert x_i, y_i, z_i to integer voxel indices

foreach *point* (x_i, y_i, z_i) *in* $(x_{temp}, y_{temp}, z_{temp})$ **do**

  // Determine neighbourhood bounds

  $x_0 \leftarrow \max(0, x_i - \rho)$;

  $x_1 \leftarrow \min(n, x_i + \rho + padding)$;

  $y_0 \leftarrow \max(0, y_i - \rho)$;

  $y_1 \leftarrow \min(n, y_i + \rho + padding)$;

  $z_0 \leftarrow \max(0, z_i - \rho)$;

  $z_1 \leftarrow \min(n, z_i + \rho + padding)$;

  foreach (x_s, y_s, z_s) *in the neighbourhood of* (x_i, y_i, z_i) **do**

    if $(x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2 \leq \rho^2$ **then**

      $V[x_s, y_s, z_s] \leftarrow 1$

    end

  end

end

return V;

the heatmap data, allowing it to adjust more freely to hotspots based on a given percentage of the dataset and reducing dependence on dataset, or domain specific values.

The updates to our SihouletteGen algorithm are minimal for this stage. The principle change is the incorporation of another dimension of the activation map from the volumetric shape of our data. And adding the aisle (or depth) to the search for connected points. We also have our threshold set as a percentile of the data, mean that paramter T must now be between 0 and 100. These changes are shown in Algortihm 5.2. For completeness, the FloodFill3D approach used is available in the Appendix as Algorithm A.2.

Algorithm 5.2: SilhouetteGen (3D)

Data: Voxel grid V, trained CNN model M, layer index d, threshold percentile (0 - 100) T, rescale factor s, minimum size clusters *minSize*
Result: Set of grouped activation regions

activations $\leftarrow M$.predict(V) ; // Process V with pre-trained model
layer_activation $\leftarrow$ activations[0] ; // Remove batch dimension
heat-map $\leftarrow$ layer_activation$[:,:,:,d]$; // Extract feature map at depth d
bool-array $\leftarrow$ np.percentile(heat-map, $> T$) ; // Threshold to binary mask
final_array $\leftarrow$ resize(bool-array, s) ; // Resize activation map
clusters $\leftarrow$ [ ];
// Find connected regions using Flood Fill
for $i \leftarrow 0$ **to** *rows*(*final_array*) $- 1$ **do**
  for $j \leftarrow 0$ **to** *cols*(*final_array*) $- 1$ **do**
    for $k \leftarrow 0$ **to** *aisle*(*final_array*) $- 1$ **do**
      if *final_array*$[i, j, k] =$ *true* **then**
        // Find connected points
        group $\leftarrow$ FloodFill3D(final_array, (i, j, k))
        if *group* $>$ *minSize* **then**
          append group to clusters
        end
      end
    end
  end
end
return *clusters*;

5.3.1 Architecture

To adapt to this new input, we need to develop a new CNN model. The new model has a Conv3D input layer with kernel size $(3,3,3)$ and stride 1. As in the previous 2D model, we standardize the input shape, but this time with our voxelization method and the grid size n, set to 64. The input layer is followed by two Conv3D layers, each are followed by a 3D max-pooling layer. The model is then flattened into a fully connected (dense) layer, with a size of 512 with a dropout of 50%. The dense layer is then halfed two more times at sized 256 and 128, each with a dropout of 30%. Followed by further reductions to layers of size 64, 32 and 16 before reaching the final classification output layer. The activation map extraction for our cluster analysis is performed after the first max pooling layer. As with the previous model, we did not gain any new information from

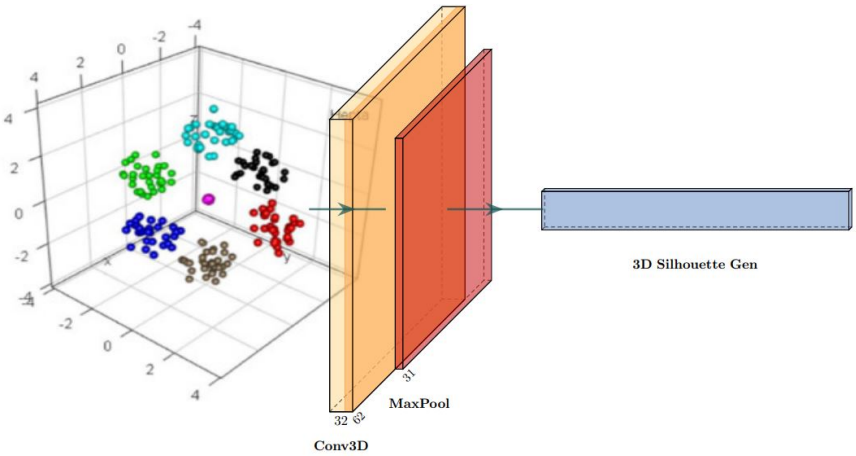


Figure 5.1: Trimmed Architecture of our 3D CNN-based Model.

processing the network past extracting the activation map, so the remaining layers leading to the output classification were trimmed after training was completed. The resulting model has a single Conv3D layer, followed by max pooling, at which point the activaion maps are read by our 3D SilhouetteGen function. The only modification to the SilhouetteGen function was to incorporate the third axis; all other functions remain unchanged. This architecture is illustrated in Figure 5.1. Trimming this new model reduced the parameter count from $14,609,029$ to 896, yielding an over 99.993% reduction over the training model.

5.4 Experiments

To conduct experiments on our 3D model, we first need to train the new network's weights. To do this, we also need to generate 3D training data, similar to the synthetic data used to train the previous model, but suitable for our new input. After training is complete, we can prune the network and feed the CNN's activation maps into our 3D SillhouetteGen algorithm. Once our model is prepared, we can then continue our benchmark comparisons with DBSCAN.

5.4.1 Datasets

With the updated architecture of this model, we need appropriate data to train the network weights. The initial model was trained with a synthetic dataset generated with certain constraints. The data generation for this model follows the same approach outline in Section 3.4.2: Generate training datasets with three (3) to seven (7) clusters each, with varying sizes and numbers of points allocated to them, all within a set radius around a central point. This time in a volumetric 3D space consisting of x, y, and z coordinates. The resulting data is then stored in a multidimensional NumPy array, with dimensions determined by the number of points in each dataset. Before inputting the data into our CNN, the data must be standardized and the point locations defined along the channel dimension, which is then appended to form a full 3D cube map of the input space. That is, we no longer have a list of points, but a full definition of the entire $n * n * n$ input space, and the locations of the points are indicated by a Boolean value. The last dimension (or the first structurally) is the batch dimension, which holds all of these datasets. This is what gives us our required five-dimensional tensor for the Conv3D input and for training. For this training cycle, we reduced the number of samples from $5,000$ to $4,000$ due to a memory constraint when training with 3D representations. The dataset is then split into training (75%), validation (20%), and testing (5%) sets.

In addition to the synthetic data used to train the model, we used additional datasets from the FCPS as collated by Thrun and Ultsch [65]. These datasets were specifically designed for the 3D clustering problem and are described in more detail in Section 5.5, where we review the results of our model and DBSCAN for each dataset.

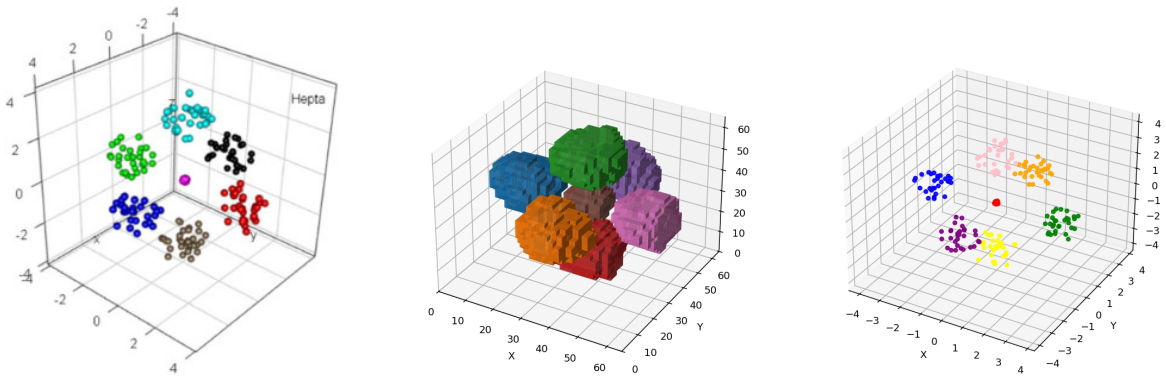
5.4.2 Training

During training of our 3D model, the same configuration was used to monitor validation set accuracy, with a callback function to save the best model. Training was stopped after the $20th$ epoch, reaching only a classification accuracy of 76.6% on the test dataset. While this

is below our expectations, this is an output that will be removed. With subsequent attempts having no meaningful impact on classification performance, we decided to continue testing on the SilouteeGen output to determine whether better results could be obtained by reading the activation maps after removing the network's categorical output.

5.5 Results and Discussions

After training our new 3D model on our synthetic dataset, we returned to the benchmark set to run our experiments and evaluate our model against a traditional approach, DBSCAN. These datasets pose similar challenges to those explored in the previous chapter, but in a 3D space. The datasets used are Hepta, Atom, Tetra, and Chainlink. Additionally, we modified Chainlink to further expand on the unique, interlocked structures proposed in that dataset. We explore these datasets given their noted challenges for clustering algorithms and the opportunity to test our approach with existing models on specific, curated problems. As in the previous chapter, we will introduce the datasets and their respective challenges, and then review quality metrics at the end of this section.

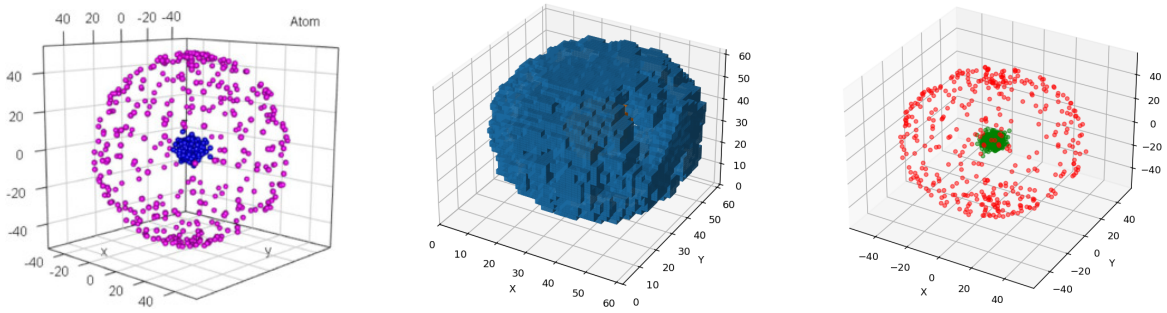


(a) Reference image (b) Results of Our Approach (c) Results of DBSCAN

Figure 5.2: Our Method Compared to DBSCAN - Hepta.

The first dataset, Hepta, comprises 212 points, partitioned into seven clusters with varying intracluster distances. One central cluster has a density almost twice that of the surrounding clusters and occupies less space than the other clusters. This dataset offers a challenge for nonoverlapping convex clusters in three dimensions. This is similar to the

training dataset, namely spherical cluster shapes with some distance between them. For this dataset, we use a ρ of 3 to expand the voxel footprint of each point and to connect the somewhat sparse clusters, each with only 30 data points. Our threshold is set to a 50% percentile of the heatmap output. Both our approach and DBSCAN correctly identified the number of clusters present. The results of this and a reference image are shown in Figure 5.2.



(a) Reference image (b) Results of Our Approach (c) Results of DBSCAN

Figure 5.3: Our Method Compared to DBSCAN - Atom.

The second dataset, Atom, consists of 800 points across two clusters. One, referred to as the core, is a densely populated cluster with 400 points, while the other, named the hull, contains the same number of points and completely encompasses the core. The challenge of this dataset is twofold. One is that the clusters are not linearly separable due to their nested structure. And secondly, the outer cluster density is much lower than the central cluster by several orders of magnitude. The outer cluster has a high inner-cluster variance between points compared to the distance between clusters. Our model again used a ρ of 3 and a 50% percentile threshold. Both our model and DBSCAN determined the correct number of clusters. Similar presentations used for other datasets are shown in Figure 5.3. It is worth noting that the central cluster is obscured in our voxel representation. However, by viewing slices of the activation map, we can observe those structures as they are read from the network at various depths, as shown in Figure 5.4. Similar views for other datasets are available in the Appendix.

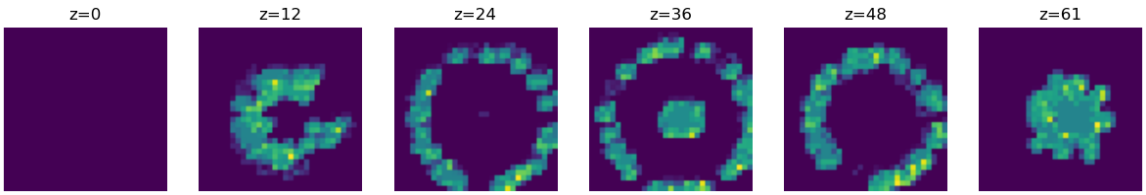
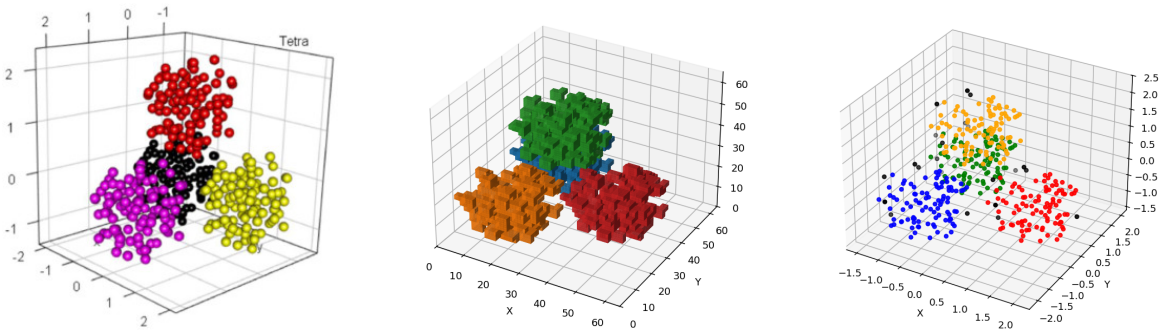


Figure 5.4: Activation map representation at various depths on the Atom dataset.



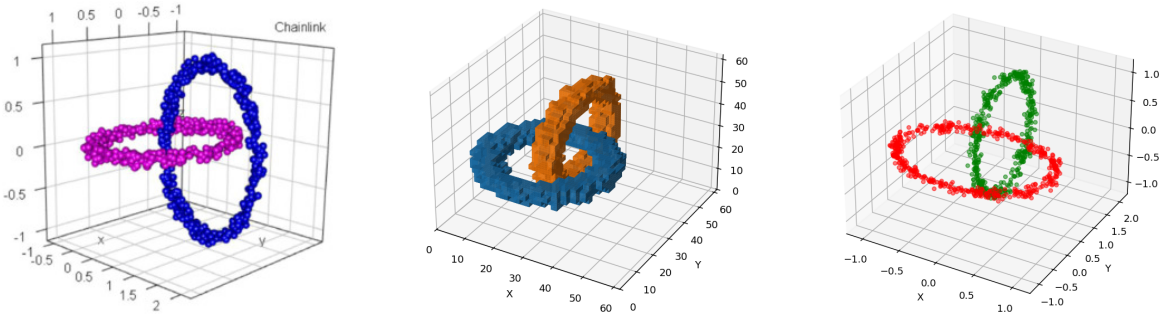
(a) Reference image (b) Results of Our Approach (c) Results of DBSCAN

Figure 5.5: Our Method Compared to DBSCAN - Tetra.

The next dataset used to evaluate our model is Tetra. This dataset contains 400 points across 4 spherical clusters that are in close proximity to one another. Although each cluster has relatively small intracluster distances, the intercluster distances are also small, resulting in the clusters nearly touching. As a result, the primary challenge in this dataset is correctly separating clusters that are spatially close and have similar densities. For this dataset, the parameter ρ was set to 0. This means that each point activates only a single voxel, without expanding to neighbouring voxels.

To compensate for the reduced spatial influence of each point, the heatmap threshold used during feature map extraction was lowered to 10%. This allows individual point activations to contribute more strongly to the grouping process, despite their smaller footprint in the voxel grid. Consequently, there is a trade-off between the voxel expansion parameter and the heatmap threshold when clusters lie in close proximity. Larger values of ρ increase the spatial influence of each point but risk merging nearby clusters, while lower heatmap thresholds increase sensitivity to individual points. For the Tetra dataset, we prioritized spatial flexibility to better preserve the separation between neighbouring clusters. Both our

model and DBSCAN can determine the correct number of clusters, but both require parameter tuning and still result in more outliers than in other, more well-separated datasets. These results are shown in Figure 5.5.

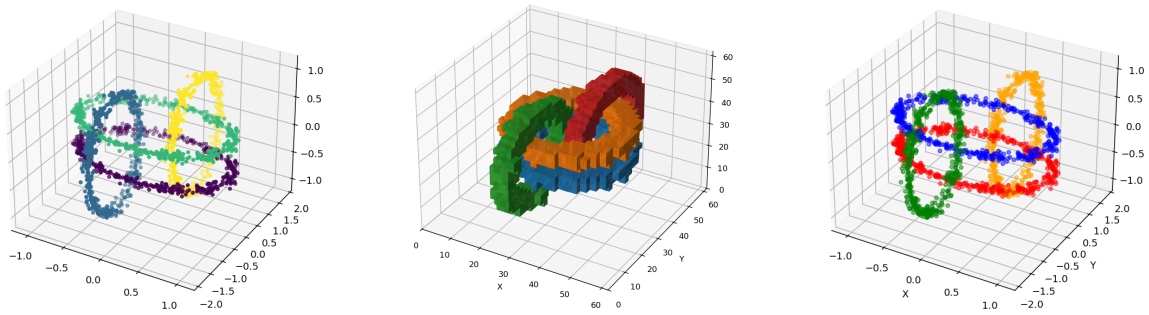


(a) Reference image (b) Results of Our Approach (c) Results of DBSCAN

Figure 5.6: Our Method Compared to DBSCAN - Chainlink.

The last dataset used from the Thrun and Ultsch benchmark set is Chainlink, which is a pair of 500 point clusters that form interlocking links of a chain. The challenge of this approach arises from the complex pattern between the two groups, namely, two clusters that intersect at each other's centres. This dataset is well suited to test density-based approaches ability to separate shapes in a 3D space. For this dataset, we set ρ to 1 to allow some voxel expansion, while not requiring much, since the data points are visually closely connected within the reference image. The threshold was set to 50% to be consistent with our other experiments, although in these examples, where the clusters have larger gaps between them, this value is inconsequential, as the gaps in the activation map will separate the cluster regardless of this value. Figure 5.6 shows a visualization of the reference image of the dataset, the voxelization of our results and the results of DBSCAN.

In order to further investigate our proposed approach, we modified the Chainlink dataset in include more rings. In this modification, we doubled the number of rings and therefore the number of data points, to a new total of $2,000$ points across 4 clusters. The purpose was to increase the number of rings that pass through one another. In this modified version, we have called Chainlink Ext (extended); each ring still consists of 500 points, but has



(a) Reference image (b) Results of Our Approach (c) Results of DBSCAN

Figure 5.7: Our Method Compared to DBSCAN - Chainlink Ext.

two rings passing through its centre. This increases the complexity of interaction between clusters while only duplicating and augmenting the original dataset. We continued to use a value of 1 for ρ, but needed to increase the activation-map threshold to 90% to prevent clusters from merging. These results are shown in Figure 5.7.

Table 5.1: Quality Comparisons on Figures 5.2, 5.3, 5.5, 5.6, and 5.7.

Dataset	Approach	Metric			
		Precision	Recall	F1 Score	Purity
Hepta	Our Model	1	0.991	0.995	0.995
	DBSCAN	1	1	1	1
Atom	Our Model	1	0.95	0.974	1
	DBSCAN	1	1	1	1
Tetra	Our Model	1	0.725	0.84	0.803
	DBSCAN	1	0.945	0.971	0.96
ChainLinks	Our Model	1	0.963	0.981	0.984
	DBSCAN	1	1	1	1
ChainLinks Ext	Our Model	1	0.981	0.99	0.986
	DBSCAN	0.996	0.996	0.996	0.996

The results of these experiments are convincing and encouraging, indicating that our proposed approach can interpret 3D data. We have not only demonstrated that our model can expand to volumetric inputs but also fully utilize the input shape of CNNs. We also maintained the performance of our previous model while increasing the complexity of the

input data and the structures that can be processed. The specific cluster quality metrics from the evaluated dataset in this chapter are shown in Table 5.1. We observe that DBSCAN continues to have a small advantage over our approach, scoring either the same or slightly higher on the included metrics. However, our model achieved these results without processing the raw distance calculations between points. Only scanning a small, constant-sized representation of the data, which is independent of the number of points to process.

5.6 Summary

This chapter made significant changes to how our data is prepared, processed by the CNN with an updated architecture, and expanded our SilhouetteGen approach to incorporate 3D data. In those changes, we adopted a voxel representation of our data to standardize the input, ensure our model stays adaptable to datasets of varying sizes. We also incorporated a point-scaling parameter ρ to aid processing of sparse datasets, where increased dimensionality increases the void space around individual points, requiring more data to define an area as dense, as in the case of dimensionality. Although there is a trade-off between the additional input parameter ρ and the need to adjust it for sparse datasets, the increased flexibility and the improved results when incorporating it into sparse datasets are worth balancing

The experiments in this chapter show that our model has adapted well to the addition of a third dimension and the new voxel based input. This is represented in its performance on the included clustering benchmark datasets. Cluster quality metrics, compared with DBSCAN, are also encouraging for our approach, which does not require pairwise distance calculations to determine the membership of points in a cluster.

Our 3D approach can be further extended by adopting different upscaling techniques, such as the deconvolution approach for 3D U-Net architectures. Replacing upscaling with a decoding/deconvolution step could allow more details to be captured in the heatmap and extend the information available within SilhouetteGen with context from earlier layers.

However, an analysis of the information gained relative to the increased computational cost would be something to consider. Other possible extensions of this approach include expanding the voxel information in the input channel layer. This could include colour or updating the voxel scaling to fade away from the true point rather than being a binary representation.

Chapter 6

Conclusion

6.1 Results and Discussions

Throughout our experiments, we demonstrate that it is feasible to generate cluster information via deep feature extraction with convolutional neural networks. In doing so, presenting models that are transferable and adaptable in processing new data and generating clusters from previous examples.

In Chapter 3, we introduced an initial approach that is trained on predictable cluster appearances and established the conceptual idea of extracting a feature map to locate and determine the shape of clusters. The accuracy of this approach was evaluated using labelled data from our synthetic dataset to assess the number of cluster predictions and the individual point placements.

Chapter 4 then extended our approach to density-based problems and simplified the architecture by removing sections of the CNN that did not provide additional information beyond what our SilhouetteGen algorithm could discover. In addition, we improved the representational capabilities of our approach by interpreting the algorithm's output as a point cloud of connected data points, rather than producing convex hulls to represent each cluster. Without retraining the model, we applied these adaptations to our clustering approach on benchmark datasets and evaluated our performance against a well-known and cited method, DBSCAN. While not always outperforming DBSCAN on the included benchmarks, our approach ranked highly in the included quality metrics and was encouraging, given that it groups points based only on visual observations of a projection of the dataset.

Finally, in Chapter 5, we extended our approach to use a volumetric 3D representation of the data via voxelization, thereby enabling the 3D data structure that CNNs can use as inputs. To incorporate this change, we generated new synthetic data to train the model. Once training was complete, we again pruned the network layers beyond the point at which we extract the information using the SilhouetteGen algorithm, which was also updated to process the additional dimension in the data. To mitigate data sparsity in the expanded input space, we introduced a parameter ρ that represents the expansion of data points in the voxelized representation of the dataset. This allows one point to have a larger footprint and to expand into neighbouring voxels. This allowed greater flexibility in the data our 3D model could process. We again tested the new approach on benchmark datasets, compared it with the existing DBSCAN approach, and obtained results consistent with our previous 2D approach.

Each phase of the research introduced new challenges that guided the iterative development of the approach. The investigation progressed from simple, predictable shapes represented in 2D images to more complex structures and patterns in the data that we embedded in a volumetric 3D representation via voxelization, enabling processing with our CNN-based approach.

6.2 Summary of Contributions

Our notable objectives and contributions presented in this research include:

- RO1: Demonstrating that meaningful cluster representations and information can be obtained from dataset visualizations without explicitly computing distances between individual data points, which is our primary contribution. This observation is significant because traditional clustering algorithms rely heavily on pairwise distance calculations through an unsupervised search of the data to determine group similarity. Although our proposed approaches do not outperform the existing clustering method in some cases, they are comparable in other cases.

- RO2: Avoided the possibility of poor centroid placement requiring multiple iterations of the same data to produce global optimal solutions. This is not a concern for our approach, which does not define its clusters by a centroid and only requires a representation or the data, which is supplied to the CNN once for processing. For parameter selection, we improve on some approaches by not requiring the number of clusters to be specified, using only a threshold to determine what constitutes a dense region, and a minimum cluster size to create, which is comparable to the two parameters in DBSCAN, ε and MinPts. This holds until 3D volumetric data is interpreted, at which point we introduce an additional parameter ρ used in the data-encoding stage.

- RO3: We establish that our approach is flexible in identifying clusters of arbitrary shapes and automatically determining the number of clusters present in a dataset. This was demonstrated in the experiments in Chapters 4 and 5, which label individual sections of the input space as belonging to distinct groups, similar to image segmentation, and therefore do not impose restrictions on the shapes or number of groups that can be distinguished.

- RO4: Through our experiments, we have observed that our CNN-based approach is flexible in the three key areas that were documented challenges for previous ANN-based clustering models, being plasticity, reliability and output restrictions. Plasticity, or the ability to adapt to changes in the size of the input, as our input is independent of the number of data points in a dataset. This is directly shown in Chapter 5 by doubling the data in the Chainlink dataset to create Chainlink Ext, along with other datasets of varying sizes throughout our experiments. Reliability, as our model consistently produces the same results, and is able to generate cluster information on a wide range of patterns, as shown throughout the benchmark datasets. Lastly, in removing any restrictions on the range of possible clusters that can be identified, which is explored in Chapters 4 and 5, using the pruned CNN architectures that no longer rely on the categorical output to determine the number of clusters discovered.

These findings suggest that CNN-based processing of dataset representations offers a promising alternative perspective on clustering, motivating further investigation into how deep feature extraction can be leveraged for unsupervised data analysis.

6.3 Future Directions

While the results obtained in this research addressed the primary goals of the study, they also highlighted several opportunities for further investigation. The proposed approach opens a number of potential research directions that may extend and refine this approach to clustering. Possible areas for future work include:

Exploring alternative visual encodings of dataset attributes to extend the number of attributes that can be interpreted. For example, varying the colour or shape of data points could allow additional attributes to be embedded within the representation. This approach may provide a mechanism for incorporating categorical variables or derived statistical features, similar to those used in grid-based clustering approaches, potentially improving the number of attributes that can be processed from the dataset representation. Another extension to visual encoding was discussed in Section 5.6, updating voxel scaling in the 3D model to make neighbouring points fade or decrease in scale as the distance increases. This approach would support the heatmap interpretation and could help fill some intra-cluster voids.

Removing the minSize parameter and attempting to merge small clusters with neighbouring groups could improve the accuracy of these approaches, and possibly even the need for the minSize parameter. It is likely that removing these small groups has led some points to be classified as outliers because their collections were too small to constitute their own cluster. Establishing a method to merge them into larger neighbouring groups could potentially improve the completeness of these groups in such cases.

Another approach to locating and filling gaps in point clouds could be to use a recurrent network. Where an input is processed by the CNN and with SilhouetteGen before being

processed again by a CNN. Another consideration is to apply a blur to the input to soften the cluster's edges. This could lead to clusters of similar density in close proximity merging. More consideration of the threshold value is needed, as it may need to change with subsequent passes.

Conducting a detailed analysis of outlier detection and robustness when applied to real-world datasets. While the experiments in this study focused primarily on synthetic benchmark datasets, evaluating performance on more complex real-world data would provide further insight into the approach's strengths and limitations. This would need to be done in conjunction with attribute ranking or dimensionality reduction techniques to ensure that the relevant information to group the data is present.

Performing a detailed analysis of the computational time and memory requirements associated with the 2D visualization and 3D voxelization processes used to prepare datasets, in addition to the requirements to process those representations once they are created. Because the proposed method produces visually interpretable inputs and outputs, further study could also examine its potential advantages in explainability relative to traditional clustering methods.

Investigating the use of alternative pre-trained convolutional neural network architectures, such as AlexNet or VGG, to replace the trained model used in this work and generate feature representations for SilhouetteGen. Leveraging established feature extractors may improve generalization and could fully remove training requirements, since this model does not rely on a categorical or standard network output. However, flexibility in data encoding may be limited, as these models have a predefined image input resolution.

Bibliography

[1] Mohamed Abdelnaby and Marmar R. Moussa. A benchmarking study of random projections and principal components for dimensionality reduction strategies in single cell analysis. *bioRxiv*, 2025.

[2] Charu C. Aggarwal. *Neural networks and deep learning: A textbook*. Springer International Publishing Springer, 2nd edition, 2023.

[3] Syed Fahim Ahmed, Fairuz Shezuti Rahman, Tasmia Tabassum, and Md. Tariqul Islam Bhuiyan. 3d u-net: Fully convolutional neural network for automatic brain tumor segmentation. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6, 2019.

[4] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods, 2018.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Symposium of Discrete Algorithms*, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[6] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2):261–287, August 2012.

[7] Eric Bair. Semi-supervised clustering methods. *WIREs Comput. Stat.*, 5(5):349–361, September 2013.

[8] Rabeya Basri, Mohammad Reduanul Haque, Morium Akter, and Mohammad Shorif Uddin. Bangla handwritten digit recognition using deep convolutional neural network. In *Proceedings of the International Conference on Computing Advancements*, pages 1–7. Association for Computing Machinery, 2020.

[9] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[10] James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2):191–203, 1984.

[11] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1), July 2015.

[12] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[14] Chollet Francois. *Deep learning with python, third edition*. Manning Publications, New York, NY, 3 edition, December 2025.

[15] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980.

[16] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness Paperback*. W.H. Freeman and Company, 1979.

[17] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.

[18] Aadya Goel, Pallavi Mishra, and Rachna Bhatia. Enhancing image feature matching detection: Orb and hdbscan algorithm integration. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–8, 2024.

[19] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325–328, July 1997.

[20] Michael Hahsler, Matthew Piekenbrock, and Derek Doran. dbscan: Fast density-based clustering with r. *Journal of Statistical Software*, 91(1):1–30, 2019.

[21] Jiawei Han, Jian Pei, and Hanghang Tong. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 4 edition, 2023.

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[23] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[24] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Science*, 622(C):178–210, April 2023.

[25] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[26] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks, 2017.

[27] Rohit Katyal and Neeshu Sharma. Performance analysis of standard k-means and enhanced k-means clustering algorithm. In *2024 12th International Conference on Intelligent Systems and Embedded Design (ISED)*, pages 1–5, 2024.

[28] P Kaufman and LKPJ Rdusseeun. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, volume 31, page 28, 1987.

[29] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. *q-means: a quantum algorithm for unsupervised machine learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[30] Dae-Won Kim and Kwang H. Lee. A new validity measure for fuzzy c-means clustering, 2024.

[31] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

[33] Levi Lelis and Jörg Sander. Semi-supervised density-based clustering. In *2009 Ninth IEEE International Conference on Data Mining*, pages 842–847, 2009.

[34] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 3rd edition, 2020.

[35] Rebecca Leygonie, Sylvain Lobry, Guillaume Vimont, and Laurent Wendling. Transforming multidimensional data into images to overcome the curse of dimensionality. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 700–704, 2023.

[36] Xinwang Liu. Simplemkkm: Simple multiple kernel k-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5174–5186, 2023.

[37] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[38] Changzhou Long, Yasunori Futamura, Xiucai Ye, and Tetsuya Sakurai. Quantum algorithm for regularized spectral clustering. In *Proceedings of the 2023 15th International Conference on Machine Learning and Computing*, ICMLC '23, page 5–11, New York, NY, USA, 2023. Association for Computing Machinery.

[39] Mahnoor, Imran Shafi, Mahnoor Chaudhry, Elizabeth Caro Montero, Eduardo Silva Alvarado, Isabel de la Torre Diez, Md Abdus Samad, and Imran Ashraf. A review of approaches for rapid data clustering: Challenges, opportunities, and future directions. *IEEE Access*, 12:138086–138120, 2024.

[40] Xiaoyi MAI and Romain COUILLET. Semi-supervised spectral clustering. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 2012–2016, 2018.

[41] Siti Ummi Masruroh, Asyifa Tasya Fadilah, Khodijah Hulliyah, Ahmad Fadlan Ramadhan, Rizka Amalia Putri, and Mohamad Ali Irfan. Implementation of the k-means clustering algorithm for targeting ads. case study: Ibm watson analytics car insurance customer data. In *2023 11th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–3, 2023.

[42] Samoilov Mikhail. 2d clustering dataset collection - an educational dataset for mastering clustering techniques. https://www.kaggle.com/datasets/samoilovmikhail/2d-clustering-dataset-collection, Jan 2025. Accessed: 2025-07-26.

[43] Seyed Mahdi Miraftabzadeh, Cristian Giovanni Colombo, Michela Longo, and Federica Foiadelli. K-means and alternative clustering methods in modern power systems. *IEEE Access*, 11:119596–119633, 2023.

[44] Thomas J. Misa. Understanding 'how computing has changed the world'. *IEEE Annals of the History of Computing*, 29(4):52–63, 2007.

[45] Mirtill-Boglárka Naghi, Levente Kovács, and László Szilágyi. A review on advanced c-means clustering models based on fuzzy logic. In *2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000293–000298, 2023.

[46] Mirtill-Boglárka Naghi, Levente Kovács, and László Szilágyi. A parameter selection strategy for the generalized fuzzy-possibilistic c-means algorithm. In *2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000533–000538, 2025.

[47] Arpita Nagpal, Arnan Jatain, and Deepti Gaur. Review based on data clustering algorithms. In *2013 IEEE Conference on Information & Communication Technologies*, pages 298–303, 2013.

[48] Mariá Nascimento, Franklina Toledo, and André Carvalho. A hybrid heuristic for the k-medoids clustering problem. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, page 417–424, New York, NY, USA, 2012. Association for Computing Machinery.

[49] Filip Novoselnik, Hrvoje Leventić, Irena Galić, and Danilo Babin. 3d u-net based method for fast segmentation of whole heart from ct images. In *2022 International Symposium ELMAR*, pages 159–164, 2022.

[50] Naaman Omar, Adel Al-zebari, and Abdulkadir Sengur. Improving the clustering performance of the k-means algorithm for non-linear clusters. In *2022 4th International Conference on Advanced Science and Engineering (ICOASE)*, pages 184–187, 2022.

[51] N.R. Pal, J.C. Bezdek, and E.C.-K. Tsao. Generalized clustering networks and kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):549–557, 1993.

[52] Chandro Pardede, Agusti Frananda Alfonsus Naibaho, and Frans Sitohang. Implementation of the k-means, k-medoids, and mini batch k-means algorithms on the classification of natural disaster prone areas in indonesia. In *2024 Ninth International Conference on Informatics and Computing (ICIC)*, pages 1–5, 2024.

[53] Yuting Peng. Exploratory data analysis and credit score classification for credit scoring based on improved artificial neural network models with channel attention mechanisms. In *Proceedings of the 2024 International Conference on Digital Society and Artificial Intelligence*, page 96–100, New York, NY, USA, 2024. Association for Computing Machinery.

[54] Van Phung and Eun Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9:4500, 10 2019.

[55] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 11 1958.

[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[57] Said Salloum, Dina Tahat, Khalaf Tahat, Raghad Alfaisal, and Ayham Salloum. Clustering medical transcriptions using k -means. In *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*, pages 291–294, 2024.

[58] Devin Schafthuizen and John Z. Zhang. Clustering through classifying: A novel neural-based approach. In *2025 18th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2025.

[59] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3), July 2017.

[60] Ayushi Sharma. From point clouds to voxel grids: A practical guide to 3d data voxelization, Aug 2025.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[62] Kowshick Srinivasan, Aakanksha Trivedi, and Shuhui Yang. Quantum-enhanced clustering for intrusion detection system in iot devices. In *2025 IEEE 22nd International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, pages 610–615, 2025.

[63] Artur Starczewski, Magdalena Scherer, Wojciech Książek, Maciej Debski, and Lipo Wang. A novel grid-based clustering algorithm. *Journal of Artificial Intelligence and Soft Computing Research*, 11:319–330, 10 2021.

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[65] Michael C. Thrun and Alfred Ultsch. Clustering benchmark datasets exploiting the fundamental clustering problems. *Data in Brief*, 30:105501, 2020.

[66] Neha Tyagi, Rashmi Mishra, Sitanshu Kumar Patel, and Somraj Karki. Performance enhancement of dbscan density-based clustering algorithm in data mining. In *2025 3rd International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC)*, pages 1341–1345, 2025.

[67] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, page 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[68] Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal, and James Foulds. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 5th edition, 2025.

[69] Qingyang Wu. A review of methods used in machine learning and data analysis. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, page 43–51, New York, NY, USA, 2019. Association for Computing Machinery.

[70] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016.

[71] Yaqiang Yao, Yang Li, Bingbing Jiang, and Huanhuan Chen. Multiple kernel k-means clustering by selecting representative kernels. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4983–4996, 2021.

[72] Rauf Yasin, Seçkin Ari, and Ammar Aljer. Dbscan-leak, novel real-time water leak detection: A case study of sunrise demonstrator. *IEEE Access*, pages 1–1, 2026.

[73] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. Self-supervised convolutional subspace clustering network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5468–5477, 2019.

[74] Rui Zhang, Hanghang Tong, Yinglong Xia, and Yada Zhu. Robust embedded deep k-means clustering. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1181–1190, New York, NY, USA, 2019. Association for Computing Machinery.

Appendix A

Appendix

Table A.1: Model Summary of our proposed CNN (Chapter 3).

Layer	Input Shape	Output Shape	Param #
Conv2d	[180, 180, 3]	[90, 90, 32]	1,568
Conv2d	[90, 90, 32]	[45, 45, 32]	16,416
batch-normalization	[45, 45, 32]	[45, 45, 32]	128
Conv2d	[45, 45, 32]	[23, 23, 64]	18,496
Conv2d	[23, 23, 64]	[12, 12, 64]	36,928
batch-normalization	[12, 12, 64]	[12, 12, 64]	256
Conv2d	[12, 12, 64]	[10, 10, 64]	36,928
Conv2d	[10, 10, 64]	[8, 8, 64]	36,928
spatial-dropout2d	[8, 8, 64]	[8, 8, 64]	0
batch-normalization	[8, 8, 64]	[8, 8, 64]	256
flatten	[8, 8, 64]	[4096]	0
dense	[4096]	[64]	262,208
dropout	[64]	[64]	0
dense	[64]	[32]	2,080
dropout	[32]	[32]	0
dense	[32]	[5]	165

Table A.2: Model Summary of the trimmed CNN (Chapter 4).

Layer	Input Shape	Output Shape	Param #
Conv2d	[180, 180, 3]	[90, 90, 32]	1,568
Conv2d	[90, 90, 32]	[45, 45, 32]	16,416

Algorithm A.1: Flood Fill Using Stack (Chapter 3)

Data: Binary grid G of size $m \times n$, start coordinate (x_0, y_0)
Result: List of connected points

```
1  stack ← [(x_0, y_0)];
2  points ← [ ];
3  while stack is not empty do
4  |   (x, y) ← pop(stack);
5  |   if G[x, y] = true then
6  |   |   append (x, y) to points;
7  |   |   G[x, y] ← false ;                          // mark as visited
   |   |   // Check 8-connected neighbours
8  |   |   if y < n − 1 then
9  |   |   |   push (x, y + 1) onto stack ;                    // up
10 |   |   end
11 |   |   if x < m − 1 and y < n − 1 then
12 |   |   |   push (x + 1, y + 1) onto stack ;          // upper-right
13 |   |   end
14 |   |   if x < m − 1 then
15 |   |   |   push (x + 1, y) onto stack ;                  // right
16 |   |   end
17 |   |   if x < m − 1 and y > 0 then
18 |   |   |   push (x + 1, y − 1) onto stack ;          // lower-right
19 |   |   end
20 |   |   if y > 0 then
21 |   |   |   push (x, y − 1) onto stack ;                    // down
22 |   |   end
23 |   |   if x > 0 and y > 0 then
24 |   |   |   push (x − 1, y − 1) onto stack ;           // lower-left
25 |   |   end
26 |   |   if x > 0 then
27 |   |   |   push (x − 1, y) onto stack ;                   // left
28 |   |   end
29 |   |   if x > 0 and y < n − 1 then
30 |   |   |   push (x − 1, y + 1) onto stack ;           // upper-left
31 |   |   end
32 |   end
33 end
34 return points;
```

Table A.3: Model Summary of our proposed CNN - Conv3D (Chapter 5).

Layer	Input Shape	Output Shape	Param #
Conv3d	[64, 64, 64, 1]	[62, 62, 62, 32]	896
max-pooling3d	[62, 62, 62, 32]	[31, 31, 31, 32]	0
Conv3d	[31, 31, 31, 32]	[29, 29, 29, 64]	55,360
max-pooling3d	[29, 29, 29, 64]	[14, 14, 14, 64]	0
Conv3d	[14, 14, 14, 64]	[12, 12, 12, 128]	221,312
max-pooling3d	[12, 12, 12, 128]	[6, 6, 6, 128]	0
flatten	[8, 8, 64]	[27648]	0
dense	[27648]	[512]	14,156,288
dropout	[512]	[512]	0
dense	[512]	[256]	131,328
dropout	[256]	[256]	0
dense	[256]	[128]	32,896
dropout	[128]	[128]	0
dense	[128]	[64]	8,256
dense	[64]	[32]	2,080
dense	[32]	[16]	528
dense	[16]	[5]	85

Table A.4: Model Summary of the trimmed CNN - Conv3D (Chapter 5).

Layer	Input Shape	Output Shape	Param #
Conv3d	[64, 64, 64, 1]	[62, 62, 62, 32]	896
max-pooling3d	[62, 62, 62, 32]	[31, 31, 31, 32]	0

Algorithm A.2: Flood Fill 3D Using Stack in (Chapter 5)

Data: Binary grid G of size $m \times n \times p$, start coordinate (x_0, y_0, z_0), boolean flag include_corners

Result: List of connected points

stack $\leftarrow [(x_0, y_0, z_0)]$;
points $\leftarrow [\,]$;
if *include_corners* $=$ *true* **then**
    `// 26-connected neighbourhood`
    neighbour_offsets $\leftarrow \{(d_x, d_y, d_z) \mid d_x, d_y, d_z \in \{-1, 0, 1\},\ (d_x, d_y, d_z) \neq (0, 0, 0)\}$;
else
    `// 18-connected neighbourhood`
    neighbour_offsets $\leftarrow \{(d_x, d_y, d_z) \mid d_x, d_y, d_z \in \{-1, 0, 1\},\ (d_x, d_y, d_z) \neq (0, 0, 0),$
        $\text{not}(d_x \neq 0$ **and** $d_y \neq 0$ **and** $d_z \neq 0)\}$;
end
while *stack is not empty* **do**
    $(x, y, z) \leftarrow \text{pop}(\text{stack})$;
    if $G[x, y, z] =$ *true* **then**
        append (x, y, z) to points;
        $G[x, y, z] \leftarrow$ false ; `// mark as visited`
        foreach $(d_x, d_y, d_z) \in$ *neighbour_offsets* **do**
            $n_x \leftarrow x + d_x$;
            $n_y \leftarrow y + d_y$;
            $n_z \leftarrow z + d_z$;
            if $0 \leq n_x < m$ ***and*** $0 \leq n_y < n$ ***and*** $0 \leq n_z < p$ **then**
                push (n_x, n_y, n_z) onto stack;
            end
        end
    end
end
return *points*;

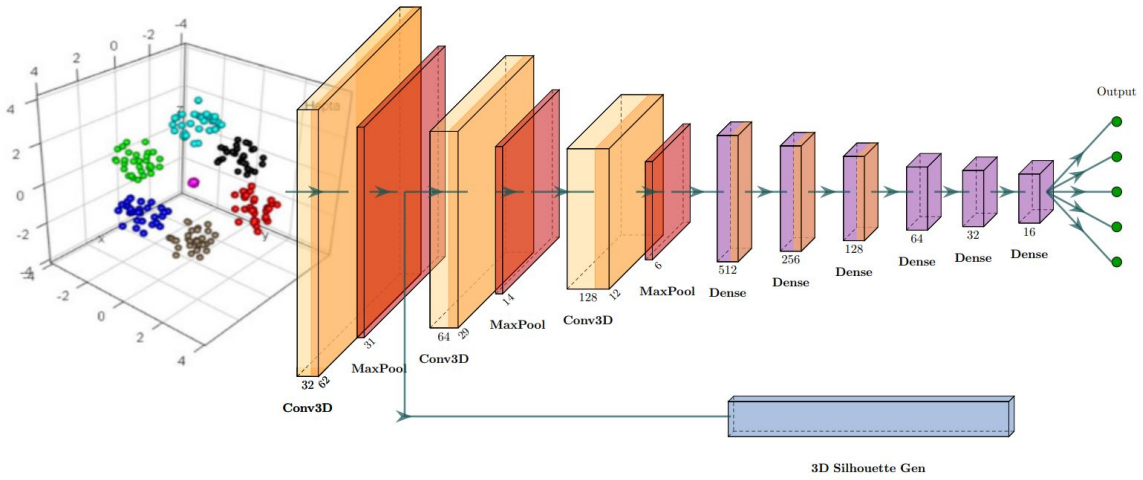


Figure A.1: Full architecture of our 3D CNN-based Model used for training (Chapter 5).

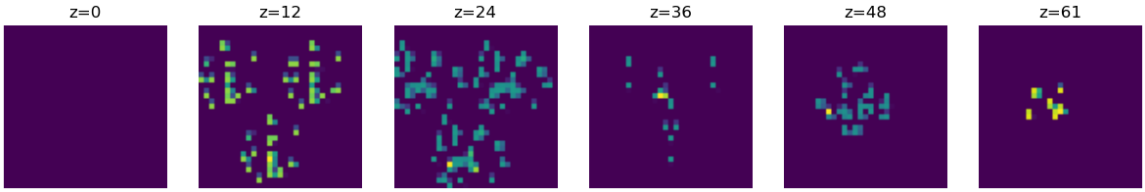


Figure A.2: Activation map representation on the Tetra dataset (Chapter 5).

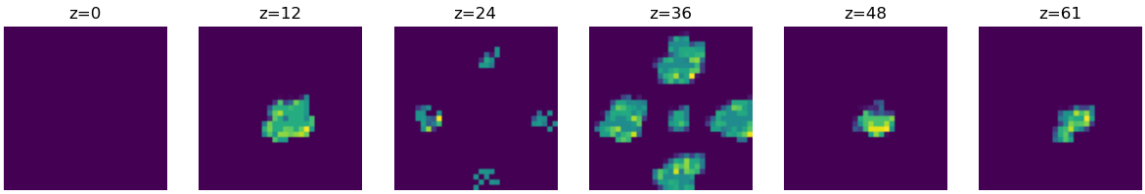


Figure A.3: Activation map representation on the Hepta dataset (Chapter 5).

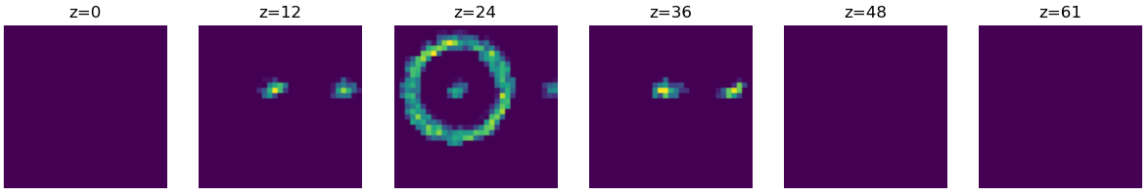


Figure A.4: Activation map representation on the Chainlink dataset (Chapter 5).

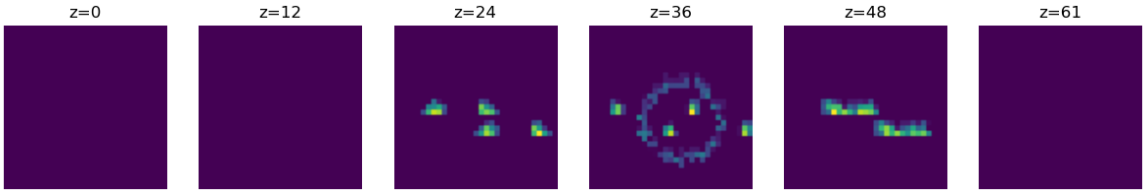


Figure A.5: Activation map representation on the Chainlink Ext dataset (Chapter 5).