Enhancing Spike System through Advanced Feature Extraction and Clustering Algorithms on High-Density Multi-Electrode Arrays

Aakansha Ramesh

Master of Science
Computer Science
School of Informatics
University of Edinburgh
2024

Abstract

Advancements in neurotechnology have enabled the recording of neuronal activity with unprecedented detail, using high-density multi-electrode arrays (HD-MEAs). Spike sorting, the process of distinguishing and categorizing neuronal action potentials, is pivotal for interpreting these complex signals. This dissertation investigates various feature extraction and clustering methods to enhance spike sorting accuracy in neural data obtained from HD-MEAs. Techniques such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Uniform Manifold Approximation and Projection (UMAP), and Isometric Mapping (Isomap) were evaluated alongside clustering algorithms including K-means, Agglomerative Clustering, HDBSCAN, and Mean Shift. Systematic parameter tuning and rigorous performance evaluation, including accuracy, precision, and recall measures, reveal that combining HDBSCAN with PCA offers superior performance with an accuracy of 0.87, by efficiently minimizing noise while preserving essential data features. The study also considers the computational complexities of these methods, ensuring a balance between effectiveness and efficiency. The findings highlight the importance of choosing the appropriate methods and parameters, that can guide future developments in brain-machine interfaces and neurological research.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Aakansha Ramesh*)

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr Matthias Hennig, for his invaluable guidance, support, and encouragement throughout this dissertation. His insightful feedback and unwavering patience played a significant role in helping me complete this work well within the time frame.

Table of Contents

1 Introduction **1**

2 Background **4**

2.1 Understanding Neural Activity 4

2.2 Spike Sorting Pipeline 6

2.2.1 Pre-Processing 6

2.2.2 Spike Detection 7

2.2.3 Feature Extraction 8

2.2.4 Clustering 10

2.2.5 Post-processing 10

3 Related Works **12**

3.1 Template Matching 12

3.2 Bayesian Clustering and Classification 13

3.3 Principal Component Analysis (PCA) 14

3.4 Independent Component Analysis (ICA) 15

4 Methodology **17**

4.1 Software Dependencies and Toolkits 17

4.2 Ground Truth Generation 18

4.3 Feature Extraction 18

4.3.1 Principal Component Analysis 20

4.3.2 Independent Component Analysis 20

4.3.3 Uniform Manifold Approximation and Projection 21

4.3.4 Isometric Mapping 21

4.4 Clustering 22

4.4.1 Density-Based Spatial Clustering of Applications with Noise . 23

4.4.2 Agglomerative Clustering 23

4.4.3 K-Means Clustering 24
4.4.4 Hierarchical Density-Based Spatial Clustering of Applications with Noise 25
4.4.5 Mean Shift Clustering 25
4.5 Performance Evaluation 26

5 Results **28**
5.1 Challenges with Spike Data 28
5.2 Parameter Tuning 30
5.3 Performance Evaluation 31

6 Discussions **34**
6.1 Computation Complexities 35
6.2 Limitations and Future Scope 38

7 Conclusion **40**

Bibliography **42**

A Appendix **49**
A.1 Spike Sorting and Influence on Brain-Computer Interfaces 49
A.2 Wavelet-based feature extraction 50
A.3 Spectral Clustering 52
A.4 Justification of Principal Component Selection Using Reconstruction Error Analysis 53

B Additional Results **55**
B.1 Agglomerative Clustering 55
B.2 DBSCAN Clustering 56
B.3 HDBSCAN Clustering 58
B.4 K-Means Clustering 60
B.5 MeanShift Clustering 61

Chapter 1

Introduction

The brain is made up of a vast network of specialized cells called neurones or neurons. These neurons are responsible for the daily functioning of our lives. Advances in technology have enabled us to record the activity of these cells and specifically the electrical impulses they generate called spikes. Spikes or action potentials are the electrical impulses generated by neurons when they communicate, this indicates the transmission of information. We can think of spike as a notification on a cellphone. When the mobile receives a message, it makes a sound to alert the user. Similarly, when a neuron fires a spike, it's similar to sending a signal to the brain to alert it about an event, such as sensing a touch, hearing a sound, or seeing an object. Just like the notification sound alerts a person to check the message, spikes alert different parts of the brain to process information and respond [1].

The frequency and pattern of spikes convey information that the brain uses to process sensory signals, make decisions and control motor functions. Analysis of spikes can help us design the Brain-Machine Interface (BMI) where real-time intercommunication between each neuron is required for predicting motor movement and possibly complex neural processes involved in decision-making and sensory perception [2] (elaborated in detail in Appendix A.1).

The extraction of spikes is made possible by a device known as the high-density (HD) multi-electrode array (MEA). The HD-MEA consists of a grid of closely spaced electrodes that simultaneously record many neurons' activity. They are placed in close contact with the neural tissue, where they detect extracellular electrical signals generated by neurons. HD-MEAs have a much higher density of electrodes, allowing for more precise spatial mapping of neuronal activity. These arrays are typically placed in close contact with neural tissue, enabling the recording of extracellular electrical signals

generated by neurons. These signals can be analyzed to study brain functions and neural networks in detail [3].

Spike sorting is a Machine Learning technique for monitoring, capturing, and processing multiple neuron's activity simultaneously. This technique allows for the dissection of activities from several neurons situated in juxtaposition with each other, which with further processing can help to obtain an individualized view of the activity of a single cell. The main challenge lies in analyzing neural data, as neurons are extremely small. Even with closely spaced recording sites in HD-MEAs (extracellular electrodes), the tiny size of neurons results in multiple channels recording spikes from several neurons at once [4]. As a result, when recording neural signals, a single electrode may pick up spikes from multiple neurons at the same time, leading to overlapping signals. This overlap makes it difficult to accurately identify which neuron each spike belongs to.

Another issue with processing neural data is that, given the billions of neurons in the brain, each producing their respective spikes during the triggering of a particular activity, it becomes increasingly difficult to decipher whether the spikes detected by multiple channels are associated with different neurons or the same neuron [5].

To address the aforementioned issues, feature extraction is implemented to handle the complexities pertaining to high-dimensional neural recordings. Since neurons generate overlapping spikes for closely spaced recording sites, feature extraction can assist in separating and identifying these overlapping spikes by reducing the data dimensionality [6]. This facilitates a clear distinction between signals from different neurons. By focusing on the most informative features (like spike time, amplitude, shape and spike patterns), feature extraction can enhance the accuracy of spike sorting and result in a more optimised spike analysis system. Here, features represent the characteristics of spikes recorded from neurons.

For instance, analyzing the location and shape features of the spike waveform might reveal important details about neural activity. A spike waveform is the visual representation of the electrical potential (spike) generated by a neuron when it fires. It captures the shape of the spike over time, including its amplitude, duration, and specific phases such as the rise, peak, and fall. The location feature refers to the spatial position of the neuron within a small volume of brain tissue, which is crucial for distinguishing spikes from different neurons. By identifying these key features, we can enhance data representation and reduce computational requirements for analysis [5].

This leads to addressing the following research questions:

- How can spike data be efficiently analyzed to reveal distinct patterns?
- What methods are potentially reliable for categorizing neural activity based on spike data analysis?

Feature extraction followed by clustering of spike data can enhance our understanding of neural recordings because it empowers us to isolate and analyze the behaviour of each and every neuron signal from the HD-MEA recordings. This process can aid in reducing the noise and redundancy in spike data and can make it easier to understand the interrelationship in neurons. Application of dimension reduction techniques can compress the data while ensuring that essential information is not lost, this results in a faster and more efficient clustering system (grouping of data that exhibit similar traits). Thus optimising the spike sorting mechanisms (study of neural activity at the single-cell level) by increasing the accuracy and speed at which data is analyzed, potentially addressing major challenges in neuroscience [7].

The goal of this dissertation is to explore different dimension reduction techniques and understand how they can improve the clustering at different parameter settings and ultimately contribute towards more effective analysis of neural data. This is important as it enables researchers to process large volumes of neural data efficiently and get more insights into the operation of the brain and the development of cutting-edge neurological systems.

The upcoming section includes the Background section, which offers insights into the general understanding of the project and its context. The Related Work section reviews relevant existing literature about the project. The Methodology section expands on the approach that has been employed for this dissertation. The Results section describes the outcomes observed and the Discussions section analyzes the implications, and relevance of the results, highlights the limitations of the work, and the recommendations for further research. Lastly, the Conclusion section provides an overview of the major findings of the study.

Chapter 2

Background

This chapter lays a foundation on neural terminologies and the methods of spike sorting implemented in this dissertation. It emphasizes the relevance of spike sorting in understanding neural activity, providing essential background knowledge for the subsequent sections.

2.1 Understanding Neural Activity

To achieve a holistic understanding of the brain, it is essential to distinguish and classify the activity of every neuron from the extracellular recordings. This empowers scientists and researchers to understand spikes, offering valuable insights into how neurons transmit signals, analyze information, and contribute to diverse brain functions and behaviours.

It is crucial to understand how spikes are generated to grasp their importance. When a neuron receives input through structures called synapses (sites where signals are transmitted between cells) it can lead to an accumulation of these inputs. If this accumulation causes the membrane potential (the electrical charge difference across the neuron's membrane) to exceed a specific threshold, the neuron responds by generating a brief, transient electrical pulse known as a spike or action potential. The neuron essentially integrates excitatory and inhibitory signals from multiple synapses, and if the summed input surpasses this threshold, the neuron fires, similar to how a McCulloch-Pitts neuron model operates [8].

Figure 2.1 demonstrates the activity of a spiking neuron. The neuron receives stimuli as represented in coloured arrows, in response to this a bump or membrane voltage is observed. Membrane voltage occurs when the difference in the electric

potential between the external and internal environment of the cell is beyond the neuron threshold(ϑ, represented in dotted lines). This induces an exchange of ions across the cell membrane thereby resulting in the generation of spikes. After the occurrence of the action potential, the neuron enters the refractory phase where the chances of another spike occurring soon after are unlikely [9].

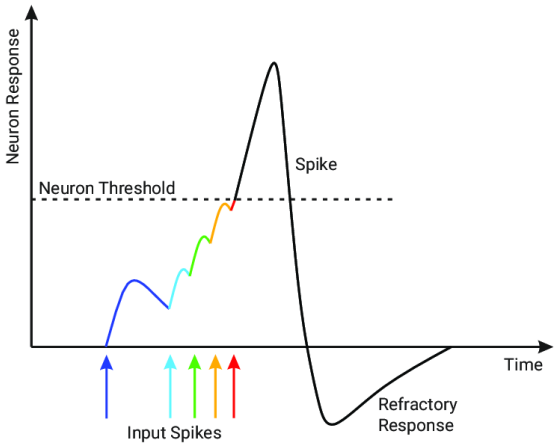

Figure 2.1: Illustration of the dynamic behaviour of a neuron that generates spikes. Source: Adapted from *Scholarpedia* article [9].

Modern technology has made it possible to record these neural impulses through devices like multi-electrode array (MEA). MEAs comprise thousands of microscopic electrodes spanning a small surface in each MEA plate or chip. When the cells exhibit electrical activity, the spikes are captured by each electrode on a microsecond time frame. Thereby, recording both temporally and spatially accurate data [3].

One issue that arises from the extracellular recordings of the MEAs is that they typically encompass the neural activity of multiple neurons simultaneously, making it difficult to separate the potentials of individual neurons. Despite capturing a vast amount of data, no meaningful insights can be observed. This calls for a system that studies the spike shape, firing rate, timing, and pattern of neural electrophysiological activity. Spike sorting is one such neuroscience technique implemented to record and analyze the collective electrical activity of a group of neurons, known as ensemble neural activity. This technique can identify and classify the electric potentials of neurons. They detect, identify, and isolate the neural potentials generated, and distinguish them from the background noise and with each other, even when multiple neurons are recorded simultaneously on the same electrode [5, 4].

2.2 Spike Sorting Pipeline

The traditional spike sorting pipeline consists of a series of stages to identify and classify the extracellular spike records generated by each neuron while being able to separate them from recordings of neighbouring cells and noisy signals. It aims to reconstruct the spike trains (series of action potentials generated by a specific neuron over a time frame) of all neurons with a clear signal absent of noise. It can be thought of similar to a scenario where we have an orchestra with multiple instruments playing concurrently; spike sorting is analogous to listening to the orchestra and being able to identify which symphony originates from a particular instrument, the spike train here is the instrument's sound over an interval. The general pipeline was derived from the articles published by Alessio P. Buccino et al. and David Carlson et al. [7, 10], as illustrated in Figure 2.2.

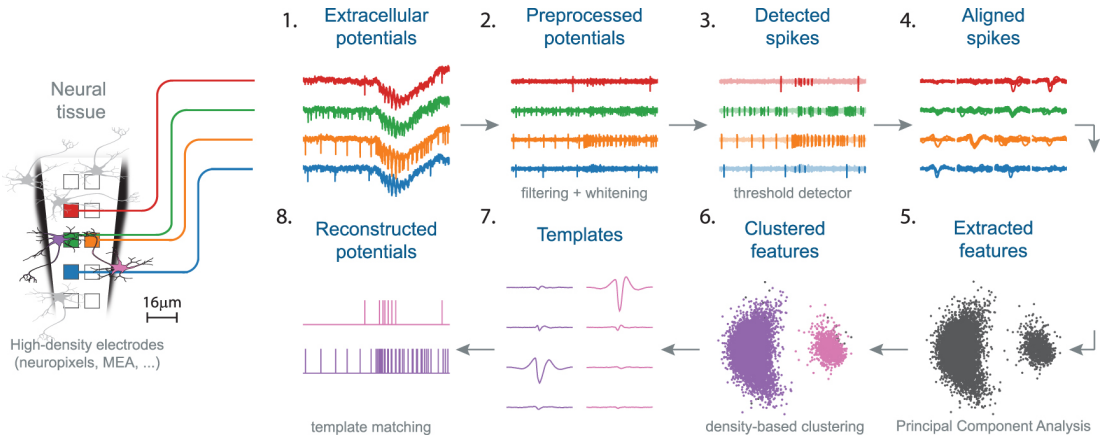

Figure 2.2: The general pipeline for spike sorting. Source: Adapted from Alessio P. Buccino et al. [7].

2.2.1 Pre-Processing

The pre-processing step is essential as it prepares the raw electrophysiological recordings for the subsequent stages. This step involves transforming and sorting the data to ensure accurate detection and classification of the spikes.

The first step involves passing the raw data through a band-pass filter; this ensures that low-frequency data such as drifts, and high-frequency noise are separated, retaining only the data of interest. Drifts are the recordings observed when there is a shift in the position of the electrode during the spiking action of a neuron; this external interference can result in an inconsistent shape or amplitude of the recorded spike waveform. Effective correction is necessary to ensure that the spike recordings can be

accurately attributed to the appropriate neuron. Following band-pass filtering, spatial whitening is performed. Whitening separates the signals from the band-pass filter of different electrodes and normalizes their variance. This is done to effectively minimize the overlap of spikes from nearby neurons, ensuring that all signals contribute equally [7].

Lastly, Common Average Reference (CAR) removal is performed. An average of all the signals is subtracted from the signals of each of the electrodes. This normalization step ensures that we isolate the true signals by removing the common-mode noise, a noise that is consistent across all the recording electrodes due to the presence of external or systemic issues [7, 10].

2.2.2 Spike Detection

Once the recordings are pre-processed, the detection of spike potentials is executed. A threshold detector is used, and spikes are detected when the potential exceeds a given amplitude threshold. The threshold parameter 'Thr' is set based on the Median absolute deviation (MAD) [11]. The detection threshold is typically established as a multiple of MAD, a robust estimator of noise variability (σ_n) (Equation 2.1 and Equation 2.2). Commonly, this threshold is established within the range of 3 to 5 times MAD, with $5\times$ MAD being frequently used to balance sensitivity to spikes against minimizing noise, corresponding to a false positive rate of about 1 spike per second per channel under typical noise conditions [7].

$$\text{Thr} = 5\sigma_n \tag{2.1}$$

$$\sigma_n = \text{median}\left\{\frac{|x|}{0.6745}\right\} \tag{2.2}$$

Here, σ_n is derived from the MAD of the recorded signal values (x), scaled by the constant 0.6745. This approach provides a percentile-based measure of variability around the median, offering a more reliable threshold in noisy conditions compared to the standard deviation, which is less robust to outliers [7].

However, the detected spikes tend to be misaligned with time because depending on the spike shape or the noise ratio, the precise point at which they cross the threshold varies. Thus, resulting in spikes being detected at varying points on the waveform. A temporal alignment is performed where a feature (like the spike shape) is slightly adjusted to a point where all the detected spikes occur at the same point in time. The aligned spikes maintain consistency across all the detected spikes [7].

2.2.3 Feature Extraction

In this step, we extract the features that best represent the spikes. These features should have the ability to provide an optimal separation between the different clusters. Additionally, the extracted features should be able to eliminate or separate the noise.

Dimensionality reduction is performed before the feature selection and extraction. Retention of the most significant features is done while simultaneously lowering the number of features or dimensions. This ensures the system is computationally efficient and prevents the subsequent steps from being overloaded with large volumes of input data. The goal is to eliminate the dimensions redundant with noise and improve the scope of clustering [12].

To select the features, it's important to understand the characteristics of the spike. These characteristics could be the spatial location of the spike, the amplitude, the shape, and the associated waveform, to name a few. Spike location is the spatial positioning of the neuron that generates the spike in a small volume of brain tissue, recorded by electrodes adjacent to one another. Studying them can give a better understanding of the interactions of neurons. Amplitude refers to the intensity of the spike when the neuron fires. Higher the amplitude, higher the neural activity. Amplitude shape tells about the structure of the spike waveform, this could be the rise, peak and fall of the waveform. All these features can help distinguish neurons and accurately sort spikes during analysis [13].

As detailed by Rui-Qi Song et al. [14] Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of the spike waveform data. PCA transforms the waveform, which typically consists of many samples, into a set of orthogonal components that capture the most variance within the data. This process simplifies the data while preserving the most important features. PCA is an eigendecomposition of the covariance matrix of the data, identifying the directions (principal components) that capture the most variance. This allows for effective dimensionality reduction while preserving the most significant features. Other feature extraction techniques include Uniform Manifold Approximation and Projection (UMAP) and Isometric Mapping (Isomap).

UMAP is a dimensionality reduction technique used to project data into lower dimensional space, this can assist in better visualization and analysis of multidimensional data. It aids in maintaining non-linear relationships that can uncover the underlying structure of data [15].

On the other hand, Isomap is a non-linear dimensionality reduction method that preserves the intrinsic structure of data by projecting the high-dimensional space into a low-dimensional manifold. It achieves this by calculating distances between points using geodesic distances, which represent the shortest paths on a curved surface or manifold rather than relying on straight-line (Euclidean) distances [16]. The Isomap algorithm first constructs a neighbourhood graph of features and then computes the geodesic distances between all points within the graph. Using multidimensional scaling (MDS), the data is projected into a lower-dimensional manifold while maintaining the same geodesic distances as in the high-dimensional space. This ensures that the intrinsic structure of the features is preserved in the reduced space.

Figure 2.3, as described by Graf et al. [17], illustrates the process of extracting features for spikes. An instance of a particular spike waveform is shown in Panel 'A', representing the raw data from which features are derived. Panel 'B' demonstrates a range of extracted features, such as peak-to-peak amplitude, maximum and minimum peaks, slope, duration, and area under the curve (AUC). These features offer an in-depth description of the spike's shape and characteristics. In Panel 'C', the result of applying a dimensionality reduction technique (in this case, PCA) to the extracted features is presented. The plot depicts the distribution of spikes in the reduced feature space, with clear clusters indicating spikes from different neurons. This visual depiction emphasizes the significance of feature extraction and dimensionality reduction in accurately recognizing and categorizing neural spikes.

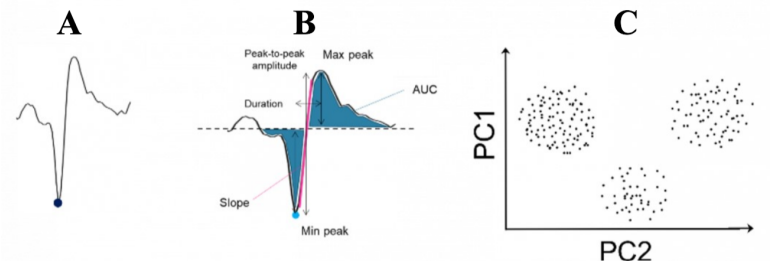

Figure 2.3: Illustration depicting the process of extracting features from spikes, showcasing the distinctive waveform properties of spikes and their clustering in a reduced feature space. Source: Revolutionizing CNS Drug Discovery Research with Cutting-Edge Technology and Innovative Analysis Tools [17].

2.2.4 Clustering

Now in this reduced feature space, spikes are arranged into clusters, such that each cluster is associated with spikes belonging to a single neuron. This density-based clustering process isolates the spike events into distinct groups by identifying areas with high concentrations of similar spikes, based on the density of points in the feature space. Each cluster centroid represents the average spatial and temporal pattern observed on the channels (an electrode or sensor in a MEA that detects neural activity) when a specific neuron generates a spike. This critical step in spike sorting distinguishes the spike trains of individual neurons from the combined signals recorded by electrodes, ensuring precise identification and categorization of neural activity [18].

Some clustering methods that can be implemented are K-means, DBSCAN, Mean-Shift clustering, Agglomerative clustering, and HDBSCAN clustering. The choice of clustering method largely depends on the characteristics of the neural recording, such as the number of neurons, template waveform, spike shapes, and computational capabilities. For example, K-means requires specifying the number of clusters in advance, which can be problematic if the exact number of neurons is unknown. DBSCAN does not need the number of clusters but relies on density, making it effective for varied spike shapes. Mean-Shift automatically determines the number of clusters based on data density, useful when the number of neurons is uncertain. Agglomerative clustering does not require the number of clusters but is computationally intensive. HDBSCAN clustering is effective for complex shapes but may require specifying the number of clusters and is also computationally demanding [19, 20, 21, 22, 23].

2.2.5 Post-processing

After the clustering stage, the spike sorting system might terminate with the clusters being assigned as units, or it can be continued by adding a template-matching step. This additional step works by creating an average spike waveform (template) for each cluster and then matching new incoming spikes to these templates. The advantage of this method is that it helps to resolve collisions (overlapping spikes) by accurately reconstructing the signal as a linear combination of the identified templates, particularly improving the detection of spikes with low signal-to-noise ratio (SNR). SNR is a metric used to compare the linearity of a signal with the noise; low SNR indicates that noise is more prominent compared to the spikes [7].

Spikes that are newly detected are matched with the closest template, which helps

improve the accuracy of spike sorting by taking into account overlapping spikes and background noise. Utilizing templates for matching improves the dependability of pinpointing spikes from the identical neuron in various recording sessions, thereby maintaining consistent and precise spike classification [7].

The reconstructed spikes as seen in Figure 2.2 step 8 are obtained by positioning the raw detected spikes (indicated by vertical lines) with the templates as observed in step 7. Each detected spike is matched with the nearest template, and the signal is reconstructed by summing these template matches. Certain spike sorting algorithms incorporate iterative refinement stages where templates and clustering results are continuously updated to enhance the accuracy of the sorting process.

Finally, validation is performed, and metrics like SNR and the distribution of inter-spike intervals are employed to evaluate the quality of the sorted spikes and the performance of the sorting algorithm. Usually, actual data does not provide a ground truth about what is correct or incorrect, making it challenging and subjective to evaluate the effectiveness of various spike sorting systems [24].

Chapter 3

Related Works

This chapter elaborates on some of the current implementations of varying techniques in spike sorting, highlighting the drawbacks of each method and discussing the advancements made to address these issues. The aim is to provide a comprehensive overview of the evolution of spike sorting techniques and the solutions developed to overcome the limitations of previous systems. This sets the stage for understanding the progression and current state of spike sorting technology.

3.1 Template Matching

One of the earliest implementations of spike sorting was initiated by G.L. Gersteiand and W.A. Clark in 1964 [25]. It initiated extensive research on spike patterns of multiple neurons. A tungsten-structured micro-electrode was used, the electrode's shaft had tiny holes that supported the recording of signals from neurons that were within proximity to each other.

In this algorithm, the user selected a characteristic spike shape for each cluster, and the remaining spikes were then successively assigned by comparing with the selected standardized waveform using the mean square distance measure. Here, a dissimilarity value was estimated. Its value ranged from zero onwards, where zero indicated the highest similarity to the standardized waveform, and higher values suggested more disparity. Following waveform matching, sorting was performed, and computers were used to automate the clustering of waveforms based on the previously calculated metric value. An innovative approach was used to reduce the extent of overlapping clusters. This was achieved using an ”iterative refinement” stage where waveforms most similar to one another were averaged to produce a new standardized waveform for further

sorting [25].

This method, while foundational for advanced spike sorting techniques, was limited by its reliance on user-mediated template selection, making it impractical for large datasets. Challenges in identifying representative templates in cases of overlapping neuronal spike shapes further reduced its reliability. Modern systems now employ unsupervised techniques to automate template matching and validation, enhancing efficiency and accuracy [25].

3.2 Bayesian Clustering and Classification

Gray et al. clustering system implemented the use of tetrodes to improve the separation of readings in an extracellular recording of a cat's visual cortex. Tetrodes (consisting of four electrodes) were placed in close proximity. Different channels associated with each tetrode generated the readings. Once the spikes were recorded, the raw waveforms were subjected to PCA, and clusters were created based on the principal components of spike amplitude and waveform. Cluster separation is completely dependent on the subjective view of the tester, requiring a human intervention to manually assign clusters. Thus variability arose mainly because the definition of clusters and how the clusters were cut for different systems varied for each user [26].

Harris et al. [27] enhanced spike differentiation by using tetrodes to capture the spikes' spatial locations and it required manual clustering through an interface called 'gclust,' which allowed users to reallocate spikes. Despite the precision this approach offered, it was time-consuming and susceptible to errors due to the limited dimensionality of the visual clustering space and inherent human biases. The testers often struggled to accurately define cluster boundaries in a high-dimensional feature space, leading to significant errors. Nonetheless, compared to the system by Gray et al. [26], Harris et al. [27] introduced a semi-automated feature, 'AutoClass,' which initially automated the clustering process. This was followed by human intervention for verification, effectively combining the strengths of both automated and manual clustering methods.

Michael S. Lewicki's [28] clustering system addressed the limitations of manual spike sorting by employing a Bayesian probabilistic model. Utilizing a glass-coated platinum-iridium electrode in the zebra finch nucleus lMAN, Lewicki modelled spike waveforms with a recursive linear function and Gaussian noise. This approach applied a Bayesian method to determine the most plausible spike parameters, using a multivariate Gaussian to model each cluster and calculate the probability of a data point's cluster

membership. This system facilitated clear boundary separation for classification and managed overlapping spikes to some degree by treating them as outliers. Key challenges tackled included defining the shapes of action potentials (APs), determining the number of distinct AP shapes, and addressing issues with overlapping spikes [28].

The model assumes that clusters follow a Gaussian distribution, asserting that variability in spike shape within a cluster is primarily due to Gaussian-distributed background noise. While this assumption holds to some extent, Fee et al. [29] contend that noise cannot always be strictly represented as Gaussian due to factors like electrode drift, overlapping spikes, multi-cellular activity, and detection errors, which can lead to non-Gaussian cluster distributions. Both Lewicki et al. [28] and Harris et al. [27] systems also struggle with handling non-Gaussian clusters. This limitation stems from their reliance on the assumption that feature vectors, which represent data points, adhere to a normal (Gaussian) distribution. However, this assumption fails in scenarios where spikes from different neurons overlap simultaneously, causing significant deviations from the Gaussian model. Such overlap complicates accurate spike clustering due to these deviations from the expected distribution pattern.

Despite this when compared to the template matching described in Section 3.1, Bayesian methods implemented a probabilistic approach that minimized the extent of overlapping spikes by treating them as outliers.

3.3 Principal Component Analysis (PCA)

The “Multispike Train Analysis” proposed by Abeles et al. [30] used PCA for the projection of the spike signals to its principal components. The goal of the project was to create clear and distinct clusters of spike shapes. The waveforms received were used to create the principal components so that they could be used to represent the input waveforms in a low-dimensional space. This low- dimensional representation was done such that there was maximum differentiability between each component, the average distance between the clusters for each waveform had to be the highest possible distance. The system was able to identify spike shapes that displayed the most significant variation, facilitating real-time processing and clustering of neural data even in noisy environments.

The model [30] was able to distinguish low-amplitude spikes from noise and facilitated online monitoring, identification, and classification of multiple neurons simultaneously. However, it struggled with accurately detecting spike times for low-amplitude

neurons, as these spikes were easily masked by noise in a dense spectral environment. Moreover, the system faced challenges with overlapping spikes, which complicated identifying individual spike times and mapping them to the correct neurons in high-frequency settings due to the complex waveforms created by overlapping spikes.

E. M. Glaser et al. [31] enhanced a previous system by utilizing a sophisticated PCA approach, which estimated optimal basis functions to maximize average cluster distance and reduce waveform overlap. This improved system effectively handled noise by handling EEG (Electroencephalogram) signals which complicate spike detection by mimicking spike characteristics. PCA transformed the signal into a set of uncorrelated and orthogonal principal components. By transforming the signal into these principal components, arranged by maximum variance, the system could distinguish components dominated by EEG noise. This capability allowed for the separation of actual spikes from the EEG noise, a significant improvement over the Abeles et al. [30] model, which struggled with this issue.

Additionally, it also optimized the real-time processing by allowing for immediate feedback and correction, making it more optimal for real-time data analysis compared to the more computationally intensive methods used in Abeles et al. [30] project. However, one significant challenge was the initial selection of the basis function for the identification of PCA. If the basis function implemented was not suitable to accurately encapsulate the features of the neuronal spikes then the system would not be able to efficiently filter out the noise. Realizing the appropriate basis function would require a tedious trial-and-error approach, something that is very time-consuming and computationally expensive.

3.4 Independent Component Analysis (ICA)

The algorithm implemented by Bell et al. [32] used unsupervised neural networks supported by blind separation for higher-order statistics (HOS) and separation of signals. The model aimed to satisfy the Infomax or Maximum Information Preservation Principle [28], by maximizing mutual information, a measure of knowledge transfer between the inputs and outputs of a neural network (formula 3.1), enabling the network to separate independent sources (signals that are independent of one another) from mixed signals This was done to allow the network to solve blind separation tasks, which involve separating overlapping signals into their original independent sources without any prior knowledge of the overlap, by reducing redundancy across the network layers.

Bell et al. [32] utilized Independent Component Analysis (ICA) to achieve a high Mutual Information (MI) score, a metric indicating the shared information between input and output variables. In their model, the input variables were the mixed signals received by the neural network, while the output variables were the signals separated by the network. The MI score measures how much knowing one variable reduces uncertainty about the other. The implementation of ICA improved the process by training network layers to increase the entropy of the output signal $H(Y)$. This training, achieved through stochastic gradient ascent on the parameters of a sigmoidal function in high-density areas of the input layer, reduced redundancy between the output layer units. This reduction in MI between outputs effectively addressed the issue of blind signal separation, facilitating the distinct separation of signals in multichannel recordings.

$$I(Y;X) = H(Y) - H(Y|X) \tag{3.1}$$

Here, $I(Y;X)$ is the mutual information between the input X and the output Y. $H(Y)$ is the entropy of the output, and $H(Y|X)$ is the conditional entropy of the output given the input. In a scenario where $H(Y|X)$ is zero (i.e., there is no noise), $H(Y)$ increases, reducing the redundancy among the output signals and thereby facilitating the separation of independent sources in multichannel recordings.

ICA provided an effective solution for the separation of overlapping signals, a limitation in Bayesian clustering as previously mentioned in Section 3.2. ICA addressed the issue by assuming that all the signals are independent and have non-Gaussian distribution. This practice as detailed by Takahashi et al. [33], where the decomposition of overlapping signals into statistically independent components effectively isolates individual neuron activities. This feature is especially useful for non-stationary signals.

The model proposed by Bell et al. [32] has some limitations, notably in how it assumes that sources are combined in a straightforward, linear way. However, the model is adaptable in situations where the number of measured signals (or recordings) doesn't match up exactly with the number of actual sources (like different sounds or electrical signals from the brain). The model is also not equipped to handle adaptive time delay, which refers to the latency due to the delay in the signal reception and propagation.

For an additional methodological approach utilizing Wavelet-based feature extraction, please see the Appendix A.2, which provides further insight into this technique.

Chapter 4

Methodology

This chapter outlines the methodologies used for feature extraction, clustering, and validation in the dissertation, including dimensionality reduction techniques like PCA, ICA, UMAP, and Isomap, and various clustering algorithms. It explains the selection and mathematical formulations of these methods and describes the experimental setup for validation and benchmarking, assessing the accuracy and reliability of the results. The project's workflow is illustrated in Figure 4.1.

4.1 Software Dependencies and Toolkits

The neural data analysis and spike sorting pipeline employed the SpikeInterface toolkit (Version: 0.101.0), developed by Alessio Buccino and Samuel Garcia [7]. This Python toolkit is specifically designed for seamless integration into neural data workflows, facilitating rapid data extraction, and quality control. Essential functions were managed using its modules: `spikeinterface.extractors` for data extraction, `spikeinterface.preprocessing` for preprocessing, and `spikeinterface.qualitymetrics` for evaluating quality metrics.

Electrode probe configurations were handled through the `ProbeInterface` package, ensuring precise control over probe layout and geometry. Visualizations employed `matplotlib` for 2D graphics and `mpl_toolkits.mplot3d` for 3D visualizations, offering clear representations of complex neural data. Data operations utilized `NumPy` for numerical tasks and `Pandas` for structured data management. Further analysis leveraged dimensionality reduction techniques like `PCA` and `FastICA`, and clustering methods such as `MeanShift` from the `scikitlearn` library, with hierarchical clustering facilitated by `SciPy`, enhancing the pipeline's analytical depth.

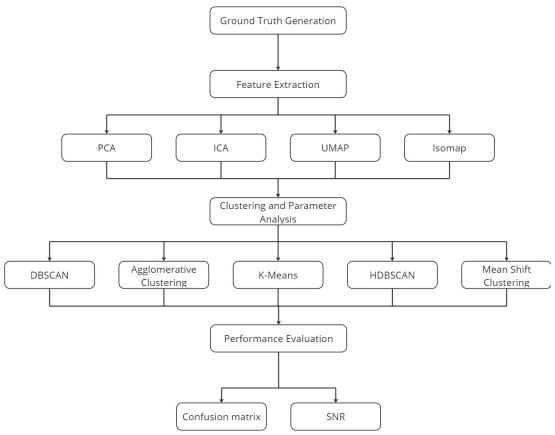

Figure 4.1: The figure illustrates the methodologies implemented in the dissertation, including feature extraction, clustering, and validation processes.

4.2 Ground Truth Generation

To validate the spike sorting pipeline, synthetic neural data was generated using the `generate_drifting_recording` function from the `SpikeInterface` library. This simulation replicated a neural recording from 40 neurons over 600 seconds at a `30,000 Hz` sampling rate. The probe layout featured 4 columns with 8 contacts each, arranged with pitches of 16 µm in the x-direction and 40 µm in the y-direction, this bears similarities to the design of Neuropixels probes introduced by Jun et al. [34]. Contacts were configured as `square` with 12 µm widths. Spike templates followed an ellipsoid mode, capturing 1.5 milliseconds pre-spike and 3.0 milliseconds post-spike. Waveforms varied from 150.0 µV to 500.0 µV in amplitude, with decay between 10 µm and 45 µm.

Neuron positioning within the 6 µm to 25 µm z-axis range ensured realistic spatial distribution, with the closest spacing of 12 µm. Neuron firing rates ranged from 0.1 Hz to 1.0 Hz, incorporating a 4-millisecond refractory period. Environmental noise was modeled between 5.0 µV and 10.0 µV, with spatial decay set at 25 µm.

This controlled synthetic dataset allows for rigorous testing of the spike sorting algorithms, establishing the pipeline's accuracy and reliability.

4.3 Feature Extraction

As outlined in Section 2.2.3, feature extraction is crucial for identifying essential features and reducing dimensionality within the spike sorting pipeline. In this project,

the spike detection steps were intentionally omitted and instead utilized events from the ground truth dataset. While spike detection is a typical preliminary step in spike sorting, it was omitted here because the main focus of this project was on the subsequent stages—feature extraction and clustering. This dissertation aligns with another dissertation project that specifically focuses on spike detection, allowing this work to concentrate on optimizing feature extraction and clustering methods. By collaborating and integrating findings from both dissertations, the goal is to develop a comprehensive system that effectively manages the entire spike sorting process. This structured task division allows each project to refine specific aspects of the pipeline, enhancing the system's robustness and accuracy.

A structured approach was adopted to estimate spike sparsity, extract relevant waveform features, and pinpoint spike locations. The `estimate_sparsity` function, with the `best_channels` option, was utilized to identify the most relevant channels for each unit, focusing on channels that best capture neural activity. This function is integral to ensuring attention is directed towards channels containing the most significant signals for each spike. The setup for feature extraction was configured using the `create_sorting_analyzer` function in the `SpikeInterface` library, facilitating the performance of subsequent tasks such as waveform extraction, feature calculation, and spike location analysis. The results from recording extractors and analysis results were managed within a `sortinganalyser` folder. The feature extraction sequence commenced with the `analyzer` variable, which spearheaded the extraction of waveforms capturing data 0.2 milliseconds before and 2.0 milliseconds after each spike event. This step is critical for analyzing the shape and abnormalities of individual spikes and helps in distinguishing between different units.

Further, the command `analyzer.compute("templates")` was employed to calculate the template for each unit. These templates are pivotal as they encapsulate the characteristic spike shape of neurons, serving as foundational elements for later stages of clustering and classification. Following this, the `analyzer.compute("spike_locations")` function was used to estimate the spatial positions of spikes. The sorting extractor then translated sorted spike data into a vector of spike times using `sorting_extractor.to_spike_vector()`. Once all pertinent features were assessed, they were subjected to feature extraction methods for enhanced dimensionality reduction and further analysis.

Various feature extraction methods, detailed in the following sections, were then applied to these features to evaluate each method's ability to extract the most pertinent

information from the data. The libraries imported for the usage of PCA, ICA, UMAP and Isomap are `sklearn.decomposition.PCA`, `sklearn.decomposition.FastICA`, `umap.UMAP`, and `sklearn.manifold.Isomap`, respectively. These libraries were used to implement the feature extraction methods, the upcoming sections will detail the reasoning behind applying these methods in the pipeline.

4.3.1 Principal Component Analysis

In the PCA analysis, principal components were computed with two components using the `analyzer.compute("principal_components", n_components=2, mode="by_channel_local")` command. After calculating the spike locations, a feature vector was created by stacking the x and y coordinates of the spike locations with the two principal components. The features were then scaled for consistency before being used in further analysis. The feature set consisted of Spike location (x), Spike location (y), PC1 and PC2 had a shape of (4,13909). 13,909 represents the total number of spikes (or data points) in the dataset and 4 represents the four rows corresponding to the features.

The graph in Appendix A.4 shows the reconstruction error relating to the number of principal components used in PCA. By analyzing the mean squared error between the original and reconstructed data with varying component numbers, we identified that two components are optimal. This number marks where the error significantly drops and stabilizes, suggesting minimal benefit from additional components. This consistent component number was also applied across other methods for comparability in analyses.

The paper by Rui-Qi Song et al. [14] discusses using PCA to extract spike features, aiding classification by reducing overlap in spike amplitudes. PCA captured essential waveform characteristics in a lower-dimensional space, enhancing the ability to distinguish spikes even when amplitude differences were subtle in the original high-dimensional space.

4.3.2 Independent Component Analysis

ICA (detailed in Section 3.4) process began with extracting the relevant waveform data (variable `w`). The waveform data was retrieved by accessing the pre-computed waveforms. A channel-wise selection was necessary as ICA was applied locally to each channel's waveform data. ICA was conducted using the FastICA algorithm,

configured to extract two components. This transformation was applied to the waveform data `w`, resulting in two independent components that encapsulate critical features of the waveform's structure. To improve the interpretability of these components when integrated with spike location features, they were scaled by a factor of 5. The final feature vector was assembled by combining the x and y coordinates of the spike locations with the scaled ICA components, thus forming a cohesive dataset ready for clustering analysis.

Buccino et al. [35] explore how ICA effectively isolates individual neuronal signals from different electrodes for spike sorting. This method proves crucial in extracellular recordings where multiple neurons fire simultaneously. ICA also reduces the need for manual intervention, thus eliminating biases in signal categorization, making it particularly advantageous for handling complex, noisy data in HD-MEA recordings. Furthermore, the study notes that ICA excels in environments with fewer overlapping signals, improving signal-to-noise ratio (SNR).

4.3.3 Uniform Manifold Approximation and Projection

As explained in Section 2.2.3, UMAP ensure that we are preserving the original structure of the data in high-dimensional space even when projecting it to a low-dimensional manifold. The pipeline uses the `UMAP` function from the `umap` library. Similar to the previous extraction methods, the UMAP function was applied to the waveform data. The resulting UMAP features were scaled and combined with the spike locations (x and y coordinates) to form a unified feature set for further analysis.

Zhao, Shunan, et al. [36] spike sorting model details how UMAP can project data into non-linear space, something that PCA fails to achieve. This characteristic gives UMAP the ability to identify patterns that are not visible in linear space and aid in better separation of spikes. UMAP has better reproducibility, it can ensure that the results generated will be consistent provided the environmental conditions remain the same. This implies that cluster results will not change and are not dependent on random initialization or variations within the data (though ICA and Isomap have slight reproducibility issues).

4.3.4 Isometric Mapping

Isomap is a non-linear dimensionality reduction technique that expands on MDS by retaining the geodesic distance of all spikes (Section 2.2.3). The

`sklearn.manifold.Isomap` function constructs the neighbourhood graph using the closest neighbour of each spike, then it uses geodesic distance to find the shortest path between each pair of points. Lastly, MDS is applied to the previously calculated distance to create a low-dimensional space that preserves the original manifold structure of the data. A four-set feature set was obtained by scaling and merging of Isomap components with the spike location to form a consolidated feature set.

Adamos, Dimitrios A., et al. [16] discusses how preserving the structure of the original data is a very important feature for HD-MEA data as the spikes tend to be muti-dimensional. Reducing dimensionality and being able to maintain the geometric structure helps the algorithm identify patterns that would otherwise not be evident in its original space. Furthermore, Isomap helps to form well-clustered shapes that are robust and can accurately represent the ground truth. Lastly, the paper also mentions that Isomap is well-equipped for handling non-linear data because of its ability to preserve such relationships.

Despite the widespread use of PCA in numerous projects for spike sorting, the goal behind exploring the less popular extraction methods was to evaluate whether combining them with different clustering techniques could generate higher accuracy. And if so, to hypothesize the reason behind such an occurrence.

4.4 Clustering

After feature extraction, the resulting dataset is fed into the clustering pipeline, undergoing multiple stages of analysis. The process starts with the extraction and analysis of spike waveform features. It then progresses to applying dimensionality reduction techniques as detailed in Section 4.3. Following this, the optimal parameter settings that yielded the highest model performance were selected. The core of the pipeline focuses on systematically tuning parameters for each clustering method used.

A specific function, `analyze_clustering`, was developed to evaluate the performance of each method's parameter configurations. Key responsibilities of this function include filtering out noise and outliers, such as -1 labels commonly used in DBSCAN and HDBSCAN, and comparing the predicted clusters against ground truth labels. Performance analysis is carried out using a `NumpySorting` object from the `SpikeInterface` toolkit, enabling detailed evaluations through confusion matrices and accuracy metrics. Beyond accuracy, precision and recall metrics are computed to assess the trade-offs inherent in each clustering approach comprehensively. The optimal

parameters resulting in the highest accuracy are then selected for further evaluation in the Performance Evaluation function (explained in Section 4.5).

4.4.1 Density-Based Spatial Clustering of Applications with Noise

DBSCAN is used to cluster spikes based on the density of points. The algorithm can mark noisy data separately, as they usually lie in low-density areas. Points that are within a set radius ε are merged to form clusters, and a cluster is formed when the threshold for the minimum number of points within ε is satisfied. The `sklearn.cluster` library is imported to use the `DBSCAN` algorithm. As mentioned earlier, different parameter values were selected for experimental purposes.

The parameters for DBSCAN included the ε range and minimum samples. Karami et al. [37] examined methods for setting these parameters and emphasized the impact of varying ε and minimum samples on clustering results. ε values ranged from 0.1 to 1.5, a choice justified by literature, underscoring the algorithm's sensitivity to this parameter which can significantly affect cluster formation. The minimum number of samples(3, 5, 10, and 15) enabled the formation of clusters with varying point densities, balancing the detection of smaller clusters and the exclusion of noise, as supported by previous studies [38, 39, 40]. These parameter configurations were chosen to optimize clustering effectiveness and minimize noisy clusters.

DBSCAN was implemented because it efficiently separates true spikes from noise, ensuring that the clusters are not skewed by the presence of noise. Unlike other clustering methods, DBSCAN is not constrained to spherical shapes, allowing it to form non-linear cluster shapes makes it particularly suited for the complex and variable nature of spike sorting [41]. Additionally, it adapts to the data by not requiring a predefined number of clusters.

4.4.2 Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering to form clusters by recursively merging the closest cluster pair till the set number of clusters are formed. From the `sklearn.cluster` library's `AgglomerativeClustering` was imported and a range of values was defined for the number of clusters.

The parameter for the number of clusters (`n_clusters_range`) was set to [40, 50, 60]. The paper by Ardelean et al. [42] discusses how different cluster sizes can affect the quality and interpretation of clustering results. This range was chosen to balance

granularity and generalization, aiming to distinguish clusters without overfitting noise or small data variations. For the linkage criterion, 'ward,' 'complete,' 'average,' and 'single' were used. Blashfield et al. [43] paper evaluates these linkage methods and highlights their strengths in different scenarios, emphasizing that the choice of linkage can drastically impact cluster formation. The ward method, known for minimizing variance within clusters, was included due to its robustness in generating compact and spherical clusters. Meanwhile, the complete and average methods were selected for their ability to handle varying cluster shapes, and single linkage was tested to observe its impact on chaining effects.

4.4.3 K-Means Clustering

K-means clustering is an iterative method to break the dataset into `k` clusters. The `sklearn.cluster` library's `KMeans` algorithm was a recursive method where every datapoint was assigned to the closest centroid and then the centroids were estimated as the mean of all points. The algorithm continued until no points changed clusters, and the centroids remained unchanged after recalculating the mean [21]. In the clustering algorithm, different parameter values were used to understand the behaviour and performance of the algorithm.

One of the primary parameters is the number of clusters (`n_clusters_range`), which was tested in the range of 20 to 60. A similar setting was used by Pedreira et al. [44]. Another important parameter is the number of initializations (`n_init_range`), which controls how many times the algorithm is executed with varying initial centroid seeds. This is crucial for avoiding suboptimal clustering solutions that can arise due to the K-means algorithm's sensitivity to initial conditions. Studies like Arthur et al. [45] have shown that using 10, 20, or 30 initializations can improve clustering stability by ensuring that the best possible solution is found across multiple runs. Lastly the maximum number of iterations (`max_iter_range`) was defined, commonly used values, such as 300, 500, and 700 (like those from Arthur et al. [45]). While 300 iterations are often sufficient for typical convergence, extending this range to 500 and 700 allowed for the examination of more complex cases where additional iterations might yield more accurate clusters.

By adjusting these parameters, the K-means implementation was fine-tuned to handle a diverse range of scenarios, ensuring robust clustering results. E Chah el al. [46] study mentions that despite the simplistic nature of the algorithm, the ability

to reassign cluster centres makes it optional for the spike pipeline as it groups spike showing similar tendencies into one group. Additionally, when K-means is paired with a feature extraction method it is robust and adaptable in handling sophisticated neural data.

4.4.4 Hierarchical Density-Based Spatial Clustering of Applications with Noise

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is an advancement of DBSCAN, it introduces a hierarchical approach to density-based clustering. Unlike DBSCAN, it works on the global density parameter. The HDBSCAN implemented used a range of density thresholds.

Yuan et al. [22] publication examines the implication of selecting a flexible range for the minimum cluster size, as this parameter directly affects the algorithm's ability to detect small but meaningful clusters without treating them as noise. The study emphasizes the need to fine-tune the `min_cluster_size` parameter based on the data's structure to achieve optimal clustering stability. Similarly, the `min_samples` plays an important role in defining the core points. By experimenting with a range of 5 to 20, the goal is to balance the detection of small, tightly-packed clusters against the risk of treating larger, more distributed clusters as noise. Wang et al. [47] highlight that this parameter is key to adapting HDBSCAN to diverse clustering scenarios without needing extensive pre-tuning. Although less critical in HDBSCAN than in traditional DBSCAN, the epsilon parameter `epsilon_range` was also analysed. The range from 0.05 to 0.3 was selected, ensuring that clusters are detected even when density differences are subtle, which is crucial in complex datasets.

HDBSCAN was selected due to its ability to handle variable data densities and superior noise tolerance, making it ideal for analyzing extracellular spike recordings. Unlike rigid methods like K-means or DBSCAN, HDBSCAN's hierarchical structure enables cluster detection at multiple resolutions, accurately identifying neuron groups of different sizes [47]. Its proven effectiveness in challenging tasks like complex image segmentation further supports its suitability for the dissertation's objectives.

4.4.5 Mean Shift Clustering

Mean Shift Clustering is a clustering technique that does not require prior knowledge of the number of clusters. The algorithm repeatedly shifts each data point towards

the mode (the area of highest density) of the data points in its neighbourhood until convergence, forming clusters where data points converge to the same mode [48].

For the dissertation two parameter values (`bandwidth_range` and `min_bin_freq_range`) were defined. The bandwidth parameter determines the specific area in which data points are considered while calculating the mode or peak of density. For each feature set, an initial bandwidth estimate was calculated using a quantile-based method. The ranges were then refined based on successful trial results. The refined bandwidth ranges included 28 to 32 for ICA, 4.8 to 5.2 for PCA, 35 to 40 for UMAP, and 450 to 500 for Isomap. Georgescu et al. [49] emphasized that tuning the bandwidth parameter is critical to achieving the right balance between noise filtering and preserving cluster detail in high-dimensional spaces. Another essential parameter is the minimum bin frequency, which controls the threshold for how many points must be in a bin (or neighbourhood) for it to be considered a valid cluster. The range [1, 3, 5, 10] was selected. Lower values allow the detection of smaller clusters that may be more prone to noise, while higher values filter out these smaller, potentially spurious clusters.

Mean Shift Clustering is effective in spike sorting due to its adaptability and non-parametric nature, allowing it to handle data without predefined cluster assumptions—a key advantage in neural data analysis where neuron cluster counts are often unknown. This algorithm is skilled at uncovering the inherent structure in data, essential for distinguishing between spikes from different neurons. It also effectively manages the complex, non-linear shapes typical of spike clusters, which defy simple geometric categorization. By adjusting to data density, Mean Shift provides robust spike detection and clustering, even in noisy environments, making it highly suitable for such applications [48, 23].

4.5 Performance Evaluation

To test the performance of the pipeline, the comparison of the clustering results with the ground truth dataset was conducted. Initially, the code filters noise and outliers (labelled as -1) from spike time data, retaining only valid spikes for analysis. These filtered spikes are then encapsulated in a `NumpySorting` object, facilitating ground truth comparison via the `SpikeInterface` library's `GroundTruthComparison` tool. This comparison provides summaries and key performance metrics, such as accuracy (see Equation 4.1), demonstrating how well the predicted spike labels match the actual spike labels.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.1}$$

Where (TP) stands for true positives (correctly identified spikes), (TN) for true negatives (correct rejections of non-spikes), (FP) for false positives (non-spikes incorrectly labelled as spikes), and (FN) for false negatives (spikes missed in the detection).

$$\text{Precision} = \frac{TP}{TP+FP} \tag{4.2}$$

Here Precision measures the accuracy of positive predictions.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4.3}$$

Lastly, the Recall measures the ability to identify all relevant instances (true positive rate).

Visualization of results includes a confusion matrix displayed by the `ConfusionMatrixWidget`, providing a clear view of how predicted clusters correspond to actual spike labels. Additionally, a scatter plot showcases the relationship between SNR and sorting accuracy. Plotted with SNR on the x-axis and accuracy (Equation 4.1) on the y-axis, this visualization indicates how higher SNR, indicative of clearer signals, enhances sorting performance, aligning with insights from Yang et al. [24].

Following this, refined analysis using precision (Equation 4.2) and recall (Equation 4.3) metrics offers a comprehensive evaluation of clustering results. Precision measures the proportion of correctly identified spikes within the detected units, while recall assesses the completeness of detected true spikes against the total in ground truth, aiding in understanding the trade-offs between false positives and false negatives.

Moreover, a comparative graph examines the accuracy across different feature extraction and clustering techniques, exploring a wide range of parameter configurations. This methodical testing aligns with ensemble learning principles, as described by Polikar et al. [50], to pinpoint optimal combinations that maximize feature representation and cluster delineation, thereby enhancing prediction reliability and performance. These assessments are supported by detailed visualizations and directly guide the selection of the most effective clustering approaches for neural spike sorting.

Chapter 5

Results

In this chapter, the efficacy of various clustering algorithms and feature extraction methods for spike sorting is evaluated to identify the combination that maximizes clustering quality and accuracy. The analysis begins with an examination of spatial clustering results to assess the strengths and challenges of each method, especially in handling overlapping clusters. It then explores the critical role of parameter tuning and its influence on performance. Comparative graphs and tables are used to review key metrics such as accuracy, precision, and recall across different method combinations. These analyses provide detailed insights into the clustering landscape, guiding the selection of optimal clustering methods.

5.1 Challenges with Spike Data

Figure 5.1 depicts the results of clustering spatial locations using Agglomerative Clustering, with features extracted using Isomap. Each point in the plot represents a spike, while the different colours represent distinct clusters. The x and y coordinates represent the spatial location, offering insight into the distribution of spikes across this region. The centroids of each cluster are marked with black crosses, highlighting the central tendency of each grouping. Upon comparing the clustering results using Isomap (Figure 5.1) with the clustering of raw locations (Figure 5.2), it is evident that different dimensionality reduction techniques and clustering parameters can influence the overall clustering performance and separation of spike data (something that isn't achieved in Figure 5.2).

Upon close inspection of Figure 5.1, it is evident that there exist distinct clusters that show clear separation in some regions. These clusters demonstrate high densities,

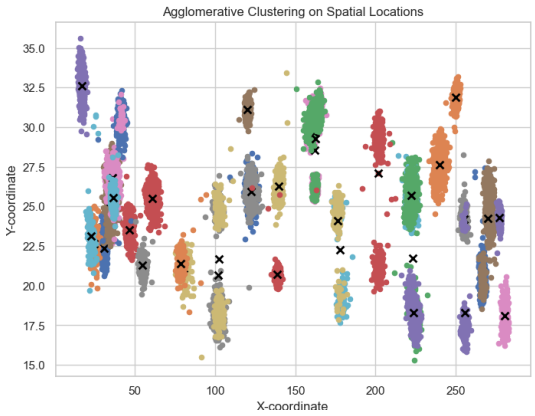

Figure 5.1: Agglomerative Clustering on Spatial Locations using Isomap with accuracy of 0.10. The clusters are represented the visible overlap, highlighting the challenges in differentiating closely situated spikes from nearby neurons.

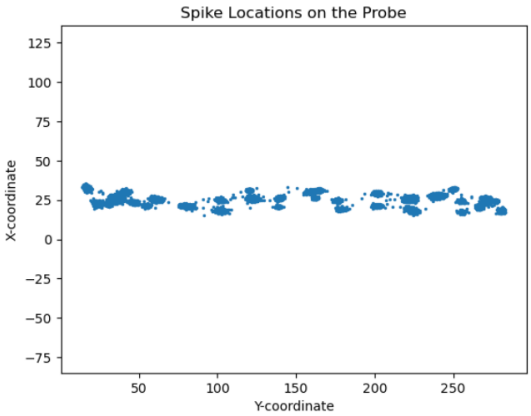

Figure 5.2: Clustering of Raw Locations. The spatial distribution of detected spikes across the probe is visualized with clusters, focusing on distinguishing nearby neurons based purely on their coordinates.

indicating that spikes originating from these areas are consistent and well-defined, making it easier to identify them as originating from a specific neuron or neuron group. However, in other areas, there is a noticeable overlap between clusters. This overlap is particularly evident in regions where spikes appear near one another. This overlapping of clusters is a well-known challenge in spike sorting, particularly when spikes originate from close neurons. When neurons are spatially adjacent, it would seem as though the action potentials are originating from the same neuron, leading to clustering ambiguity. The overlapping points demonstrate how spikes from different neurons can be difficult to differentiate due to their spatial closeness. This pattern is common in extracellular recordings, where spikes from neurons in close proximity can be misclassified. As a result, even with a robust hierarchical clustering technique like Agglomerative clustering, the resulting clusters can be blurred.

This overlap highlights the need for incorporating additional pre-processing techniques. While spatial location provides valuable information, it alone is insufficient to resolve the complexity of spike sorting, particularly when distinguishing between spikes from closely located neurons. Implementing advanced pre-processing methods such as parameter tuning becomes of utmost importance. With systematic modifications of clustering parameters like bandwidth, minimum cluster size, and other algorithm-specific settings, the performance can be significantly enhanced, leading to clearer distinctions

between clusters and improved overall accuracy in spike sorting.

5.2 Parameter Tuning

To reduce the extent of overlapping tendencies in spikes and improve the clustering performance, a recursive approach was developed that iterated through a range of parameter values (as explained in Section 4.4). The goal was to identify the near-perfect combination of parameters that could result in the highest clustering performance for each extraction and clustering method combination.

The effectiveness of this approach lies in its ability to fine-tune the algorithm's settings, ensuring that the clusters are as distinct and well-separated as possible. Typically, default settings can result in suboptimal clusters where overlap and ambiguity are more dominant. This strategy was vital as it could handle the variability in spike data, where even small changes in parameter values can significantly impact the clustering outcome.

The graph 5.3 presented illustrates the accuracy of HDBSCAN clustering as the `min_samples` parameter is scaled, specifically using PCA features. The x-axis represents the different values of `min_samples`, ranging from 5 to 20, while the y-axis shows the corresponding accuracy values.

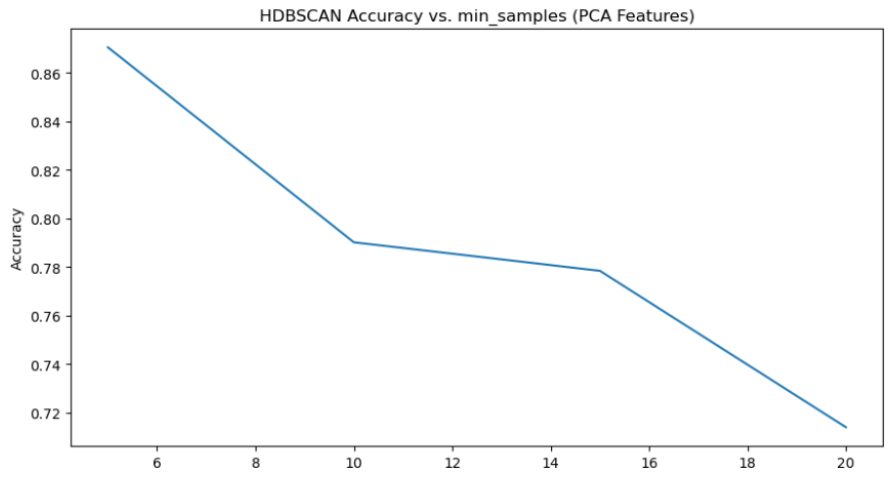

Figure 5.3: Accuracy vs. Min Samples for HDBSCAN with PCA Features. The plot shows how accuracy varies with changes in the `min_samples` parameter. Initially, accuracy is high at lower values but gradually declines as `min_samples` increases.

Initially, the graph 5.3 shows high accuracy at around 0.89 when `min_samples` is set to 5. However, as the parameter value increases to 10, a noticeable drop occurs, bringing the accuracy down to approximately 0.79. Beyond this point, the accuracy

remains relatively stagnant up until the `min_samples` value reaches 15, where it ranges around 0.78. Finally, as the value reaches 20, a significant decline is observed, with the accuracy falling to around 0.72. This pattern highlights the sensitivity of HDBSCAN to this particular parameter, where smaller values result in better cluster formation, while larger values deteriorate performance.

This analysis was systematically performed across all combinations of feature extraction methods and clustering techniques. Interestingly, different combinations exhibited varying trends in their accuracy versus parameter plots. For instance, while some showed a clear peak followed by a decline similar to this graph (Agglomerative clustering accuracy with linkage for ICA), others displayed erratic behaviour with no observable trend (K-Means with maximum iterations for ICA). The variety of patterns observed emphasizes the importance of parameter tuning specific to each feature extraction and clustering combination. The complete set of these accuracy-to-parameter graphs and discussions can be found in Appendix B, providing a broader context of the clustering behaviour across different settings.

5.3 Performance Evaluation

Given the multiple combinations of clustering algorithms and feature extraction methods, each with a unique set of parameter configurations, it was essential to conduct a thorough evaluation to determine which combination yielded the highest clustering accuracy. As a result, a comprehensive accuracy graph was generated, which serves as a visual summary of the performance of each combination.

By experimenting with a wide range of parameter settings, including variations in the number of clusters, linkage methods, minimum samples, and other critical factors, the analysis aims to uncover which methods consistently perform well and which struggle to effectively handle the complexity and noise present in the data.

The graph 5.4 presents a comparative analysis of the clustering accuracy achieved by various feature extraction and clustering method combinations. Each line represents a specific feature extraction method, while the x-axis corresponds to different clustering techniques and the y-axis represents the accuracy value. The top two best-performing combinations (highlighted with green circles) and the worst two (highlighted with red circles) are marked for clarity. From the graph 5.4, it is evident that HDBSCAN with PCA features delivers the highest accuracy at around 0.87. This is closely followed by K-means with ICA features, which achieves approximately 0.86 accuracy. In contrast,

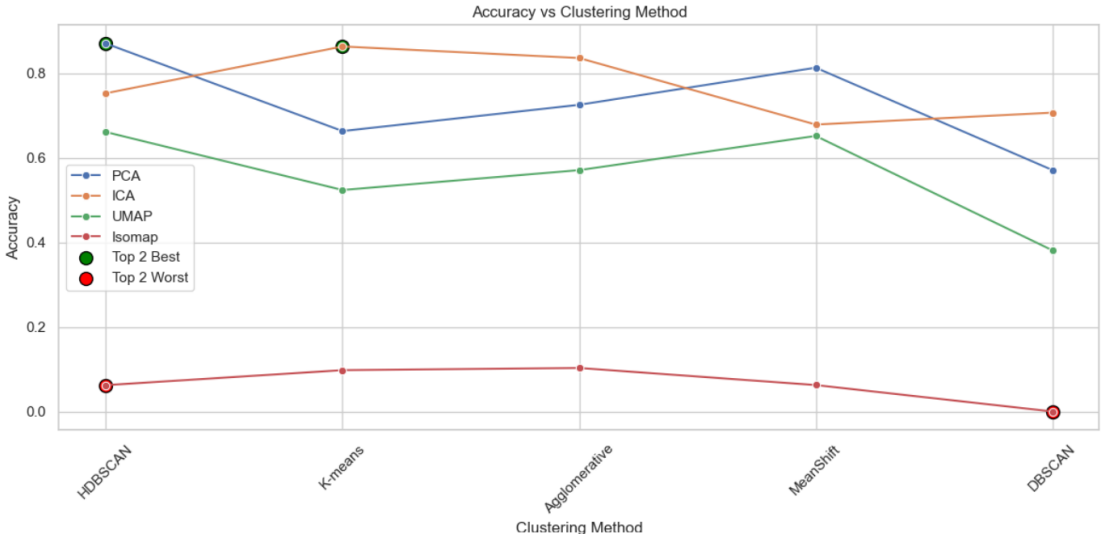

Figure 5.4: Accuracy Comparison Across Clustering Methods and Feature Extraction Techniques. The graph shows PCA with HDBSCAN clustering has the highest accuracy, while Isomap combinations perform with low accuracy

DBSCAN with Isomap features performs poorly, with accuracy dropping close to 0.0, meaning all clusters were identified as noise. Similarly, MeanShift with Isomap features shows low performance as well, emphasizing the challenge in using Isomap for this type of spike data. On the other hand, PCA and ICA, when combined with K-means and HDBSCAN, consistently show strong performance across the board. For example, MeanShift with PCA achieves an accuracy of about 0.80, indicating that both MeanShift and PCA are well-suited for capturing the structure of this dataset. The combination of Isomap with Agglomerative clustering shows a sharp decline in accuracy to around 0.1, which highlights Agglomerative's limitations in this context. This performance drop suggests that Agglomerative clustering is unable to effectively group data when the feature space is derived from Isomap.

While accuracy is a commonly used metric, it is not always reliable, especially in imbalanced datasets where certain clusters might dominate the classification results. Spike sorting often exhibits such skewness, where the distribution of neuron locations or spike counts is uneven, potentially leading to inflated accuracy figures that do not reflect true performance. In the context of spike location, clusters representing neurons with more frequent spikes might overpower others, leading to misleadingly high accuracy.

To address this limitation, additional metrics like precision and recall were evaluated. The table 5.1 provides detailed precision and recall values for each combination. Notably, while K-means with ICA features shows high accuracy (0.86), it also maintains

a solid balance with precision (0.75) and recall (0.75), making it a reliable choice. In contrast, K-means with UMAP features, despite having an accuracy of 0.52 have a precision and recall of 0.43, indicating that despite the decent accuracy, the precision and recall metrics suggest less reliable detection of relevant spikes.

Method and Feature Set	**Precision**	**Recall**
DBSCAN PCA	0.102	0.500
DBSCAN ICA	0.706	0.600
DBSCAN UMAP	0.031	0.275
DBSCAN Isomap	0.000	0.000
Agglomerative PCA	0.480	0.600
Agglomerative ICA	0.750	0.750
Agglomerative UMAP	0.475	0.475
Agglomerative Isomap	0.050	0.050
K-means PCA	0.400	0.600
K-means ICA	0.750	0.750
K-means UMAP	0.425	0.425
K-means Isomap	0.050	0.025
HDBSCAN PCA	0.919	0.850
HDBSCAN ICA	0.758	0.625
HDBSCAN UMAP	0.742	0.575
HDBSCAN Isomap	0.077	0.025
MeanShift PCA	0.784	0.725
MeanShift ICA	0.289	0.600
MeanShift UMAP	0.537	0.550
MeanShift Isomap	0.000	0.000

Table 5.1: Precision and Recall for Various Clustering and Feature Extraction Combinations

These metrics emphasise the importance of considering a broader range of performance indicators rather than relying solely on accuracy. Methods like HDBSCAN with PCA, which show both high precision (0.92) and recall (0.85), are more likely to provide consistent clustering results, even in noisy datasets.

Chapter 6

Discussions

As evident in iFgure 5.4, HDBSCAN with PCA in parameter setting `min_cluster_size`=5, `min_samples`=5, `epsilon`=0.05 shows high accuracy(0.87), recall(0.85), and precision(0.92). This high performance can be potentially attributed to the specific characteristics of HDBSCAN and PCA, which are well-suited for handling complex, high-dimensional datasets where cluster density and structure vary (explained in Section 4.4.4). PCA, by reducing dimensionality while preserving essential features and mitigating noise, has been shown in other studies, such as that by Ye et al. [22], to enhance clustering results. This aligns with the consistent high accuracy observed across various clustering methods using PCA, as documented in Abeles et al.'s project [30] (detailed in Section 3.3).

A hypothesis for the observed performance could be related to the settings for `min_cluster_size` and `min_samples`, which might have enhanced the algorithm's ability to identify clean-cut clusters even for sparsely distributed spikes. These parameters aid in distinguishing true clusters from noise, crucial in cases with overlapping neural spikes and tightly packed groups needing precise separation [14]. Furthermore, the `epsilon` value may help refine cluster boundaries, ensuring a balance between noise and cluster points [22, 47] (elaborated in Section 4.4.4). The strong correlation between these parameter settings and the high performance observed suggests that HDBSCAN's adaptability to varying densities is well-suited for spike sorting, as supported by related signal processing studies [14]. HDBSCAN's ability to automatically adapt to changing densities coincides with the need to handle the non-uniform, high-dimensional distributions typical in spike sorting. This flexibility, combined with the dimensionality reduction afforded by PCA, likely contributed to the well-defined clusters noted in the results.

Apart from this K-means with ICA in parameter settings (`n_clusters`=40, `n_init`=20, `max_iter`=300) also generated high accuracy recall and precision values. The underlying hypothesis here is that ICA's assumption that neuron spiking activities are independent processes (discussed in Section 3.4) is particularly beneficial for HD-MEA recordings with overlapping signals. By treating these overlapping signals as independent, ICA effectively separates individual neuron contributions, enhancing accuracy in spike sorting. These findings align with the findings by Buccino et al. [35] (detailed in Section 4.3.2) that illustrated how ICA improves the SNR and increases the sensitivity of identifying individual spikes.

But if that were the case, why is low accuracy observed for MeanShift and ICA? This could perhaps be due to the parameter settings and the algorithm's suitability. One possible explanation is that Mean Shift clustering which targets high-density regions within data, contrasts with K-means' ability to adapt to a predefined number of clusters, here 40. Because MeanShift is density-based and produces variable cluster counts depending on the data distribution, it may struggle with the high-dimensional, non-uniform densities created by ICA's feature extraction. As noted by Toosi et al. [23], Mean Shift clustering can struggle with non-uniform densities and skewed data distributions, particularly in spike sorting tasks where clusters are not symmetrically distributed. This hypothesis aligns with our situation, where ICA's features do not cluster effectively under Mean Shift, resulting in lower accuracy and indistinct clusters. Chah et al. [46] stated that K-means when paired with dimension reduction techniques like ICA, enhances spike separation due to its iterative updates and a fixed number of clusters, producing stable and consistent results. This method efficiently handles data variability, as K-means maintains consistent cluster boundaries through its iterative process and fixed parameters, thus potentially improving the segregation of spike signals.

6.1 Computation Complexities

This brings us back to the research questions stated: How can spike data be efficiently analyzed to reveal distinct patterns and what methods are potentially reliable for categorizing neural activity based on spike data analysis? The results presented earlier provide a comprehensive answer to these questions by identifying the optimal combinations of feature extraction and clustering methods that maximize accuracy in spike sorting. The findings indicate that combinations such as HDBSCAN with PCA and K-means with ICA consistently yield high accuracy, precision, and recall, highlighting their reliability

in handling complex spike data. However, it's also important to consider additional factors when selecting the most optimal methods to uncover key patterns in spike data.

To do so the pipeline can be expanded beyond the identification of the best combination, additional insights could further aid in the selection of the most appropriate method. This could be done by refining the decision process and wondering about the computational complexities as well. By evaluating the time and space complexities associated with each algorithm, one can determine the most computationally efficient approach without sacrificing clustering quality.

From Table 6.1, we can analyze how the parameter settings correlate to performance and computational complexities for different methods. The complexity variables n (number of data points), k (number of clusters), t (iterations), and d (dimensions), directly influence how the algorithms scale as dataset size and complexity increase.

DBSCAN's complexity ranges from $O(n \cdot \log n)$ in the best case (for efficient spatial indexing) to $O(n^2)$ in the worst case (for poorly defined clusters) [19]. The pipeline generates a moderately good performance for DBSCAN with PCA having `eps`=1.5 and `min_samples`=3. Although this setting doesn't necessarily give the highest performance, it balances out the computational efficiency. Contrasting, DBSCAN with Isomap with an accuracy of 0 (identifies all the clusters as noise) highlights the worst-case scenario. The parameter settings of `eps`=0.1 and `min_samples`=3 results in high computational costs without any benefit. Poor performance in this case shows that bad parameter choices can greatly increase complexity without yielding meaningful outcomes

Agglomerative clustering's worst-case complexity is $O(n^3)$, making it computationally demanding, especially for large datasets [51]. The best parameter setting for Agglomerative Clustering with PCA involves `n_clusters`=50 and `linkage`= `ward`, giving an accuracy of 0.73. The significant computational effort required by this algorithm, due to its cubic complexity in calculating pairwise distances and merging clusters, underscores a clear trade-off. Although the method is effective, its heavy computational demand reduces scalability when applied to larger datasets or a greater number of clusters.

K-means algorithm operates with a complexity of $O(n \cdot k \cdot t \cdot d)$. here the best-case scenario occurs when convergence is achieved with few iterations t [52]. For instance, best parameter settings for K-means with ICA include `n_clusters`=40, `n_init`=20 and `max_iter`=30 yielding an accuracy of 0.86. Although this setup yields high accuracy, it comes at a significant computational cost due to the large number of clusters and iterations. The trade-off lies in achieving better performance for higher computation

resources, thus making K-means computationally expensive for large, high-dimensional datasets.

HDBSCAN, with a complexity of $O(n \cdot \log n)$, shows an optimal trade-off between performance and computation [53]. The combination of HDBSCAN with PCA achieves the highest accuracy (0.870442) using `min_cluster_size`=5, `min_samples`=5, and `epsilon`=0.05. This setup shows that HDBSCAN can provide superior performance with moderate computational demands, particularly when parameters are well-tuned to the data structure. The minimal trade-off makes this configuration highly efficient in the analysis.

MeanShift, with a time complexity of $O(n^2 \cdot t)$ [54], becomes computationally expensive as the bandwidth parameter increases and more iterations are required. For example, with ICA, `bandwitdh`=4.8 and `min_bin_freq`=1, achieves an accuracy of 0.68. The computational time is high due to the quadratic scaling of pairwise distance calculations. Contrary to this, MeanShift with Isomap with `bandwitdh`=32 and `min_bin_freq`=10 performs poorly (accuracy=0.063) despite high computational effort. This suggests that increased computation does not necessarily lead to improved performance unless the parameters are optimally configured to match the characteristics of the dataset.

Clustering Algorithm	**Computational Complexity**
DBSCAN	$O(n \cdot \log n)$ (best case) to $O(n^2)$ (worst case)
Agglomerative Clustering	$O(n^3)$
K-means	$O(n \cdot k \cdot t \cdot d)$
HDBSCAN	$O(n \cdot \log n)$
MeanShift	$O(n^2 \cdot t)$

Table 6.1: Computational Complexities of Clustering Algorithms

Bring us back to the research question, to efficiently analyze spike data and categorize neural activity, it's crucial to balance the effectiveness and computational demands of different techniques. The results show that while complex algorithms like HDBSCAN with PCA strike a good balance between performance and efficiency, methods such as K-means with ICA and Agglomerative Clustering consume more resources without necessarily offering better accuracy. Conversely, some computationally intensive approaches like MeanShift with Isomap may fail to yield improved results if their parameters don't align well with the data structure. Ultimately, the choice of method and parameter settings lies in balancing the computational speed against clustering accuracy. This decision will determine the method's effectiveness in revealing distinct

neural patterns and its feasibility within practical limitations.

6.2 Limitations and Future Scope

A primary challenge in this dissertation was identifying the optimal combination of feature extraction and clustering methods, as the best choice largely depends on specific developer needs and priorities. For example, methods that offer quicker responses might be preferred when speed is prioritized over precision. Conversely, if accuracy is paramount, even at the expense of computational efficiency, different methods would be more suitable. The contextual nature of these goals complicates the identification of a universally best combination.

Another limitation lies in the use of simulated data for pipeline creation. While simulated data provides a controlled environment where performance can be mapped against known ground truth, it may not perfectly translate to actual data captured in real-world scenarios. There is uncertainty regarding how well the pipeline will perform when dealing with actual HD-MEA recordings, as the results generated might indicate different patterns or behaviours. However, the use of simulated data was necessary, as it is the only way to effectively benchmark the system's performance with a known reference(ground truth), something that is not possible with real-world HD-MEA data.

Additionally, the pipeline developed in this study primarily considers spatial locations as features. Although spatial information is significant, other features could have been explored. Due to time constraints, the focus was narrowed, leaving other feature dimensions underexplored.

The system can be expanded in the future to include diverse features, such as spike shape, temporal patterns and other waveform characteristics. Moreover, incorporating spectral clustering is another avenue worth exploring (refer Appendix A.3 for more details). Although spectral clustering is computationally intensive, it could offer better clustering performance, in cases with non-convex clusters [55]. Due to its high computational demand, it was not implemented in this study, but it holds promise for future iterations of this pipeline. Furthermore, integrating advanced algorithms like Kilosort could significantly improve spike sorting accuracy. Kilosort is designed to handle large-scale electrophysiological datasets, using template matching to refine spike detection and clustering [56]. Kilosort can be integrated by incorporating its template matching output as an initial clustering step within the pipeline, refining the spike sorting process for higher accuracy (a similar template matching approach has been

described in Section 3.1). Incorporating Kilosort could enhance the system's adaptation to complex, real-world datasets, improving spike sorting accuracy and reliability.

Chapter 7

Conclusion

This dissertation has extensively analyzed various spike sorting techniques and feature extraction methods for enhancing the processing and usability of neural data from high-density multi-electrode arrays (HD-MEAs). This research primarily investigated the efficiency of different feature extraction methods such as principal component analysis (PCA), independent component analysis (ICA), uniform manifold approximation and projection (UMAP), and isometric mapping (Isomap). The effectiveness of multiple clustering methods, including K-means, Agglomerative Clustering, HDBSCAN, and Mean Shift, was also evaluated critically.

One of the key outcomes highlighted by this research is the superior performance achieved by combining HDBSCAN with PCA feature extraction. This combination emerged as highly effective due to its ability to preserve essential features while minimizing noise, which is crucial for accurately clustering complex and overlapping neural data. Additionally, ICA, when used in conjunction with K-means clustering, significantly improved clustering accuracy, demonstrating the importance of selecting appropriate feature extraction techniques based on the clustering method used. The complexity and overlapping nature of spikes in densely populated neural recordings presents significant challenges.

This dissertation found that techniques that addressed spatial relationships, such as HDBSCAN combined with PCA and ICA with K-means, effectively differentiated signals from closely situated neurons. Furthermore, the research underscored the computational costs linked to different spike-sorting algorithms. Techniques like HDBSCAN were noted for offering a balance between computational demands and high performance, making them preferable choices for real-time neural data analysis applications. Exploring the sensitivity of the clustering algorithms to their tuning param-

eters underscored a critical aspect of the research. It was observed that minor tweaks in settings could significantly influence the accuracy of spike sorting, signalling the importance of robust and adaptable algorithm configurations.

Moving forward, the integration of additional spike features like waveform shapes and temporal patterns could be explored to refine clustering outcomes further. Applying these methodologies to real-world data holds the promise of revealing new challenges and enhancing the practical utility of spike-sorting techniques.

In conclusion, the outcomes of this dissertation make substantial contributions to the advancement of neural data analysis, particularly in the realm of spike isolation and classification. These advancements pave the way for improved brain-machine interface technologies and deepen our understanding of neuronal networks, promising enriched studies and applications in neuroscience. Future research should aim to build upon these findings, optimizing and broadening the methodologies to adapt and respond to emerging challenges in the field, thus ensuring continual improvements in both the theory and application of neural signal processing.

Bibliography

[1] Eric R Kandel. Small systems of neurons. *Scientific American*, 241(3):66–77, 1979.

[2] Zaghloul Saad Zaghloul and Magdy Bayoumi. Implementable spike sorting techniques for vlsi wireless bci/bmi implants: A survey. In *5th International Conference on Energy Aware Computing Systems Applications*, pages 1–4, 2015.

[3] Yuri Kato, Yoshihisa Matoba, Katsumi Honda, Koji Ogawa, Kan Shimizu, Masataka Maehara, Atsushi Fujiwara, Aoi Odawara, Chigusa Yamane, Naohiko Kimizuka, et al. High-density and large-scale mea system featuring 236,880 electrodes at 11.72 μm pitch for neuronal network analysis. In *2020 IEEE Symposium on VLSI Circuits*, pages 1–2. IEEE, 2020.

[4] Baptiste Lefebvre, Pierre Yger, and Olivier Marre. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris*, 110(4, Part A):327–335, 2016. SI: GDR Multielectrode.

[5] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106–117, 2015. Advances in electrophysiological data analysis.

[6] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction: foundations and applications*, pages 1–25. Springer, 2006.

[7] Alessio P Buccino, Samuel Garcia, and Pierre Yger. Spike sorting: new trends and challenges of the era of high-density probes. *Progress in Biomedical Engineering*, 4(2):022005, 2022.

[8] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[9] Chethan M Parameshwara, Simin Li, Cornelia Fermüller, Nitin J Sanket, Matthew S Evanusa, and Yiannis Aloimonos. Spikems: Deep spiking neural network for motion segmentation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3414–3420. IEEE, 2021.

[10] David Carlson and Lawrence Carin. Continuing progress of spike sorting in the era of big data. *Current Opinion in Neurobiology*, 55:90–96, 2019. Machine Learning, Big Data, and Neuroscience.

[11] R. Quian Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. *Neural Computation*, 16(8):1661–1687, 08 2004.

[12] Sonia Todorova, Patrick Sadtler, Aaron Batista, Steven Chase, and Valérie Ventura. To sort or not to sort: the impact of spike-sorting on neural decoding performance. *Journal of neural engineering*, 11(5):056005, 2014.

[13] Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53, 1998.

[14] Rui-Qi Song and Hong-Ge Li. Overlapping spikes sorting using feature fusion. In *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 391–394, 2015.

[15] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[16] Dimitrios A Adamos, NA Laskaris, EK Kosmidis, and G Theophilidis. Spike sorting based on dominant-sets clustering. In *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010: May 27–30, 2010 Chalkidiki, Greece*, pages 5–8. Springer, 2010.

[17] Saber Graf, Benoît Buisson, and Julien Artinian. Ai analysis tool for hd-mea, 2024. Accessed: 2024-08-05.

[18] James Zhang, Thanh Nguyen, Steven Cogill, Asim Bhatti, Lingkun Luo, Samuel Yang, and Saeid Nahavandi. A review on cluster estimation methods and their

application to neural spike data. *Journal of neural engineering*, 15(3):031003, 2018.

[19] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

[20] Shweta Sharma, Neha Batra, et al. Comparative study of single linkage, complete linkage, and ward method of agglomerative clustering. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMIT-Con)*, pages 568–573. IEEE, 2019.

[21] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[22] Jason Yuan Ye, Christopher Yu, Tiffany Husman, Bryan Chen, and Aryaman Trikala. Novel strategy for applying hierarchical density-based spatial clustering of applications with noise towards spectroscopic analysis and detection of melanocytic lesions. *Melanoma Research*, 31(6):526–532, 2021.

[23] Ramin Toosi, Mohammad Ali Akhaee, and Mohammad-Reza A Dehaqani. An automatic spike sorting algorithm based on adaptive spike detection and a mixture of skew-t distributions. *Scientific reports*, 11(1):13925, 2021.

[24] David XD Yang and Abbas El Gamal. Comparative analysis of snr for image sensors with enhanced dynamic range. In *Sensors, cameras, and systems for scientific/industrial applications*, volume 3649, pages 197–211. SPIE, 1999.

[25] G. L. Gerstein and W. A. Clark. Simultaneous studies of firing patterns in several neurons. *Science*, 143(3612):1325–1327, 1964.

[26] Charles M Gray, Pedro E Maldonado, Mathew Wilson, and Bruce McNaughton. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *Journal of neuroscience methods*, 63(1-2):43–54, 1995.

[27] Kenneth D Harris, Darrell A Henze, Jozsef Csicsvari, Hajime Hirase, and Gyorgy Buzsaki. Accuracy of tetrode spike separation as determined by simulta-

neous intracellular and extracellular measurements. *Journal of neurophysiology*, 84(1):401–414, 2000.

[28] Michael S Lewicki. Bayesian modeling and classification of neural signals. *Neural computation*, 6(5):1005–1030, 1994.

[29] Michale S Fee, Partha P Mitra, and DAVID Kleinfeld. Variability of extracellular spike waveforms of cortical neurons. *Journal of neurophysiology*, 76(6):3823–3833, 1996.

[30] Moshe Abeles and Moise H Goldstein. Multispike train analysis. *Proceedings of the IEEE*, 65(5):762–773, 1977.

[31] EM Glaser and WB Marks. On-line separation of interleaved neuronal pulse sequences. In *Data acquisition and processing in biology and medicine*, pages 137–156. Elsevier, 1968.

[32] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[33] Susumu Takahashi, Yuichiro Anzai, and Yoshio Sakurai. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. *Journal of neurophysiology*, 89(4):2245–2258, 2003.

[34] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

[35] Alessio P. Buccino, Espen Hagen, Gaute T. Einevoll, Philipp D. Häfliger, and Gert Cauwenberghs. Independent component analysis for fully automated multi-electrode array spike sorting. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2627–2630, 2018.

[36] Shunan Zhao, Xiaoliang Wang, Dongqi Wang, Jin Shi, and Xingru Jia. Unsupervised spike sorting for multielectrode arrays based on spike shape features and location methods. *Biomedical Engineering Letters*, pages 1–25, 2024.

[37] Amin Karami and Ronnie Johansson. Choosing dbscan parameters automatically using differential evolution. *International Journal of Computer Applications*, 91(7):1–11, 2014.

[38] Anant Ram, Ashish Sharma, Anand S Jalal, Ankur Agrawal, and Raghuraj Singh. An enhanced density based spatial clustering of applications with noise. In *2009 IEEE International Advance Computing Conference*, pages 1475–1478. IEEE, 2009.

[39] Chen Xiaoyun, Min Yufang, Zhao Yan, and Wang Ping. Gmdbscan: Multi-density dbscan cluster based on grid. In *2008 IEEE International Conference on e-Business Engineering*, pages 780–783. IEEE, 2008.

[40] Yasser El-Sonbaty, Mohamed A Ismail, and Mohamed Farouk. An efficient density based clustering algorithm for large databases. In *16th IEEE international conference on tools with artificial intelligence*, pages 673–677. IEEE, 2004.

[41] Panagiotis C Petrantonakis and Panayiota Poirazi. A novel and simple spike sorting implementation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(4):323–333, 2016.

[42] Eugen-Richard Ardelean, Ana-Maria Ichim, Mihaela Dînşoreanu, and Raul Cristian Mureşan. Improved space breakdown method–a robust clustering technique for spike sorting. *Frontiers in Computational Neuroscience*, 17:1019637, 2023.

[43] Roger K Blashfield. Mixture model tests of cluster analysis: accuracy of four agglomerative hierarchical methods. *Psychological Bulletin*, 83(3):377, 1976.

[44] Carlos Pedreira, Juan Martinez, Matias J Ison, and Rodrigo Quian Quiroga. How many neurons can we see with current spike sorting algorithms? *Journal of neuroscience methods*, 211(1):58–65, 2012.

[45] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006.

[46] E Chah, V Hok, A Della-Chiesa, JJH Miller, SM O'Mara, and RB Reilly. Automated spike sorting algorithm based on laplacian eigenmaps and k-means clustering. *Journal of neural engineering*, 8(1):016006, 2011.

[47] Jingguo Wang, Yiquan Wang, Jiandong Wang, Yongkang Shen, and Xijun Zhao. Design of an image segmentation system based on hierarchical density-based spatial clustering of applications with noise for off-road unmanned ground vehicles. In *International Conference on Autonomous Unmanned Systems*, pages 3163–3175. Springer, 2022.

[48] Roland Diggelmann, Michele Fiscella, Andreas Hierlemann, and Felix Franke. Automatic spike sorting for high-density microelectrode arrays. *Journal of neurophysiology*, 120(6):3155–3171, 2018.

[49] Georgescu, Shimshoni, and Meer. Mean shift based clustering in high dimensions: a texture classification example. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 456–463 vol.1, 2003.

[50] Robi Polikar. Ensemble learning. *Ensemble machine learning: Methods and applications*, pages 1–34, 2012.

[51] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[52] K.A. Abdul Nazeer, S.D. Madhu Kumar, and M.P. Sebastian. Enhancing the k-means clustering algorithm by using a o(n logn) heuristic method for finding better initial centroids. In *2011 Second International Conference on Emerging Applications of Information Technology*, pages 261–264, 2011.

[53] Myeong-Hun Jeong, Yaping Cai, Clair J. Sullivan, and Shaowen Wang. Data depth based clustering analysis. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, New York, NY, USA, 2016. Association for Computing Machinery.

[54] Li Li Ji Zhou Jian Tao Xicheng Tan Fang Huang, Yinjie Chen and Guangsong Fan. Implementation of the parallel mean shift-based image segmentation algorithm on a gpu cluster. *International Journal of Digital Earth*, 12(3):328–353, 2019.

[55] E Wood, M Fellows, JR Donoghue, and MJ Black. Automatic spike sorting for neural decoding. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 4009–4012. IEEE, 2004.

[56] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *BioRxiv*, page 061481, 2016.

[57] Daniel Valencia, Jameson Thies, and Amirhossein Alimohammad. Frameworks for efficient brain-computer interfacing. *IEEE Transactions on Biomedical Circuits and Systems*, 13(6):1714–1722, 2019.

[58] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

Appendix A

Appendix

A.1 Spike Sorting and Influence on Brain-Computer Interfaces

Spike sorting plays an important role in the development of Brain-Computer Interfaces (BCIs) also known as Brain-Machine Interfaces (BMIs). Neural spike trains extracted from extracellular recordings are the fundamental units of information in the brain. As the efficiency of spike sorting improves the accuracy and reliability of BCIs increases.

As described by Daniel Valencia et al. [57] in their article, improved spike sorting methods have had a huge impact on the performance of BCIs as they increase the accuracy at which action potentials are identified and clustered. This increased precision ensures that BCI can accurately interpret the user's intentions by correctly relating the spikes with a specific command. Real-time spike sorting algorithms have empowered BCIs to process neural recordings with minimal lag, a characteristic that is very much needed for applications requiring immediate response. This could be controlling prosthetic limbs or communication between devices for individuals with disabilities.

Modern BCIs utilize high-density multi-electrode arrays (HDMEAs) capable of recording from millions of neurons concurrently. Advanced spike sorting methods can efficiently handle this massive flow of data, allowing BCIs to keep up with complicated tasks without compromising on performance. Spatial whitening and CAR in spike sorting help reduce noise and improve signal clarity, which is essential for the identification of precise and accurate neural signals, and for minimizing errors during command execution.

Despite these advancements, there exist many challenges that prevent us from developing a ”perfect” spike sorting. One example of this is the overlapping spikes, especially in densely packed recordings. Advanced template matching and clustering algorithms are being developed to resolve this issue, but are we still a step away from error-free systems. Another issue is neural drift. Neural signals can drift over time, affecting the accuracy of spike sorting. Implementation of drift correction methods is pivotal for the reliability of BCIs. The use of bench-marking and validation metrics is important to ensure that the algorithms can perform well across different datasets and external environmental conditions. [5]

In conclusion, the advances in spike sorting techniques have indeed improved the functionality and reliability of BCIs. As the field of neuroscience continues to develop, there is scope for a more responsive brain-computer interface to come.

A.2 Wavelet-based feature extraction

The project proposed by Quian Quiroga et al. [11] introduces a novel method for the identification and clustering of spikes by combining the wavelet transform for feature extraction with superparamagnetic clustering (SPC) for automatic classification. It was designed to automate the spike sorting method and also increase the efficiency and effectiveness of the system.

The wavelet transformation analyzed spikes by considering time and frequency, proving a holistic multi-domain representation of the signals. A four-level wavelet decomposition was implemented to get 64 wavelet units for each spike. The units represented the hierarchical structure of spike shape in different time scales. This transformation was sent to the feature extraction component to identify the most relevant wavelet units using the Kolmogorov-Smirnov (KS) test. The top 10 units that depicted more variance from normality were selected for clustering, as they would aid in providing the best separation of spike class. The SPC clustering used the selected wavelet units for clustering, it compared different threshold values to identify clusters. SPC did not assume low variance, non-overlapping cluster, or Gaussian distribution like the previously mentioned projects did (Section 3.2). The clustering method automatically determined the optimal threshold value by determining the starting point for the formation of largely dense and separated clusters [11].

Quian Quiroga et al. [11] also addressed the issues pertaining PCA (Section3.3) by selecting high-variance and appropriate features to represent the characteristics of spikes.

The combination of wavelet transformation with SPC improved the performance much more effectively than the aforementioned system which only depended on PCA. Since time-frequency domains were considered the decomposition of overlapping signals was achieved across multiple scales and resolutions. Utilization of wavelets for feature extraction gave the system the ability to capture and enhance the intrinsic variation in spike shapes. The wavelet units supported the time-oriented analysis of spikes as they were very accurate in identifying time stamps. This analysis was able to support differentiating partially overlapping spikes.

Contrary to this PCA compressed the information regarding spike shapes by projecting them to very few principal components, this dimensional reduction failed to notice necessary details as it fixated on increasing the variance rather than focusing on time oriented characteristics. This resulted in less clear clusters that were not able to identify the innate features of different spikes. Thus the study confirmed that wavelets provide better performance for spike sorting and maintain the shape information an important characteristic for classification.

Conclusion and Observations Overall, the advancements highlight the progressive refinement of spike sorting techniques towards greater automation, efficiency, and accuracy, paving the way for more robust and reliable analysis in neurophysiological research. Early implementations like Template Matching by Gerstein and Clark (1964) laid the foundation for spike sorting techniques but required significant manual intervention and struggled with overlapping spikes. The introduction of Bayesian Clustering by Lewicki (1994) marked a shift towards probabilistic models, providing better handling of overlapping spikes but still faced limitations due to the Gaussian noise assumption. ICA by Bell and Sejnowski (1995) elevated signal separation by assuming signal independence and non-Gaussian distribution, but it was questioned by real-world recording and noise handling. PCA used in the Multispike Train Analysis by Abeles et al. (1977) and later improved by Glaser et al. (1968) offered effective real-time processing and noise handling, but struggled with low-amplitude spikes and required complex basis function selection. Finally, the wavelet-based feature extraction method by Quiroga et al. (2004) demonstrated superior performance by leveraging time frequency domain analysis and automatic clustering, addressing many of the issues faced by PCA and offering a more holistic representation of spike characteristics.

A.3 Spectral Clustering

Spectral clustering uses the eigenvalue and eigenvectors of the similarity matrix for dimensionality reduction before performing clustering in low-dimensional space. The eigenvectors are the directions in which data is projected (components generated from the feature extraction methods) and the corresponding eigenvalues are representative of the variance along each of these directions. The eigenvalues are extracted from the Laplacian matrix (which is derived from the similarity matrix) and are used to understand the characteristics of the data. Small eigenvalues and their corresponding eigenvectors are used to effectively group similar spikes [58]. The pipeline's clustering function was set with a default of 3 clusters, utilizing the `SpectralClustering` algorithm from the `sklearn.cluster` library.

The `SpectralClustering` used the formula (A.1) to calculate the eigenvectors and eigenvalues that are central to the spectral clustering process. In this formula, L is the Laplacian matrix. Here, W is the similarity (affinity) matrix, where W_{ij} represents the similarity between data points i and j, and D is the degree matrix, a diagonal matrix where each entry D_{ii} is the sum of the corresponding row in W. Eigenvectors are the vectors obtained from the decomposition of the Laplacian matrix L. The corresponding eigenvalues are the magnitude of variance along each of these directions. [58]

$$L = D - W \tag{A.1}$$

A study conducted by Wood et al. [55] emphasizes that spectral clustering is needed for complex data like Gaussian distributed recordings (like extracellular recordings) as they can initialize parameters for such models. Determining parameters for the model to best fit the data helps in determining the most optimal way to separate spikes in clusters that represent individual neurons. They are resilient in noise filtration and can easily differentiate noise from spikes, resulting in the identification and grouping of only firing neurons. Since spectral clustering captures non-linear relationships they are ideal for refined spike systems wherein relationships between spikes are not uniform. These strengths might make spectral clustering a powerful tool for improving the accuracy and reliability of this pipeline.

A.4 Justification of Principal Component Selection Using Reconstruction Error Analysis

The graph A.1 illustrates the relationship between reconstruction error and the number of principal components. The graph tracks how the reconstruction error changes as the number of components varies, starting from 1 up to the dataset's maximum permissible components. On the x-axis, we see the number of components, while the y-axis indicates the reconstruction error, which initially begins around 100 and gradually decreases to approximately 20. A red dashed vertical line marks the proposed optimal number of components, which is identified as 2.

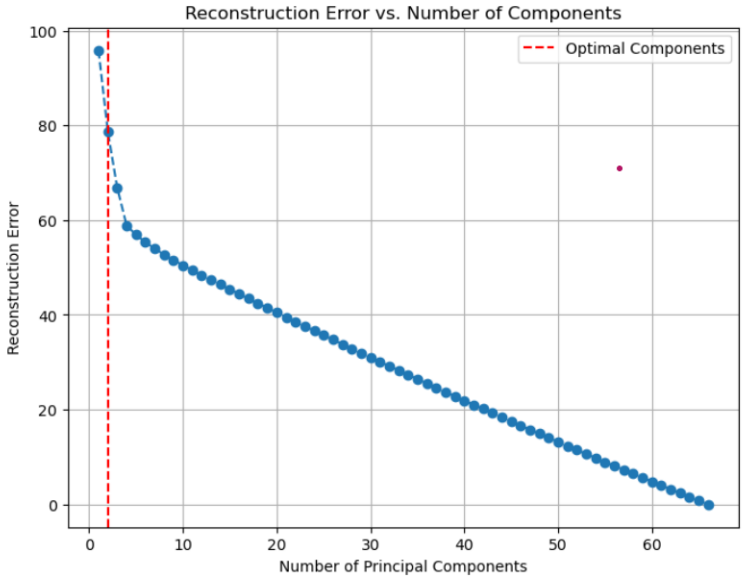

Figure A.1: Reconstruction Error vs. Number of Principal Components. The plot shows that the optimal number of components is 2, where the reconstruction error stabilizes, indicating minimal benefit from additional components.

The selection of 2 components is primarily justified by the significant initial drop in reconstruction error. As the number of components increases from 1 to 2, the error sharply decreases from nearly 100 to around 80. This substantial reduction suggests that the first two components capture most of the critical information within the data, resulting in a marked improvement in reconstruction accuracy. Beyond 2 components, the error reduction slows down significantly, indicating diminishing returns as more components are added.

The plot also showcases a distinct "elbow" at 2 components. The "elbow" marks the point where the curve transitions from a steep drop to a more gradual decline, which

is a common indicator in PCA analysis of the optimal number of components. Beyond this point, additional components contribute minimal improvements to reducing the error, making them less efficient.

This graph was generated by applying PCA to the dataset, varying the number of components, and calculating the reconstruction error for each setting. The red dashed line emphasizes that 2 components represent the optimal trade-off, capturing most of the essential information without introducing unnecessary complexity.

In summary, the graph demonstrates that using 2 principal components is the most efficient approach for this dataset. The steep drop in reconstruction error for the first two components, followed by a more gradual decline thereafter, confirms that most of the key information is captured early on. The clear presence of an ”elbow” further supports this conclusion, underscoring that 2 components strike the right balance between minimizing error and avoiding excessive complexity.

Appendix B

Additional Results

This appendix provides a detailed breakdown of the accuracy graphs corresponding to various clustering methods discussed in the main body of the dissertation. Each section delves into the performance of specific clustering algorithms—Agglomerative Clustering, DBSCAN, HDBSCAN, K-Means, and MeanShift—highlighting how the accuracy of each method is influenced by different parameter settings. The comprehensive graphs and accompanying explanations aim to visually represent and clarify the impacts of each parameter choice, such as the number of clusters and eps values, across different feature extraction techniques including PCA, ICA, UMAP, and Isomap. These detailed results underscore the subtleties of each approach, offering a clearer understanding of how each parameter adjustment affects the overall effectiveness of the spike sorting process.

B.1 Agglomerative Clustering

When analyzing the performance of Agglomerative Clustering Figure B.1, a noticeable trend emerges for PCA and ICA features: accuracy consistently improves as the number of clusters increases. For PCA, accuracy climbs from around 0.21 with 40 clusters to approximately 0.28 with 60 clusters. Similarly, for ICA, accuracy improves significantly from 0.41 to 0.60 as the number of clusters increases. This pattern suggests that more clusters help the algorithm capture finer distinctions within the data, particularly when using PCA or ICA for feature extraction. However, the story is different for Isomap features, where the accuracy steadily declines from 0.045 at 40 clusters to just 0.028 at 60 clusters. This highlights a key limitation of Isomap in high-dimensional spaces; it struggles to maintain clear separations as the number of clusters increases, leading to

less effective clustering.

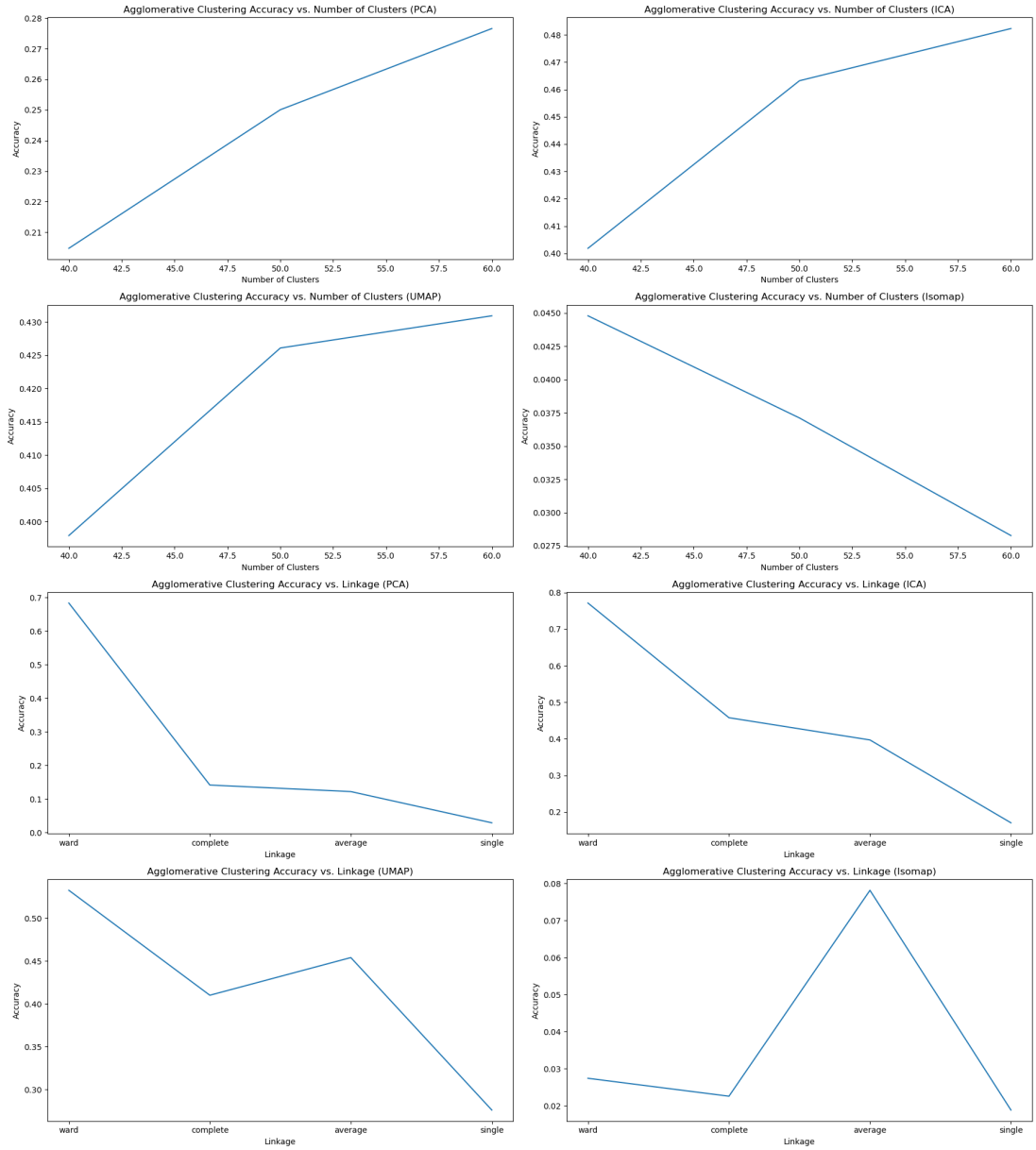

Figure B.1: Agglomerative Clustering Accuracy vs. Number of Clusters. The accuracy increases as the number of clusters increases for PCA and ICA features, reaching around 0.48 and 0.60 respectively for 60 clusters. However, for Isomap, accuracy drops from 0.045 at 40 clusters to around 0.028 at 60 clusters.

B.2 DBSCAN Clustering

DBSCAN's performance is closely tied to the choice of `eps` and `min_samples` parameters as seen in Figure B.2. As `eps` increases, accuracy improves across most feature sets,

with the most significant gains seen for ICA features, where accuracy rises from around 0.05 at `eps`=0.2 to a peak of 0.71 at `eps`=1.5. Similar trends, though less pronounced, are seen for PCA and UMAP features. However, when increasing `min_samples`, a consistent decline in accuracy is observed. For instance, with PCA features, accuracy falls from 0.28 at `min_samples`=5 to 0.16 at `min_samples`=15. This indicates that requiring more neighbors to define a core point can lead to over-smoothing, where meaningful small clusters are misclassified as noise.

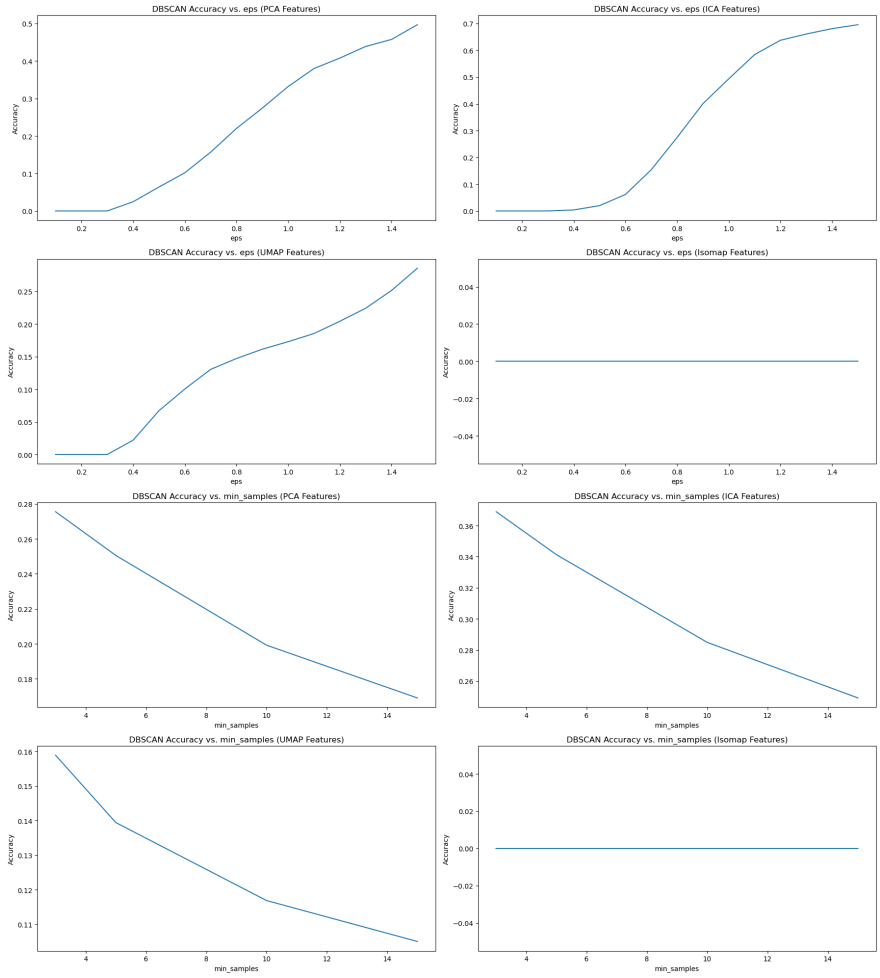

Figure B.2: DBSCAN Accuracy vs. Epsilon and Min Samples for Different Features. Accuracy rises steadily with increased `eps` across most feature sets, reaching 0.71 for ICA with `eps`=1.5. However, increasing `min_samples` causes a consistent drop in accuracy, particularly for PCA features, where accuracy decreases from 0.28 at `min_samples`=5 to 0.16 at 15.

B.3 HDBSCAN Clustering

HDBSCAN shows stable accuracy for PCA features Figure B.3, remaining around 0.77 across varying cluster sizes. This stability indicates that HDBSCAN is effective at identifying clusters regardless of small changes in the parameter settings, making it less sensitive compared to other methods. For UMAP features, accuracy initially rises from 0.53 at a cluster size of 5 to 0.60 at a cluster size of 20, but then begins to decline, highlighting the importance of fine-tuning the cluster size parameter. In contrast, for Isomap features, accuracy shows a continuous improvement as the cluster size increases, suggesting that HDBSCAN's adaptability to varying densities makes it more robust when working with challenging feature spaces like those generated by Isomap.

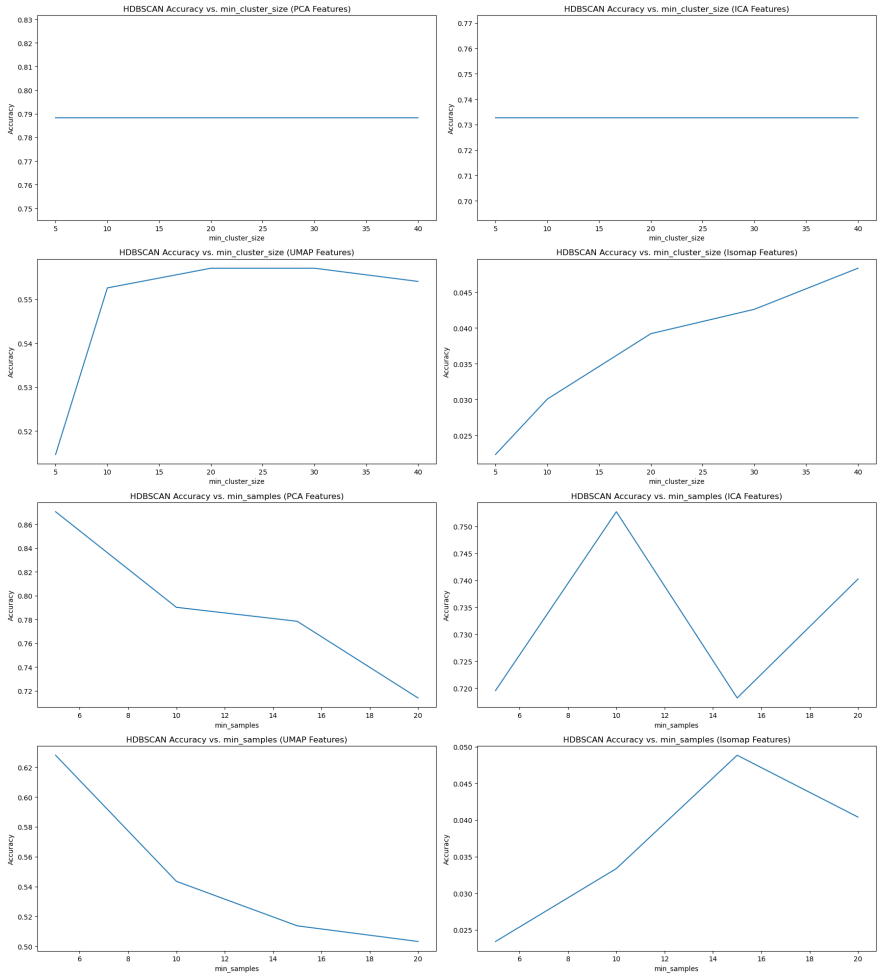

Figure B.3: HDBSCAN Accuracy vs. Min Cluster Size and Min Samples. The accuracy is relatively stable for PCA features around 0.77 across different cluster sizes. For UMAP, accuracy increases from 0.53 to 0.60 as `min_cluster_size` grows from 5 to 20, but declines thereafter.

The confusion matrix Figure B.4provides a clear visualization of the classification performance of HDBSCAN combined with PCA features. The strong diagonal pattern indicates that the majority of data points are correctly classified, reflecting the high accuracy observed for this combination. However, some off-diagonal elements are visible, representing misclassifications. These errors typically occur where clusters are close in feature space, leading to overlap in their boundaries. Despite these minor misclassifications, the overall structure suggests that this configuration is effective at distinguishing between distinct spike classes, which is further supported by the high precision and recall metrics observed.

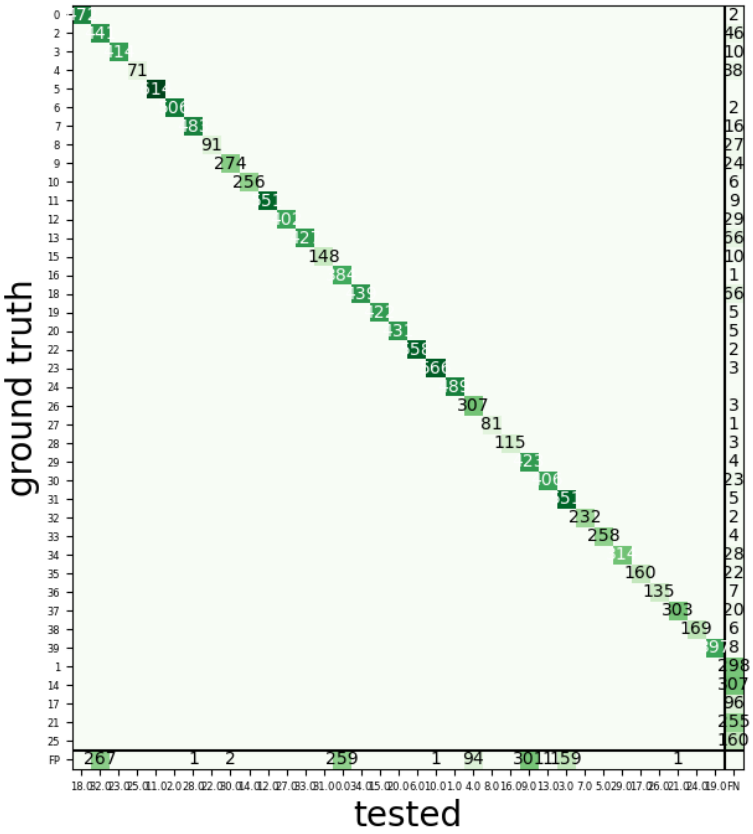

Figure B.4: Confusion Matrix for HDBSCAN with PCA Features. The matrix shows that most data points are correctly classified along the diagonal, indicating high accuracy. However, misclassifications are present, particularly where ground truth and predicted clusters are close in feature space.

The SNR vs. accuracy plot B.5 shows a strong correlation between higher SNR values and better classification accuracy. For SNRs above 10, the accuracy is consistently near 1.0, indicating almost perfect classification. Conversely, at lower SNR values

(below 5), there is a sharp decline in accuracy, with some points showing zero accuracy. This suggests that the algorithm struggles to correctly classify data when the signal is buried in noise. The plot highlights the importance of having a high SNR for reliable spike sorting and suggests that the success of HDBSCAN with PCA in this context is closely tied to the quality of the signal being analyzed.

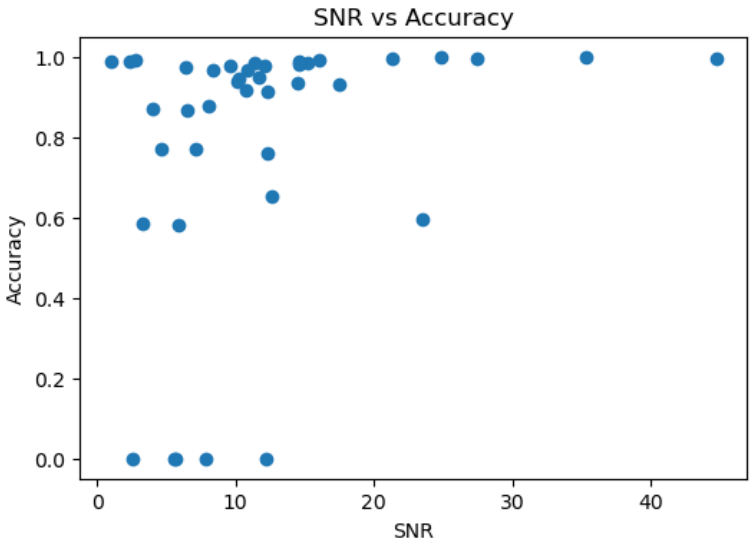

Figure B.5: SNR vs Accuracy for HDBSCAN with PCA Features. The plot demonstrates that high SNR values (greater than 10) are correlated with near-perfect accuracy. However, lower SNR values lead to significant drops in accuracy, with some points showing zero accuracy at very low SNRs.

B.4 K-Means Clustering

The performance of K-means is heavily influenced by the number of clusters and initializations, especially for ICA features as observed in Figure B.6. With 40 clusters and 20 initializations, accuracy peaks at around 0.86, demonstrating that having more clusters allows K-means to better capture the inherent structure of the data. However, the accuracy of Isomap features declines sharply as the number of clusters increases, falling below 0.1, indicating that Isomap is less compatible with K-means clustering. Additionally, varying the number of iterations has minimal impact on accuracy beyond a certain threshold, indicating that convergence is achieved relatively quickly.

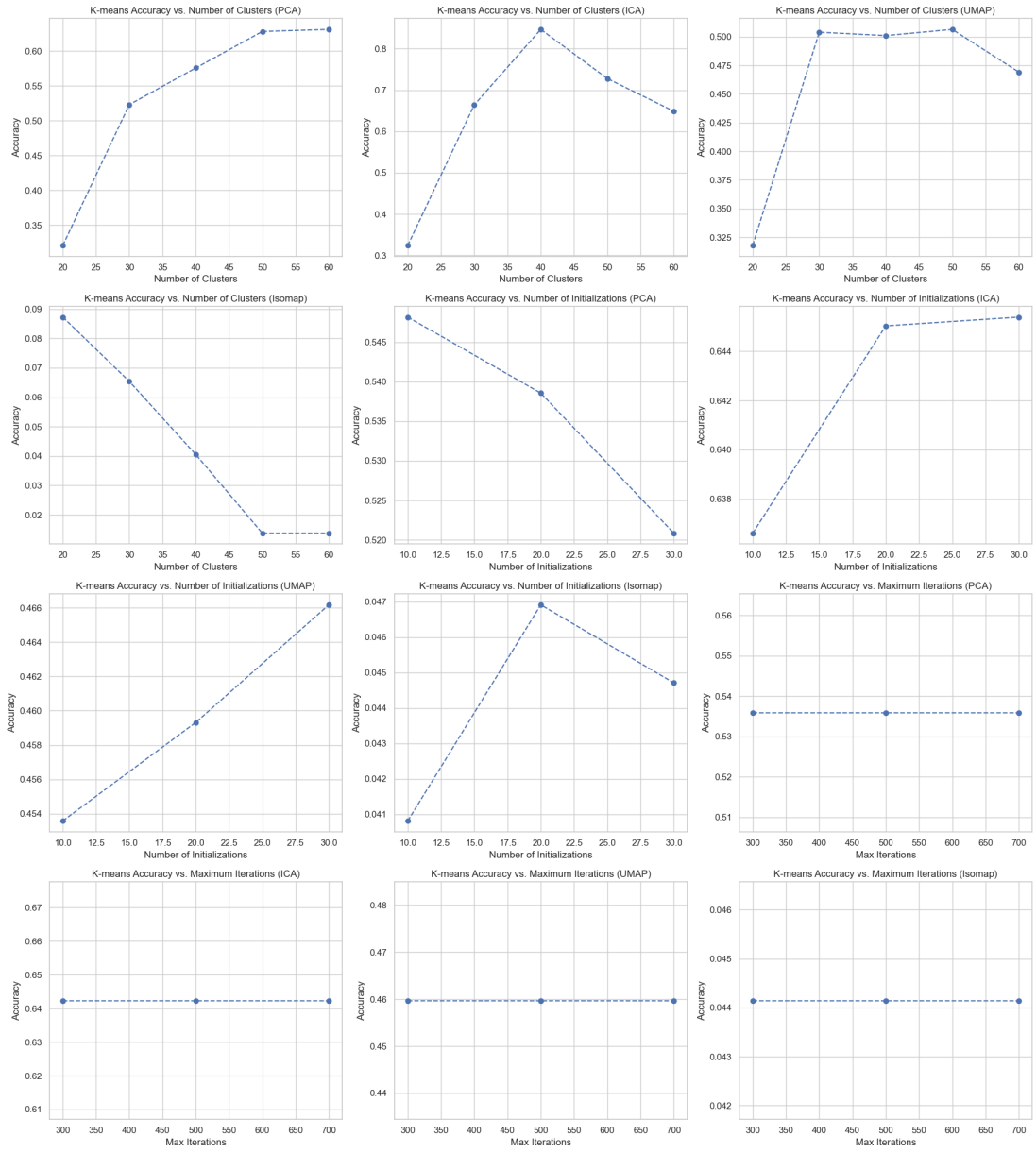

Figure B.6: K-means Accuracy vs. Number of Clusters, Number of Initializations, and Maximum Iterations. The plots show that for ICA features, accuracy peaks at around 0.86 with 40 clusters and 20 initializations, while accuracy remains constant despite varying iterations. For Isomap, accuracy falls below 0.1 as clusters increase.

B.5 MeanShift Clustering

From Figure B.7 MeanShift's performance is highly dependent on the bandwidth parameter. For ICA features, accuracy peaks at 0.68 with a bandwidth of 4.8, indicating that this setting is optimal for capturing clusters. However, Isomap features continue to

struggle, with accuracy barely improving from 0.02 to 0.03 even as the bandwidth increases. This suggests that Isomap's feature space does not align well with MeanShift's density-based approach, leading to poor clustering outcomes regardless of bandwidth adjustments.

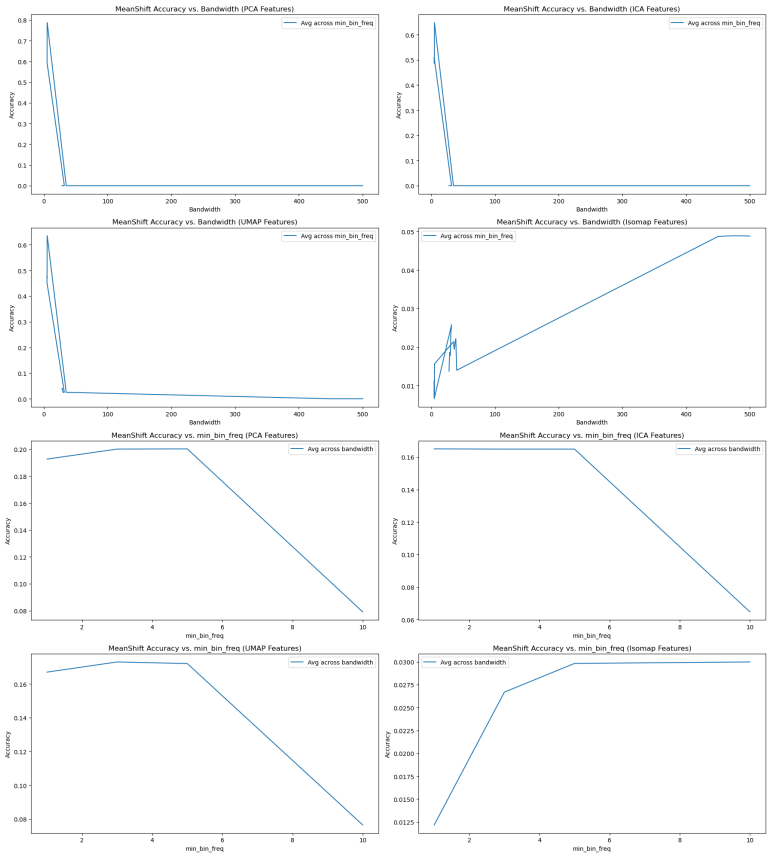

Figure B.7: MeanShift Accuracy vs. Bandwidth for Different Features. The accuracy peaks at around 0.68 for ICA features with `bandwidth`=4.8. However, for Isomap features, increasing bandwidth leads to only a minor improvement from 0.02 to 0.03.

The performance of MeanShift also varies with the minimum bin frequency (Figure B.8). For PCA features, accuracy reaches its highest point at 0.20 with a `min_bin_freq` of 2. However, for ICA features, accuracy peaks at 0.18 but drops sharply beyond a `min_bin_freq` of 4. This rapid decline indicates that the algorithm becomes too sensitive to noise as the bin frequency increases, leading to less effective clustering. UMAP features show a similar pattern, with optimal performance occurring at lower bin frequencies, highlighting that careful parameter selection is crucial for maintaining clustering quality.

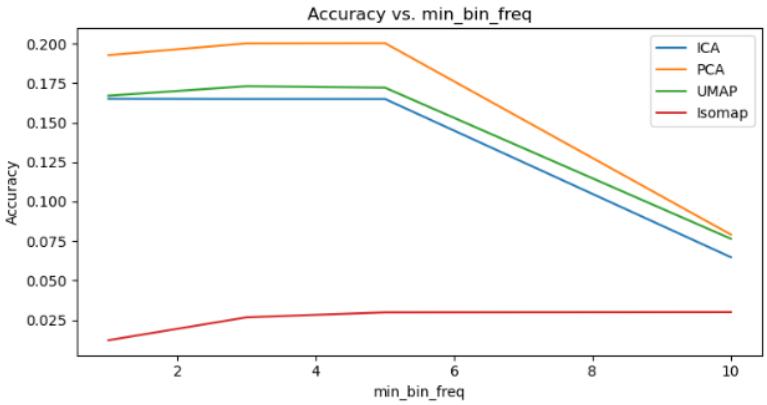

Figure B.8: MeanShift Accuracy vs. Min Bin Frequency for Different Features. Accuracy is highest for PCA features at around 0.20 with `min_bin_freq`=2. For ICA, accuracy peaks at 0.18 but drops sharply beyond `min_bin_freq`=4.