Deciphering Ion Channel Dynamics: A Clustering Approach for Signal Analysis

Mohammadreza Kazemi

A Thesis submitted to the Faculty of Graduate Studies in Partial Fulfillment of the Requirements for the Degree of Master of Science

Graduate Program in Electrical Engineering and Computer Science

York University

Toronto, Ontario

December 2024

Abstract

This thesis investigates clustering techniques for analyzing multi-channel CFTR ion channel activity recorded via the patch-clamp method. Accurately determining the number of active channels and classifying their states is crucial for understanding ion channel dynamics. Both classical and machine learning approaches are explored. Classical methods, including DBSCAN and a hybrid DBSCAN-BGMM approach, show limitations in handling noise and overlapping states, particularly with increasing channel counts. A novel machine learning approach combining a Cluster Count Neural Network (NN) for channel number estimation and a Long Short-Term Memory (LSTM) network for state classification demonstrates superior performance. The NN effectively captures underlying patterns while the LSTM leverages temporal dependencies, achieving higher accuracy and robustness even with complex, noisy signals. This research offers promising tools for analyzing ion channel activity and has implications for cystic fibrosis research and drug development.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

1.1.1 Ion Channels

In biology, cells serve as the building blocks of life, each responsible for distinct physiological functions. These functions are not executed in isolation; instead, they rely on sophisticated communication systems that involve the transmission of signals between cells and within cells themselves. This communication is orchestrated through the movement of ions—charged atoms—across cellular membranes [3]. Such ion movement is made possible by specialized proteins known as ion channels.

Consider a neuron in the brain firing an electrical signal to convey information. This process begins with the movement of ions, such as sodium (Na^+), potassium (K^+), and chloride (Cl^-), across the neuron's cell membrane[4]. These ions carry electrical charges, and their movement generates voltage changes that propagate along the neuron's length. Similarly, muscle cells' contraction relies on the orchestrated flow of calcium ions (Ca^2+), triggered by various cellular signals[5].

Beyond cell signalling, ion channels also play a crucial role in maintaining cellular homeostasis—the internal balance required for cells to function optimally. For instance, nerve cells maintain a resting membrane potential, which is the electrical charge difference across the cell membrane when the cell is not actively transmitting signals. This potential is maintained by the selective movement of ions through ion channels[6].

While computer scientists typically design algorithms and circuits to process and transmit information, the parallel to ion channels in biology is striking. In essence, ion channels can be regarded as biological signal processors that control the flow of ions, thereby generating electrical currents that transmit information within and between cells[7]. Just as computer scientists optimize algorithms for efficient data processing, cells have evolved to optimize ion channel activity for precise signal transmission.

Consider an ion channel as a molecular gate that can be opened or closed, allowing ions to move in response to specific signals. This gating mechanism is reminiscent of digital logic gates used in computer circuits, where the state of inputs determines the output. In the case of ion channels, the "inputs" could be voltage changes, chemical signals, or mechanical forces[6].

1.1.2 Cystic Fibrosis Transmembrane Conductance Regulator

The Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) is a highly significant ion channel in human biology that holds relevance for both the understanding of cellular physiology and the development of medical interventions [8]. To assist computer scientists in grasping this vital biological component, we delve into what CFTR is, how it functions, and its significance.

CFTR is a protein that acts as a channel for chloride ions (Cl^-) to move across cell membranes. It is found in various tissues, with a prominent presence in the epithelial cells lining the respiratory, digestive, and reproductive tracts[9]. The CFTR gene encodes this protein, and any mutations in this gene can lead to a condition known as cystic fibrosis (CF), a genetic disorder characterized by thick, sticky mucus production that can affect multiple organ systems[10].

The CFTR protein is a transmembrane channel, meaning it spans the cell membrane. Its structure consists of several domains, including two membrane-spanning segments and two nucleotide-binding domains (NBDs) that have ATP (adenosine triphosphate) binding sites. These ATP-binding domains are crucial for CFTR's function as an ion channel[11].

CFTR's primary function is to regulate the movement of chloride ions across the cell membrane. Normally, CFTR channels allow chloride ions to exit the cell into the extracellular space[12]. This movement of ions contributes to maintaining proper ion balance and hydration of the cell's environment.

Similar to the ion channels discussed earlier, CFTR channels can open and close in response to various signals. However, CFTR has a unique gating mechanism that involves ATP binding and hydrolysis. When ATP binds to the NBDs, the channel opens, allowing chloride ions to flow. The subsequent hydrolysis of ATP causes the channel to close again. This mechanism allows CFTR to finely regulate chloride ion movement[13].

Cystic fibrosis is caused by mutations in the CFTR gene that result in dysfunctional or absent CFTR channels. This leads to a disruption in chloride ion transport and, consequently, a thicker, stickier mucus in the affected organs[14]. This mucus accumulation can clog airways, impair digestion, and create an environment conducive to bacterial infections. The impact of CF is multi-faceted and can severely affect a patient's quality of life.

Understanding the molecular and functional aspects of CFTR has paved the way for therapeu-

tic interventions. Some medications aim to enhance CFTR function in individuals with certain mutations, while gene therapies are being explored to correct the underlying genetic defects in CF patients[15].

From a computational standpoint, CFTR can be thought of as a "molecular switch" that regulates chloride ion flow based on ATP binding and hydrolysis[16]. This mechanism involves complex molecular interactions that parallel the logical operations in computer circuits[17]. By understanding CFTR's structure, function, and implications, computer scientists can gain insights into the intricacies of biological signal processing, offering potential avenues for innovative interdisciplinary research[18].

The crossroads of ion channels and computer science offer exciting opportunities for interdisciplinary collaboration. By applying principles of signal processing and data analysis to the study of ion channel behaviour, computer scientists can contribute to deciphering the intricate language of cellular communication[19]. Furthermore, the insights gained from understanding ion channels could inspire novel computational models for simulating biological processes, potentially leading to groundbreaking discoveries in medicine, biotechnology, and beyond.

1.1.3 Patch Clamp Method

The patch clamp technique is a laboratory method that allows researchers to directly measure the flow of ions through individual ion channels on a cell's membrane. This technique, developed by Erwin Neher and Bert Sakmann in the late 1970s, revolutionized the study of ion channels and garnered them the Nobel Prize in Physiology or Medicine in 1991 [20].

In essence, the patch clamp technique involves creating a high-resistance seal between a glass pipette and a small portion of a cell's membrane. This creates a "patch" of the membrane that can be isolated and studied independently. By applying controlled voltage or pressure to the pipette, researchers can manipulate the ion channel's activity and record the resulting electrical currents [21].

While the patch clamp technique involves intricate biological processes, it is analogous to data acquisition in computer science. Imagine capturing precise voltage and current data from ion channels as "signals" and then analyzing this data to reveal the behaviour of these channels.

1.1.4 Problem statement

While the patch clamp technique offers a powerful means of studying individual ion channels, a significant challenge lies in determining the number of channels present within the patch. Ion channels are dynamic and can exist in various states of open and closed configurations. This dynamic behaviour complicates the estimation of channel counts and their associated state [22].

The uncertainty in channel counts and their state affects the interpretation of the recorded currents. Researchers often use statistical methods and mathematical models to estimate them based on observed current fluctuations. However, the accuracy of these estimates can vary, and there remains an inherent level of uncertainty.

1.1.5 Why is it of interest

Determining the number of ion channels and their respective states within a CFTR patch clamp recording is crucial, not just for understanding CFTR function, but also for advancing our ability to analyze and interpret these complex signals. Precisely isolating a single ion channel with a patch clamp is technically challenging. In practice, recordings often capture the activity of multiple channels simultaneously. This multi-channel activity introduces a significant layer of complexity to the analysis, making it essential to discern the contributions of individual channels.

Accurately identifying the number of channels and their transitions is paramount for effective signal denoising. By separating the overlapping activity of multiple channels, we can significantly reduce the noise and artifacts in the recordings, leading to more precise estimations of key channel parameters such as conductance, dwell times, and transition rates.

These parameters are not static; they can be modulated by various factors, including disease-causing mutations or therapeutic interventions. For instance, changes in CFTR channel gating parameters can be indicative of channel dysfunction. Similarly, the efficacy of drugs targeting CFTR can be assessed by monitoring their impact on these same parameters. Therefore, accurate parameter estimation, facilitated by knowing the number of channels present, is essential for both understanding disease mechanisms and developing effective therapies. Furthermore, precise determination of channel counts and states is a prerequisite for building accurate computational models of CFTR activity. These models are crucial for advancing our fundamental understanding

of cellular communication and predicting how mutations, drugs, and other conditions may affect channel behavior.

1.2 Literature Review

The analysis of ion channel recordings has always been intertwined with the challenge of noise reduction and signal interpretation. Early research grappled with limitations in recording technology and developed various techniques to extract meaningful information from noisy signals. We begin by reviewing these foundational methods, focusing on filtering and averaging techniques, before moving on to more recent statistical and machine-learning approaches.

1.2.1 Filtering Techniques

Filtering techniques have been pivotal in the pursuit of unveiling the symphony of cellular communication obscured by noise. Noise, arising from various sources such as electrical interference, thermal fluctuations, and instrument imperfections, poses a significant challenge to accurately capturing and interpreting ion channel currents. Filtering techniques emerged as early tools to enhance signal clarity, paving the way for deeper insights into ion channel behaviour. Early filtering techniques involved analog circuitry, where components like capacitors and resistors were used to shape the frequency response of the filter. With the advent of digital signal processing (DSP), filtering transitioned to the digital realm. Digital filters, implemented through algorithms and software, offer greater flexibility and precision, allowing researchers to fine-tune filtering parameters.

1.2.1.1 Low-Pass Filters

Low-pass filters are among the foundational tools in noise reduction. They function based on a straightforward principle: allowing low-frequency signals to pass through while attenuating high-frequency noise. Essentially, these filters "capture" the signals of interest while "discarding" the undesirable high-frequency interruptions. To envision this, imagine hearing the gentle rustling of leaves distinctly even amidst the high-pitched static of radio interference.

The mathematical representation of a simple low-pass RC (resistor-capacitor) filter's frequency

response is given by:

$$H(f) = \frac{1}{1 + j\left(\frac{f}{f_c}\right)}$$

Where:

- $H(f)$ is the filter's response.
- f is the input frequency.
- f_c is the cutoff frequency, defined as $f_c = \frac{1}{2\pi RC}$.
- j is the imaginary unit.
- R is the resistance and C is the capacitance of the filter.

When f is much smaller than f_c, $H(f)$ approaches 1, implying that low-frequency signals pass through the filter largely unchanged. Conversely, as f becomes much larger than f_c, $H(f)$ diminishes, thereby attenuating high-frequency signals.

1.2.1.2 Filtering and Frequency Domains

The key principle of filtering lies in manipulating signals across different frequency domains. Every signal can be decomposed into its frequency components using the Fourier Transform. The representation of a signal in the frequency domain offers insights into which frequencies dominate the signal. Mathematically, this transformation can be represented by:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt$$

Where:

- $X(f)$ is the representation of signal $x(t)$ in the frequency domain.
- f denotes frequency.
- j is the imaginary unit.

Noise often occupies higher frequency ranges, while ion channel signals are predominantly in lower frequency ranges. By analyzing the spectral components of the recorded signal, researchers can determine the optimal cutoff frequency for a low-pass filter.

For example, if a Fourier analysis reveals that a signal has meaningful components up to 200 Hz and noise predominantly starts from 250 Hz onwards, a low-pass filter with a cutoff around 220 Hz can be designed to retain the ion channel signals while attenuating the noise.

By implementing filters that selectively remove or attenuate high-frequency components, such as the one described above, researchers could effectively strip away the noise, revealing the underlying ion channel currents. This procedure has been instrumental in ensuring the clarity of recordings and facilitating accurate interpretations in electrophysiological studies.

1.2.1.3 Cascade and Multistage Filters

To achieve optimal noise reduction, researchers often employ cascade and multistage filtering. This involves applying multiple filters in series, each tailored to address specific noise components or to achieve certain desired modifications in the signal. When filters are cascaded, their individual frequency responses multiply, leading to a combined frequency response that can be expressed as:

$$H_{\text{total}}(f) = H_1(f) \times H_2(f) \times \cdots \times H_n(f)$$

Where:

- $H_{\text{total}}(f)$ represents the overall frequency response.
- $H_i(f)$ denotes the frequency response of the i^{th} filter.
- f is the frequency.
- n is the total number of filters cascaded.

For instance, consider an experiment where a researcher wants to first eliminate high-frequency noise and then correct for a slow baseline drift caused by equipment limitations:

1. A low-pass filter might first be employed with a cutoff frequency $f_{\text{c,lp}}$ to reduce high-frequency noise.
2. This can be followed by a high-pass filter with a cutoff frequency $f_{\text{c,hp}}$ to remove any low-frequency drifts.

When these filters are applied sequentially, the signal will primarily have frequency components between $f_{\mathrm{c,hp}}$ and $f_{\mathrm{c,lp}}$.

Another powerful configuration is the use of a band-pass filter, which essentially combines a low-pass and a high-pass filter, allowing only a specific frequency range to pass through. If there's a particular frequency range of interest, say between f_{start} and f_{end}, the band-pass filter can be tailored to isolate this specific range while attenuating frequencies outside it.

In practice, choosing the correct sequence and types of filters, as well as their specific cutoff frequencies, requires a comprehensive understanding of the signal's characteristics and the nature of the noise contaminating it. Cascade and multistage filtering thus provide a versatile toolkit to dissect and refine signals, ensuring clarity and precision in the acquired data.

1.2.1.4 Modern Techniques: Wavelet and Adaptive Filtering

Modern ion channel research benefits significantly from advanced filtering techniques such as wavelet transforms and adaptive filters. These methodologies provide sophisticated means of enhancing signal quality, which is pivotal for precise analysis and interpretation.

Wavelet Transforms: Unlike the traditional Fourier transform, which decomposes a signal into its sinusoidal components, the wavelet transform uses wavelets - small wave functions with finite energy. This allows for simultaneous localization in both time and frequency domains. The Continuous Wavelet Transform (CWT) is given by:

$$CWT_x(a,b) = \int_{-\infty}^{\infty} x(t)\psi^*\left(\frac{t-b}{a}\right)dt$$

Where:

- $x(t)$ is the signal of interest.
- $\psi(t)$ represents the wavelet function.
- a and b are the scaling and translation parameters, respectively.
- The asterisk denotes the complex conjugate.

An example benefit in ion channel research is the detection of short, transient events, like

channel openings, that may be obscured in noisy recordings. Wavelets, being localized in time, can effectively highlight such events without losing information on their timing.

Adaptive Filters: These filters are dynamic, adjusting their parameters in real-time based on the characteristics of the incoming data. A common algorithm employed in adaptive filtering is the Least Mean Squares (LMS). The update rule for the filter's coefficients in LMS is:

$$w(n+1) = w(n) + \mu e(n) x(n)$$

Where:

- $w(n)$ is the weight vector at time n.
- μ is the step-size parameter.
- $e(n)$ represents the error signal, often the difference between the desired and the filter's output.
- $x(n)$ is the input signal vector.

An application in ion channel research might involve the tracking of slow baseline drifts in the recorded signals. As the nature of the drift changes, the adaptive filter can modify its coefficients in real-time to continuously compensate for the drift, preserving the details of the ion channel activity.

Incorporating these advanced filtering techniques has facilitated a deeper exploration of cellular communication mechanisms, enabling researchers to obtain clearer, more accurate recordings even in challenging experimental conditions.

1.2.1.5 Trade-offs: Signal Integrity vs. Temporal Resolution

One of the challenges in filtering is striking a balance between preserving the integrity of ion channel signals and maintaining temporal resolution. Aggressive filtering can dampen noise but may distort or smoothen the true ion channel behaviour. Researchers must carefully design filters to reduce noise while minimizing signal distortion and preserving the dynamic nature of ion channel activity.

1.2.2 Averaging Techniques

Averaging methods exploit the repetitive nature of biological phenomena to mitigate the impact of noise. By collecting multiple instances of ion channel recordings and averaging them, researchers

can enhance the signal-to-noise ratio. This principle is akin to observing a celestial object through a telescope: multiple observations help separate the faint object from the twinkling stars.

1.2.2.1 Temporal Averaging

Temporal averaging enhances signal clarity by utilizing repetitive behavior within biological experiments.

Principle: Multiple time-aligned recordings are averaged point-by-point. Since the ion channel signal remains consistent over repetitions, but noise varies, the noise diminishes upon averaging. The averaged signal, $S_{\text{avg}}(t)$, from N repetitions is:

$$S_{\text{avg}}(t) = \frac{1}{N} \sum_{i=1}^{N} S_i(t)$$

Where:

- $S_i(t)$ is the i^{th} recording at time t.
- N is the total number of repetitions.

1.2.2.2 Ensemble Averaging

Ensemble averaging aggregates ion channel data across different experiments or cells, mitigating individual anomalies and highlighting consistent ion channel behavior.

Principle: Recordings from various experiments or cells are averaged. Each recording captures the same ion channel activity, but under different conditions. The ensemble-averaged signal, $E_{\text{avg}}(t)$, from N different recordings is:

$$E_{\text{avg}}(t) = \frac{1}{N} \sum_{i=1}^{N} E_i(t)$$

Where:

- $E_i(t)$ is the i^{th} recording at time t.
- N is the total number of independent recordings.

While individual recordings might exhibit variations, ensemble averaging prioritizes consistent responses, offering a reliable perspective on ion channel behavior.

1.2.2.3 Trade-offs: Temporal Resolution vs. Signal Enhancement

While averaging methods excel in noise reduction, they come with a trade-off between temporal resolution and signal enhancement. Averaging over longer time intervals increases signal clarity but sacrifices temporal detail. This trade-off must be carefully considered based on the ion channel's kinetics and the research objectives.

1.2.3 Statistical Approaches

Statistical approaches recognize that ion channel behaviour involves inherent randomness and uncertainty. Ion channels exhibit stochastic gating, where the transitions between open and closed states are influenced by probabilistic mechanisms. Statistical methods leverage this inherent variability to differentiate signal from noise.

1.2.3.1 Single-Channel Analysis

Bert Sakmann and Erwin Neher, pioneers of the patch clamp technique, introduced the concept of single-channel analysis. They realized that by examining single-channel recordings, researchers could uncover the probabilistic patterns governing ion channel gating. This approach involves idealizing the raw current traces into open and closed states, enabling researchers to estimate the probabilities of these states.

1.2.3.2 Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) are statistical models used to infer the hidden states of a system based on observed data. In the context of ion channels, HMMs are employed to infer the hidden states (open, closed) of the channel based on the noisy current recordings. HMMs take into account the statistical properties of ion channel transitions, allowing researchers to extract meaningful insights even from recordings heavily tainted by noise.

1.2.3.3 Inference on Factor Graphs for Patch Clamp Analysis

A more recent approach leverages the power of factor graph-based inference algorithms for analyzing patch clamp data. This method applies the sum-product algorithm on a factor graph representing the ion channel's hidden states, allowing for efficient and accurate estimation of state probabilities. Crucially, this method combines inference with the Expectation-Maximization (EM) algorithm for parameter estimation, enabling simultaneous inference and learning from the data. A key limitation of this approach, however, is its applicability only to single-ion channel recordings. In the presence of multiple channels within the patch clamp, the model's assumptions are violated, leading to inaccurate results. This limitation directly motivates the present work, which aims to develop clustering methods capable of accurately analyzing multi-channel patch clamp recordings, extending the analytical capabilities beyond the constraints of single-channel analysis. By addressing the multi-channel scenario, we aim to unlock the full potential of patch clamp data for understanding complex ion channel dynamics.

1.3 Objectives

Develop clustering algorithms using patch clamp traces from CFTR to detect the number of CFTR channels present in the trace. Both classical and neural network techniques will be employed, and the performance of each algorithm will be validated using synthetic and real patch clamp data. The evaluation will focus on how each algorithm performs under various noise conditions.

1.3.1 Significance and Applications

1.3.1.1 Signal Recovery and Feature Extraction

The proposed algorithm has the potential to recover meaningful components from noisy signals, aiding in signal denoising, feature extraction, and event detection. This has applications in various fields, including bioinformatics, telecommunications, and image processing.

Ion Channel Studies: In patch clamp experiments, the algorithm's ability to infer ion channel count can offer a valuable tool for biophysicists and cell biologists. This can streamline the process of identifying ion channel states and characterizing their behaviour under different conditions.

Cross-Disciplinary Implications: Beyond ion channel research, this algorithm has broader applications. It can be applied to various digital signal types to segment and decode complex patterns, making it relevant in neuroscience, genetics, sensor data analysis, and beyond.

1.3.2 Expected Outcomes

This research successfully achieved its objectives by developing and implementing novel clustering algorithms for analyzing multi-channel CFTR patch clamp data. Specifically, the following outcomes were realized:

- **Development of Classical and Machine Learning Clustering Algorithms**: Two distinct approaches were developed: 1) A refined classical clustering method combining Density-Based Spatial Clustering of Applications with Noise (DBSCAN) with Bayesian Gaussian Mixture Modeling (BGMM) to address the limitations of DBSCAN in handling noise and overlapping states. 2) A novel machine-learning based approach incorporating a Cluster Count Neural Network (NN) to predict the number of active channels, followed by a Long Short-Term Memory (LSTM) network to classify individual data points into their respective states, leveraging temporal dependencies in the signal.

- **Software Implementation and Validation**: Both the classical and machine learning algorithms were implemented in software. Their performance was rigorously validated using synthetic datasets generated from a realistic CFTR kinetic model, enabling controlled experiments and precise evaluation of accuracy under varying noise conditions and channel counts.

- **Demonstration of Improved Performance**: The developed algorithms demonstrated significantly improved performance compared to existing methods, particularly in scenarios with multiple ion channels and noisy data. The machine-learning approach, in particular, exhibited superior accuracy and robustness in identifying the correct number of channels and classifying their states. This improvement was achieved through the NN's ability to capture underlying patterns in the data and the LSTM's ability to leverage temporal dependencies, overcoming the limitations of traditional clustering methods.

Chapter 2

Methodology

2.1 Physical model

This section outlines the methodologies employed for generating synthetic patch-clamp data necessary for validating and testing computational methods. Synthetic data serves as a controlled environment where the underlying states and noise characteristics are precisely known, thus providing an essential tool to benchmark the performance of various computational algorithms[17].

2.1.1 Physiology of CFTR

The Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) is a chloride ion channel located on the apical membrane of epithelial cells. It plays a crucial role in regulating the transport of chloride and bicarbonate ions across the cell membrane, affecting fluid balance and viscosity in tissues such as the lungs, pancreas, and intestines [23]. CFTR is a member of the ATP-binding cassette (ABC) transporter family, unique in that it functions as an ion channel rather than a transporter [24].

Structurally, CFTR consists of two nucleotide-binding domains (NBDs) [25], two membrane-spanning domains (MSDs) [26], and a regulatory (R) domain [27]. The gating of the channel is controlled by the binding and hydrolysis of ATP at the NBDs and phosphorylation of the R domain by protein kinase A (PKA) [28]. ATP binding at the NBDs promotes channel opening, allowing the passage of chloride ions [24]. Hydrolysis of ATP at one of the NBDs triggers channel closure, and the cycle repeats. Mutations in the CFTR gene, such as the common ΔF508 mutation, lead to defective protein folding and impaired channel function, resulting in cystic fibrosis [29].

CFTR operates in cycles between open and closed states, with transitions between these states driven by ATP binding, ATP hydrolysis, and the conformational changes that follow. The regulation of ion flow through CFTR directly affects the fluid movement across epithelial surfaces, essential for maintaining proper tissue hydration and mucus consistency [30].

2.1.2 Patch-Clamp Technique and Signal Properties

Patch-clamp electrophysiology is a powerful technique used to study the activity of ion channels such as CFTR by measuring the ionic currents that flow through individual channels. A glass pipette with a fine tip is used to isolate a small patch of membrane containing one or more ion channels,

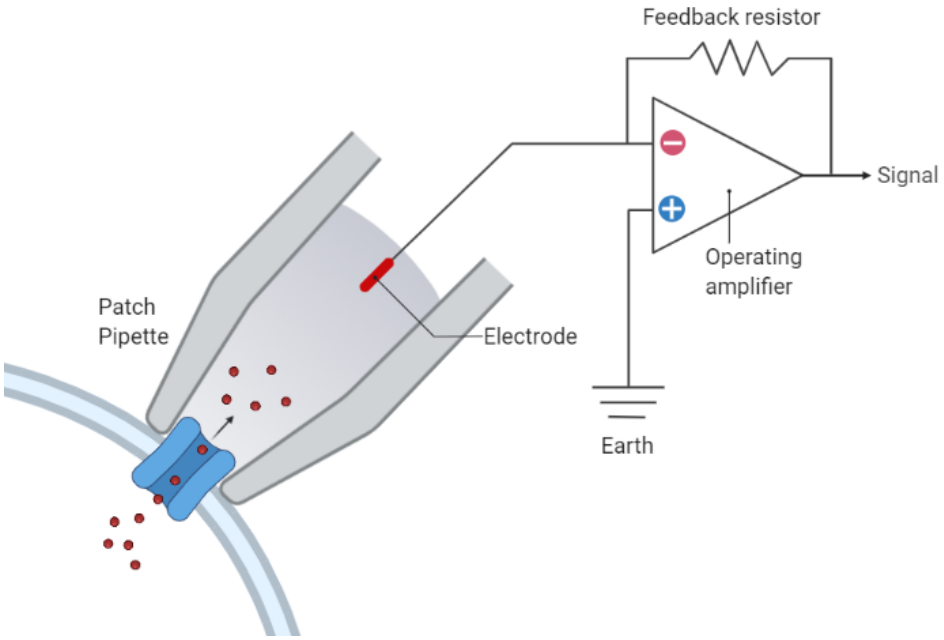


Figure 2.1: This image illustrates a patch clamp setup used in electrophysiology. It shows a patch pipette forming a seal with a cell membrane, connected to an electrode. The electrode feeds into an operational amplifier circuit with a feedback resistor, used to measure and amplify electrical signals from the cell. The diagram demonstrates how the patch clamp technique allows for the recording of ion channel activity in individual cells. Adapted from [1]

and an electrode within the pipette records the ionic current flowing through these channels in response to voltage changes [31].

The electrical current generated by ion channels is typically measured in picoamperes (pA) and is subject to various types of noise. These noise sources include thermal noise, electrode noise, and biological noise from the cell membrane. As a result, the recorded signals exhibit a combination of true channel activity and noise. The patch-clamp technique is well-suited to detect the minute current fluctuations corresponding to channel opening and closing events [32]. These signals, however, are influenced by noise, which complicates the interpretation and necessitates advanced signal processing techniques.

2.1.3 Receptor Model

This study utilizes a seven-state physical model of CFTR, as illustrated in Figure 2.2 and detailed in [2]. In this model, states C1a, C1b, C2, C3, and C4 represent fully closed states, characterized by the absence of ion flow, resulting in zero conductance. Conversely, states O1 and O2 are open

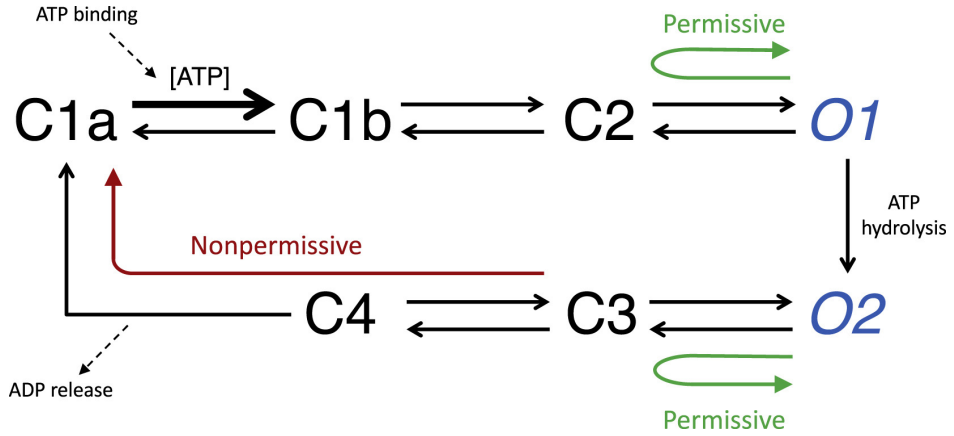


Figure 2.2: Seven-state kinetic model of CFTR gating. The model depicts transitions between open (blue italics) and closed (black) states. Black arrows represent possible state transitions. The bold arrow indicates ATP-dependent transition from C1a to C1b labeled as [ATP]. Green arrows highlight permissive closings where reopening occurs in the same open state. A red arrow illustrates a nonpermissive closing with a transition from open state O2 to O1. Adapted from [2].

states, permitting ion flow and exhibiting non-zero conductance. The labeling convention denotes closed states with 'C' and open states with 'O' [2].

The model's dynamics are initiated in state C1a with a single ATP molecule bound at the first ATP binding site. This site, incapable of hydrolysis, transitions reversibly to state C1b upon the binding of a second ATP molecule, resulting in both binding sites being occupied. This step's rate is dependent on the concentration of ATP. Subsequent reversible transitions from C1b to C2 and from C2 to O1 represent conformational changes that lead to the opening of the CFTR channel, similar to ligand-gated channels like the acetylcholine receptor [2].

The first irreversible step in the cycle occurs during the transition from O1 to O2, where one of the NBD-bound ATP molecules undergoes hydrolysis to ADP. Following this, CFTR can undergo reversible conformational changes from O2 to C3 and from C3 to C4, resulting in a closed pore. Finally, the second irreversible step takes place during the transition from C4 to C1a, involving the unbinding of the NBD-bound ADP molecule, leaving one unoccupied ATP binding site [2].

This seven-state model aligns with the simplified four-state cyclic gating model by effectively representing the essential features of CFTR gating, namely the cyclical transitions between open and closed states driven by ATP binding and hydrolysis. While the four-state model provides a simplified representation, this seven-state model incorporates additional closed and/or open states observed experimentally, offering a more granular and nuanced description of CFTR's complex

gating dynamics. The seven states can be mapped onto the four states by grouping similar conductance levels, thus preserving the overall cyclical nature of the gating process while accounting for the more detailed kinetic microstates [2].

Mathematically, the kinetic microstates of CFTR can be modeled using a master equation:

$$\frac{d\mathbf{P}}{dt} = \mathbf{PR} \tag{2.1}$$

where:

- $\mathbf{P}$ is a row vector representing the probability distribution of the system being in each of the kinetic microstates. The length of this vector is equal to the number of microstates in the model.
- $\mathbf{R}$ is the transition rate matrix, a square matrix where each element R_{ij} denotes the rate of transitioning from state i to state j.

2.1.4 Patch-Clamp Signal Model

Let $V = \{0, 1\}$ represent the set of observable conductance states, where 0 and 1 correspond to the closed and open states of the ion channel, respectively. Let $S = \{C1a, C1b, C2, C3, C4, O1, O2\}$ denote the set of hidden kinetic microstates [2].

The mapping from the hidden microstates to the observable conductance states, $m : S \to V$, is defined as follows:

$$m(s) = \begin{cases} 0, & \text{if } s \in \{C1a, C1b, C2, C3, C4\} \\ 1, & \text{if } s \in \{O1, O2\} \end{cases} \tag{2.2}$$

The patch-clamp technique measures the current flowing through the ion channel in the presence of noise. The recorded signal is then sampled at a rate of $1/\Delta t$ to generate a discrete-time signal. We represent this signal as $\mathbf{y} = [y_1, y_2, ..., y_N] \in \mathbb{R}^N$, where y_k is the measured current at time step k and N is the total number of samples [2].

The corresponding sequence of hidden microstates is represented as $\mathbf{s} = [s_1, s_2, ..., s_N] \in S^N$, where s_k is the true microstate of the channel at time step k.

The relationship between the observed signal and the hidden state sequence is modeled as follows:

$$y_k = Am(s_k) + n_k \tag{2.3}$$

where:

- A is a scaling factor representing the current amplitude when the channel is open.
- $m(s_k)$ is the conductance of the channel in state s_k as defined earlier.
- n_k is the noise present in the measurement at time step k.

We model the noise n_k using the Generalized Hyperbolic (GH) distribution. While a Gaussian distribution is a common simplification for noise in patch-clamp recordings, our analysis revealed non-Gaussian characteristics in the observed noise, specifically heavy tails and a slight negative skew, as evident in Figure 2.3. The GH distribution offers greater flexibility in capturing various noise shapes, including symmetric, asymmetric, heavy-tailed, and light-tailed distributions. This makes it a more suitable choice for accurately representing the noise in our data [33]. The GH distribution is characterized by four parameters: λ (shape), α (tail heaviness), β (skewness), δ (location), and μ (scale). We estimated these parameters from the real patch-clamp data using the method of moments for both open and closed states, allowing for state-dependent noise characteristics.

The transitions between the hidden microstates are modeled using a discrete-time Markov chain [2], where the transition probability from state i at time $k-1$ to state j at time k is given by:

$$Q_{ij} = P(s_k = j | s_{k-1} = i) \tag{2.4}$$

This transition probability matrix $\mathbf{Q}$ is related to the transition rate matrix $\mathbf{R}$ through the following equation:

$$\mathbf{Q} = \exp(\Delta t \mathbf{R}) \tag{2.5}$$

2.2 Data Simulation

To validate our inference and parameter estimation algorithms, we generated synthetic patch-clamp data using the CFTR kinetic model. This data generation process involved the following steps:

1. **State Sequence Generation:** The state sequence $\mathbf{s}$ represents the series of hidden states of the ion channel (e.g., open, closed) over time. To generate this sequence, we utilize a Markov process governed by a transition rate matrix $\mathbf{R}$. Initially, the system starts in a random state, typically selected based on a stationary distribution or uniformly at random across all possible states [2]. For each state s_k at time step k, the holding time (the time spent in that state) is sampled from an exponential distribution with a rate parameter determined by the sum of the transition rates out of that state. The next state is then selected probabilistically, where the probability of transitioning from state s_k to another state s_{k+1} is determined by the normalized row of the transition rate matrix $\mathbf{R}$ corresponding to s_k [2]. This process is iteratively repeated until the entire sequence of hidden states is generated.

2. **Signal Generation:** Once the state sequence $\mathbf{s}$ has been generated, the corresponding output signal $\mathbf{y}$ is computed based on the conductance values associated with each state. Each state s_k has an associated conductance level $m(s_k)$, which is either a fixed value or drawn from a distribution specific to that state. To generate the signal for each time step k, the conductance $m(s_k)$ is multiplied by a scaling factor A, which represents the amplitude of the conductance-to-current conversion, to produce a noiseless signal. Generalized Hyperbolic (GH) distributed noise GH(λ, α, β, δ, μ) with the estimated parameters is then added to the signal to simulate experimental noise, resulting in the final signal value y_k. The overall signal generation process thus reflects the stochastic transitions between states and the variability introduced by experimental noise, yielding a realistic synthetic patch-clamp trace.

This process generated synthetic patch-clamp data that closely resembled real-world recordings, exhibiting both realistic state transitions and noise characteristics.

2.2.1 Noise Function

Our initial attempt involved using the standard Gaussian distribution to model the noise in both open and closed states. However, a comparison of the distribution of the real data with the fitted Gaussian distribution revealed a poor fit, indicating that the Gaussian assumption was not valid. This motivated us to explore more flexible distributions that could better capture the characteristics of the noise.

After evaluating various distributions, we identified the Generalized Hyperbolic (GH) distribution as a suitable candidate. The GH distribution is a versatile distribution that encompasses a wide range of shapes, including symmetric, asymmetric, heavy-tailed, and light-tailed distributions [33]. It is characterized by four parameters:

- λ: Shape parameter.
- α: Tail heaviness parameter.
- β: Skewness parameter.
- δ: Location parameter.
- μ: Scale parameter.

By adjusting these parameters, the GH distribution can be tailored to fit a wide variety of data. We used the method of moments to estimate the parameters of the GH distribution from the real data for both open and closed states.

We then incorporated the state-dependent GH noise model into our synthetic data generation process. This resulted in a significant improvement in the realism of the synthetic data, as the noise now more accurately reflected the characteristics of the noise observed in real patch-clamp recordings.

The following table summarizes the parameter values used for data generation:

Using these parameters, we generated synthetic patch-clamp data and applied our inference and parameter estimation algorithms. The results demonstrated that our algorithms could accurately estimate the hidden state sequence and the model parameters, even in the presence of realistic non-Gaussian noise.

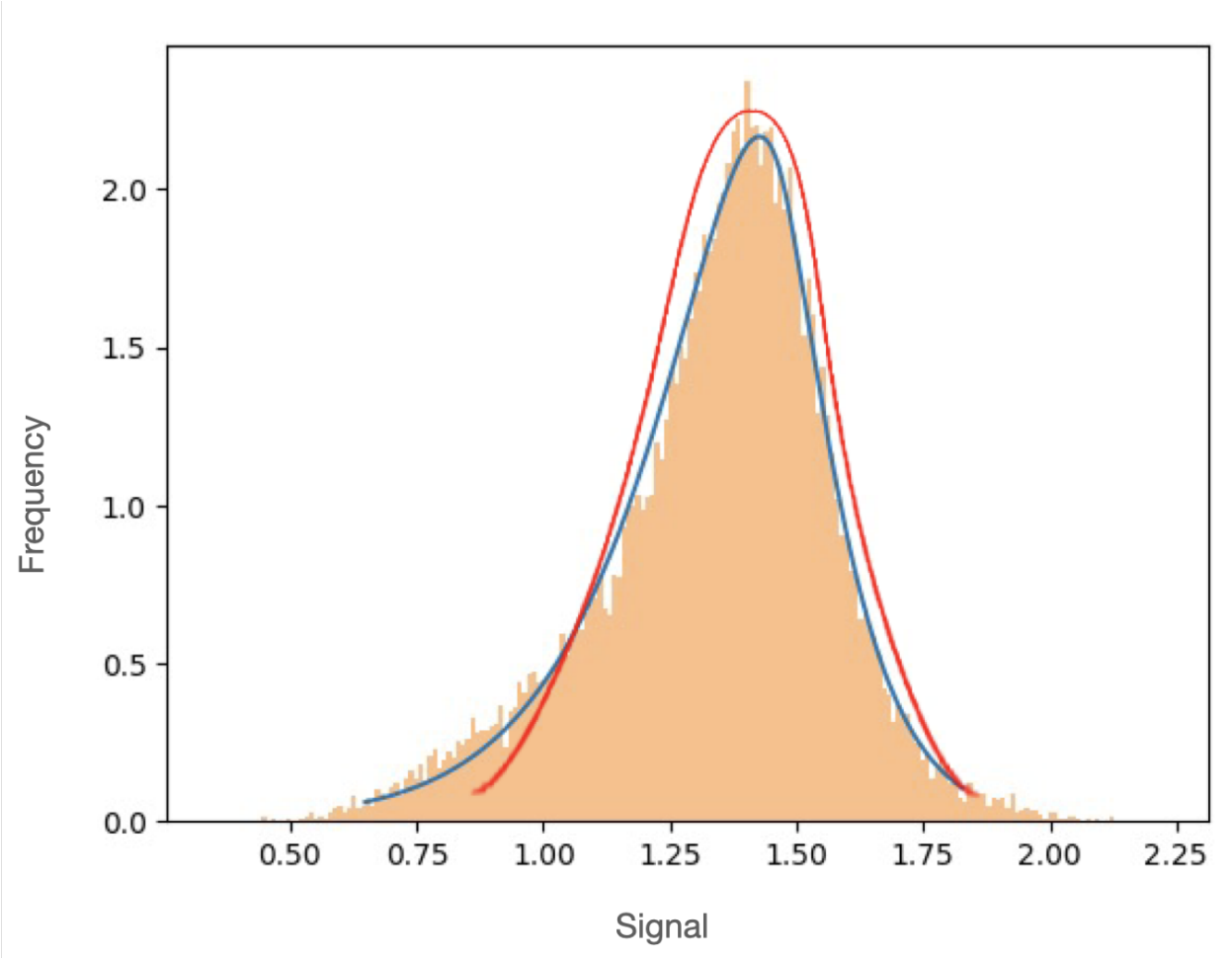


Figure 2.3: Comparison of distribution fits for open state noise. The Generalized Hyperbolic distribution, highlighted in blue, emerges as the best-fitting distribution compared to Gaussian distribution (in red), capturing the non-Gaussian characteristics of the noise with higher fidelity than the other distributions.

Parameter	Value
Number of states	7
Transition rate matrix ($\mathbf{R}$)	(see main text equations)
Current amplitude (A)	1 pA
Sampling rate ($1/\Delta t$)	10 kHz
Noise distribution	Generalized Hyperbolic
Open state GH parameters (λ, α, β, δ, μ)	$(1.5, 2.0, 0.1, 0.0, 0.2)$
Closed state GH parameters (λ, α, β, δ, μ)	$(1.0, 1.5, -0.2, 0.0, 0.1)$

Table 2.1: Summary of synthetic data generation parameters.

2.3 Classical Algorithms

In this section, we describe the classical clustering algorithms employed to classify ion channel states based on the observed data. Specifically, we will focus on two key approaches: **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** and a hybrid method combining **Bayesian Gaussian Mixture Models (BGMM)** with DBSCAN. These methods were selected for their ability to identify distinct clusters (corresponding to different ion channel states) in the data, while also handling noise and uncertainty that naturally arise in patch-clamp recordings [34].

DBSCAN is a well-known density-based algorithm that groups points in high-density regions and treats low-density regions as noise [35]. It is particularly suited for identifying channel states, as the density of data points may vary depending on the state of the ion channel and the noise present in the recordings.

The hybrid BGMM+DBSCAN method leverages the probabilistic modeling capabilities of Bayesian Gaussian Mixture Models to account for uncertainties in the data. By modeling the underlying distribution of the data as a mixture of Gaussian components, BGMM provides a more flexible approach to clustering [36]. Following the initial clustering by BGMM, DBSCAN is applied to further refine the clusters and remove outliers, resulting in a robust classification of ion channel states.

In the following subsections, we will provide a detailed explanation of how each algorithm works, the parameters used, and their application to the problem of ion channel state classification.

2.3.1 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a clustering algorithm that groups points based on the density of their surrounding data points [37]. It is well-suited for tasks involving noisy datasets, as it is designed to identify clusters of varying shapes and sizes while discarding points that do not belong to any cluster as noise [38]. In the context of ion channel state classification, DBSCAN helps to identify distinct clusters representing different channel states, while ignoring data points that correspond to noise or transitions between states.

The DBSCAN algorithm works by exploring the local neighborhood of each data point in the dataset and determining whether it belongs to a cluster [39]. The algorithm defines two key concepts: core points and border points. Core points are points that have a minimum number of neighboring points within a specified radius, while border points are those that are not core points themselves but are within the neighborhood of a core point.

The algorithm consists of the following steps:

1. For each point x_i, the algorithm calculates its ϵ-neighborhood, which is the set of all points x_j such that the distance between x_i and x_j satisfies:

$$\|x_i - x_j\| \leq \epsilon \tag{2.6}$$

   where $\|\cdot\|$ represents a distance metric, typically Euclidean distance, and ϵ is a user-defined parameter that specifies the radius of the neighborhood around each point [40].

2. If the number of points in the ϵ-neighborhood of x_i is greater than or equal to a minimum number of points, denoted as $n_{\min}$, then x_i is classified as a core point and a new cluster is formed, including all points within its ϵ-neighborhood.

3. Border points, which are within the ϵ-neighborhood of a core point but do not have enough neighboring points to qualify as core points, are added to the cluster.

4. Any points that are neither core points nor border points are classified as noise and are not assigned to any cluster.

2.3.1.1 Parameters of DBSCAN

DBSCAN requires two primary parameters to be defined:

- **Epsilon (ϵ):** This parameter defines the radius of the neighborhood around each point. A larger ϵ value results in larger clusters, while a smaller ϵ may lead to smaller clusters or an increase in the number of noise points. The choice of ϵ can significantly impact the results, and it is often determined experimentally or through heuristics such as the k-distance plot [41].
- **Minimum Points ($n_{\min}$):** This parameter specifies the minimum number of points that must exist within the ϵ-neighborhood for a point to be considered a core point. A higher $n_{\min}$ value can result in fewer clusters, as fewer points will qualify as core points [42].

2.3.1.2 Distance Metric

DBSCAN requires a distance metric to define the proximity of points in the dataset. In this study, we employed the Euclidean distance, defined as:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{2.7}$$

where $\mathbf{x}_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$ and $\mathbf{x}_j = \{x_{j1}, x_{j2}, ..., x_{jn}\}$ are n-dimensional vectors being compared, and n is the dimensionality of the data. This distance metric was appropriate for our dataset as it reflects the typical differences in conductance levels between states.

2.3.1.3 DBSCAN's Application to Ion Channel Data

When applied to our dataset, DBSCAN grouped points corresponding to distinct ion channel states into clusters, based on the density of data points. Points that did not belong to any clusters, typically corresponding to noise or transitions between states, were classified as outliers. This allowed us to focus on the most meaningful parts of the data while ignoring artifacts introduced by noise.

The choice of ϵ and $n_{\min}$ was crucial to the success of the algorithm. A small ϵ could result in over-segmentation, with too many small clusters and a large number of points classified as noise.

Conversely, a large ϵ could lead to under-segmentation, merging distinct states into a single cluster [43]. The parameter $n_{\min}$ also played a significant role in determining the granularity of the clusters. A low value of $n_{\min}$ could cause noise points to be incorrectly classified as part of a cluster, while a high value might result in missing smaller but significant clusters [44].

The final set of clusters identified by DBSCAN corresponded to different states of the ion channel, which were further analyzed to determine the number of open and closed states in the dataset.

2.3.2 Bayesian Gaussian Mixture Model (BGMM) and DBSCAN

Our proposed method for analyzing ion channel data consists of several key steps: peak detection, clustering, state prediction, and accuracy evaluation. We begin with a signal generated based on the model developed in section 2.1. An example of such signals can be seen in Figure 2.4

2.3.2.1 Peak Detection

Let $c \in \mathbb{R}^n$ be the observed current signal. We partition the range of c into m bins, defined by the set of edges $\{b_1, b_2, ..., b_{m+1}\}$ where $b_1 = \min(c)$ and $b_{m+1} = \max(c)$. The histogram of c is then computed as:

$$h_i = |\{c_j : b_i \leq c_j < b_{i+1}, j = 1, ..., n\}|, \quad i = 1, ..., m \tag{2.8}$$

We identify peaks in this histogram as local maxima. Let P be the set of peak indices:

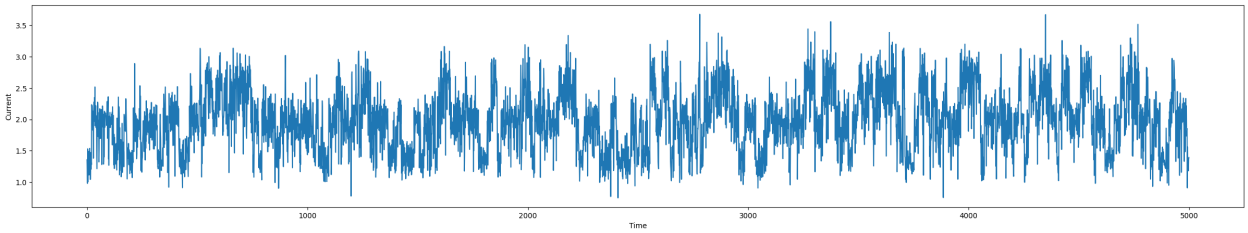
$$P = \{i : h_i > h_{i-1} \text{ and } h_i > h_{i+1}, i = 2, ..., m-1\} \tag{2.9}$$

To filter out insignificant peaks, we apply a threshold τ:

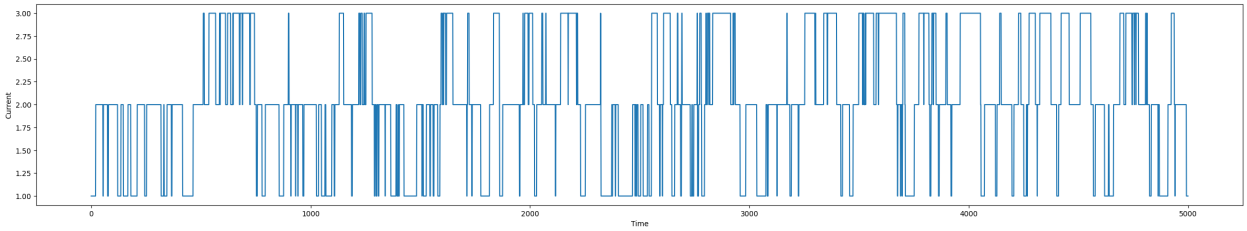
$$\tau = 0.05 \cdot \frac{1}{m} \sum_{i=1}^{m} h_i \tag{2.10}$$

The set of significant peaks P_s is then:

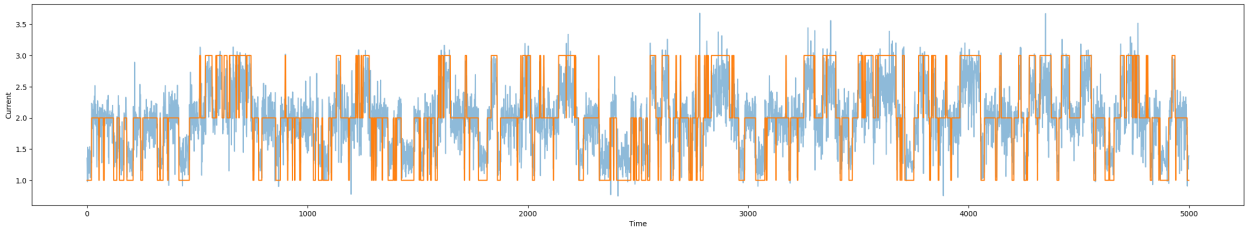
$$P_s = \{i \in P : h_i > \tau\} \tag{2.11}$$



(a) The signal current



(b) The signal state



(c) The signal current and its corresponding state superimposed

Figure 2.4: A sample signal derived from the model described in 2.1. $c(t)$ over time.

We represent each significant peak as a 2D coordinate:

$$\mathbf{x}_i = \left(\frac{b_i + b_{i+1}}{2}, h_i \right), \quad i \in P_s \tag{2.12}$$

A visualization of these steps can be seen in Figure 2.5

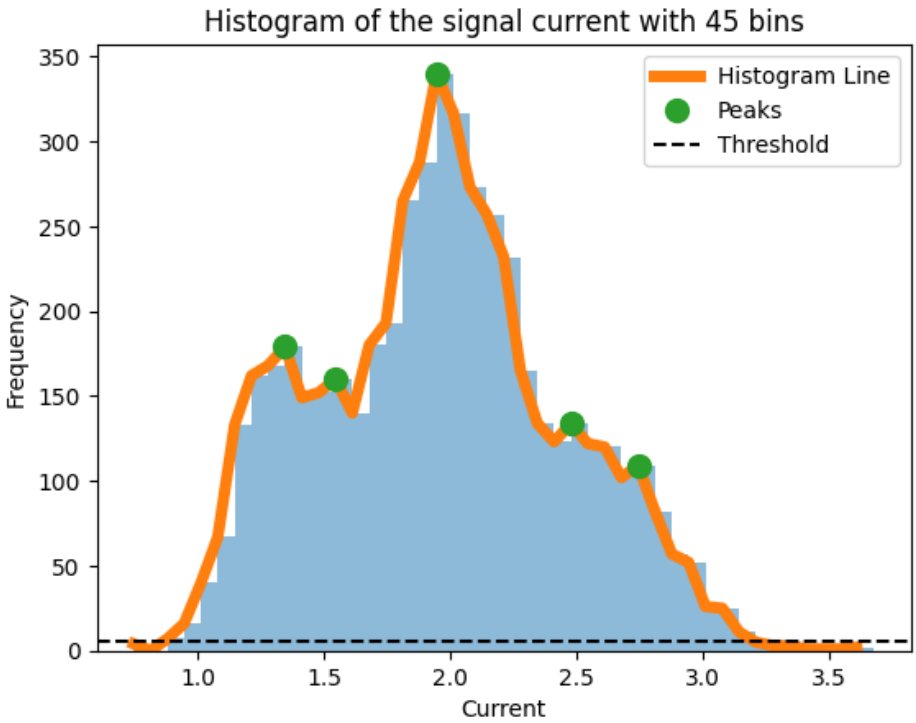


Figure 2.5: Histogram of the current values with m bins. Significant peaks (green dots) after applying the threshold τ, with their corresponding coordinates (x_i, h_i). The dashed line represents the threshold level.

2.3.2.2 Clustering

To identify distinct states, we apply DBSCAN clustering to the set of peak coordinates $\{\mathbf{x}_i : i \in P_s\}$. DBSCAN is defined by two parameters: ϵ (the maximum distance between two samples for one to be considered as in the neighborhood of the other) and $n_{\min}$ (the number of samples in a neighborhood for a point to be considered as a core point) [39].

Let $d(\mathbf{x}, \mathbf{y})$ be the Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$. We define:

$$N_\epsilon(\mathbf{x}) = \{\mathbf{y} : d(\mathbf{x}, \mathbf{y}) \leq \epsilon\} \tag{2.13}$$

A point $\mathbf{x}$ is a core point if $|N_\epsilon(\mathbf{x})| \geq n_{\min}$. DBSCAN then constructs clusters by connecting core points that are close to each other, and assigning non-core points to the clusters of their neighboring core points.

The output of DBSCAN is a set of cluster labels $L = \{l_1, ..., l_{|P_s|}\}$, where l_i is the cluster label assigned to peak $\mathbf{x}_i$.

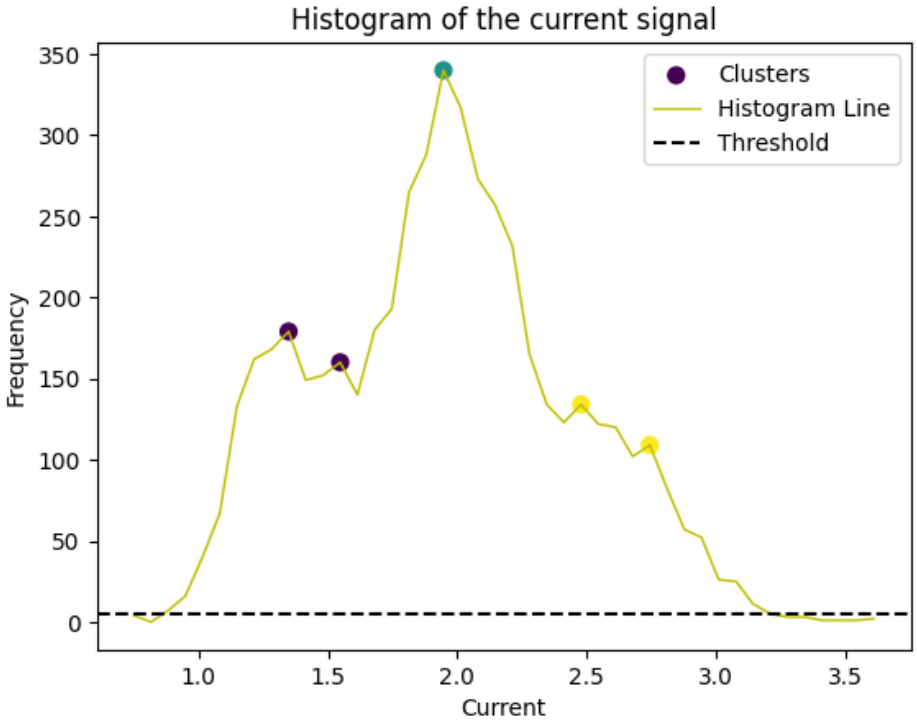


Figure 2.6: Results of DBSCAN clustering on the significant peaks. Different colors indicate distinct clusters identified by DBSCAN. The x-axis represents the current values, and the y-axis represents the histogram heights. Peaks belonging to the same cluster are likely to correspond to the same ion channel state.

2.3.2.3 State Prediction

To predict the state of each sample in the original signal c, we employ a Bayesian Gaussian Mixture (BGM) model [45]. Let K be the number of clusters identified by DBSCAN. The BGM model assumes that c is generated from a mixture of K Gaussian distributions:

$$p(c) = \sum_{k=1}^{K} \pi_k \mathcal{N}(c|\mu_k, \sigma_k^2) \tag{2.14}$$

where π_k, μ_k, and σ_k^2 are the mixing coefficient, mean, and variance of the k-th Gaussian component, respectively.

The BGM model is fitted to c using the Expectation-Maximization algorithm, which iteratively estimates the parameters $\{\pi_k, \mu_k, \sigma_k^2\}_{k=1}^K$ and the posterior probabilities of each sample belonging to each component [46].

After fitting, we predict the state s_j for each sample c_j as:

$$s_j = \arg\max_k p(k|c_j), \quad j = 1, ..., n \tag{2.15}$$

where s_j represents the predicted state at time j, corresponding to the number of open ion channels at that time instance. Thus, s_j can take on values from 0 (all channels closed) to N (all N channels open) where N is the total number of channels present in the patch. Also, $p(k|c_j)$ is the posterior probability of c_j belonging to the k-th component.

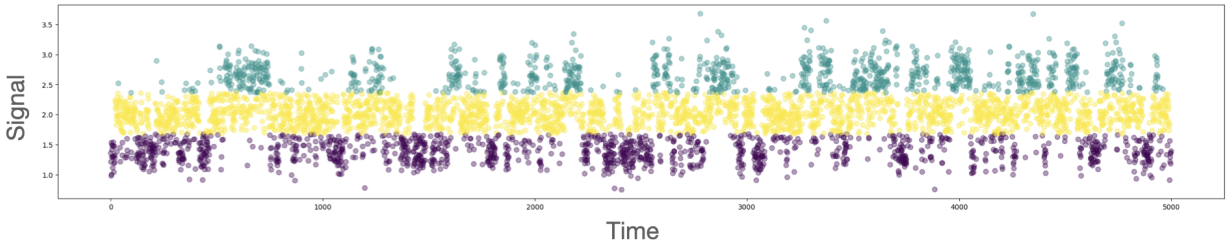


Figure 2.7: State classification of signal using Bayesian Gaussian Mixture Model (BGMM). Different colors represent distinct states, with each color corresponding to a Gaussian component in the mixture model. The BGMM classification provides a probabilistic assignment of each data point to a state.

2.3.2.4 Accuracy Evaluation

To evaluate the accuracy of our predictions, we compare them to the true states $s^* \in \{1, ..., K\}^n$ generated by our physical model. However, the predicted states s and true states s^* may use different labeling schemes. To address this, we find an optimal mapping between the two labelings.

Let $U = \{u_1, ..., u_M\}$ and $V = \{v_1, ..., v_N\}$ be the unique values in s^* and s respectively. We construct a cost matrix $C \in \mathbb{R}^{M \times N}$ where:

$$C_{ij} = -\frac{|\{k : s_k^* = u_i \text{ and } s_k = v_j\}|}{n} \tag{2.16}$$

We then solve the linear assignment problem:

$$\min_{\phi} \sum_{i=1}^{M} C_{i,\phi(i)} \tag{2.17}$$

where ϕ is a bijection from $\{1, ..., M\}$ to $\{1, ..., N\}$.

The optimal mapping ϕ^* is found using the Hungarian algorithm. We use this to relabel the predicted states:

$$\tilde{s}_j = u_{\phi^{*-1}(k)} \text{ where } s_j = v_k, \quad j = 1, ..., n \tag{2.18}$$

Finally, we compute the accuracy as:

$$\text{Accuracy} = \frac{|\{j : s_j^* = \tilde{s}_j\}|}{n} \tag{2.19}$$

This methodology provides a comprehensive approach to analyzing ion channel data, from identifying distinct states to evaluating the accuracy of state predictions.

2.4 Machine Learning Algorithms

In this section, we describe the two machine learning models used for state detection and classification: the Cluster Count Neural Network (NN) and the Long Short-Term Memory (LSTM) model [47]. These models were employed to improve the accuracy and robustness of state identification, especially in the presence of noise and long-duration recordings. The Cluster Count NN is designed to predict the number of distinct states in the signal, while the LSTM model is used to capture temporal dependencies and classify states based on the time-series data [48]. Below, we provide detailed descriptions of each model's architecture, training process, and performance evaluation.

2.4.1 Cluster Count Neural Network (NN)

The Cluster Count Neural Network (NN) was designed to estimate the number of distinct ion channels present in the given patch clamp recordings. The input to the neural network is a histogram representation of the signal, which captures the distribution of conductance levels across the patch clamp data. Below, we describe the methodology, data generation process, feature extraction, and

the architecture of the neural network model, followed by the mathematical foundations behind each step.

2.4.1.1 Data Generation

The dataset used for training the Cluster Count NN was generated by simulating patch clamp recordings with different numbers of ion channels. Each patch clamp trace was created using n_{channels} randomly generated ion channels, where n_{channels} is an integer sampled from the range 1 to 5:

$$n_{\text{channels}} \sim \text{Uniform}(1, 5).$$

As explained in 2.1, Each ion channel was modeled as a Markov process with 2 possible states, open and closed, resulting in a binary state trajectory for each channel. The overall conductance $X(t)$ at time t was the sum of the conductance contributions from all active channels at that time:

$$X(t) = \sum_{i=1}^{n_{\text{channels}}} g_i(t),$$

where $g_i(t)$ represents the conductance of channel i at time t. Each patch clamp recording was normalized using min-max scaling:

$$\hat{X}(t) = \frac{X(t) - \min(X)}{\max(X) - \min(X)},$$

ensuring that the values lay between 0 and 1.

A total of 100,000 patch clamp recordings were generated for training, with corresponding labels representing the number of ion channels present in each recording. The label y_i was assigned as the number of channels for the i-th recording, i.e.,

$$y_i = n_{\text{channels}}.$$

2.4.1.2 Feature Extraction

In order to provide a compact representation of the time-series data, each normalized patch clamp recording was transformed into a histogram of conductance levels. We partitioned the range $[0, 1]$

into $n_{\text{bins}} = 16$ equally spaced bins, and computed the histogram $H(X)$ for each signal $\hat{X}(t)$ as follows:

$$H(X)_j = \int_{X_j}^{X_{j+1}} P(\hat{X}(t))dt,$$

where $P(\hat{X}(t))$ is the probability density function of the signal, and X_j represents the edges of the j-th bin. The resulting feature vector $\mathbf{H}$, of length 16, was used as the input to the neural network.

2.4.1.3 Neural Network Architecture

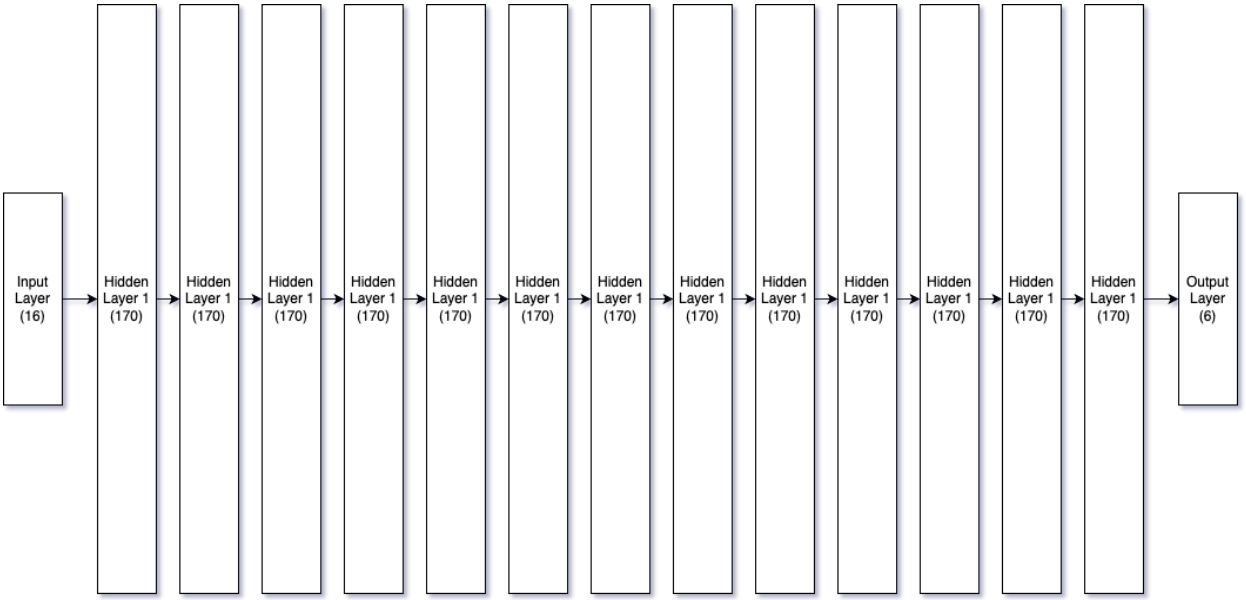


Figure 2.8: Architecture of the neural network model for ion channel number estimation. The model consists of an input layer accepting a vector of length 16, followed by 13 hidden layers, each with 170 neurons and ReLU activation functions. The output layer has 6 neurons with a softmax activation, corresponding to the probability distribution over possible ion channel numbers. Arrows represent the flow of information between layers. This deep architecture allows for complex feature extraction and transformation of the input signal, enabling accurate classification of ion channel states.

The Cluster Count NN was implemented as a fully connected feedforward neural network with multiple hidden layers. The input to the network was the histogram feature vector $\mathbf{H} \in \mathbb{R}^{16}$, representing the conductance distribution of the signal. The network was tasked with predicting the number of ion channels present, which can take integer values between 1 and 5.

The architecture consisted of an input layer, followed by 13 hidden layers, each containing 170

neurons with rectified linear unit (ReLU) activations:

$$\text{ReLU}(x) = \max(0, x).$$

The output layer consisted of 6 neurons (for 6 possible output categories, including the 'no channel' case), with a softmax activation function:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{6} e^{z_j}},$$

which converted the network's output into a probability distribution over the possible channel counts.

The final model structure can be summarized as follows:

- Input: 16-dimensional histogram vector.
- 13 hidden layers, each with 170 neurons and ReLU activations.
- Output: 6 neurons with softmax activation.

The model was trained using categorical cross-entropy loss:

$$L = -\sum_{i=1}^{6} y_i \log(\hat{y}_i),$$

where y_i is the true label (one-hot encoded) and $\hat{y}_i$ is the predicted probability for class i. The optimizer used for training was Adam, with learning rate $\eta = 0.001$, which updates the model's parameters as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L,$$

where θ represents the model parameters (weights and biases).

2.4.1.4 Training and Evaluation

The dataset was split into training and test sets using an 80/20 ratio. The categorical labels y were one-hot encoded to produce binary vectors for each class:

$$y = \text{one\_hot}(n_{\text{channels}}).$$

The model was trained for 50 epochs, with a batch size of 32, and performance was evaluated using accuracy on the test set.

The architecture's depth, breadth, and other hyperparameters were carefully tuned and the best values were selected, as presented in the results section [49].

2.4.2 Long Short-Term Memory (LSTM) Model

The Long Short-Term Memory (LSTM) model was developed to further enhance the prediction of ion channel counts by incorporating temporal dependencies within the patch clamp signals [50]. The LSTM network leverages sequential information in the time-series data, enabling it to capture the dynamic behavior of ion channels over time. Additionally, this model integrates the output of the Cluster Count Neural Network (CCNN) as an input to improve the accuracy of the prediction. Below, we describe the LSTM architecture, data preparation process, and the mathematical foundations behind the model.

2.4.2.1 Data Generation

To train the LSTM, we generated synthetic patch clamp data similar to the previous Cluster Count NN model. However, for the LSTM model, we utilized a more detailed sequence of data points, capturing the temporal behavior of ion channels. Specifically, for each sample, $N_{\text{POINTS}} = 10$ data points before and after a random midpoint in the signal were taken to form the time-series input for the LSTM.

For each patch clamp recording, n_{channels} random ion channels were simulated, with the total signal $X(t)$ representing the sum of the conductances of active channels at time t. The resulting

patch clamp signal was represented as a sequence of $2 \cdot N_{\text{POINTS}} + 1$ time points:

$$\mathbf{X} = [X(t - N_{\text{POINTS}}), \ldots, X(t), \ldots, X(t + N_{\text{POINTS}})].$$

This sequence served as the input to the LSTM model. The midpoint of the signal, $X(t)$, was used to derive the categorical label y_i, representing the ion channel state at that time.

2.4.2.2 Feature Extraction and Data Preparation

The LSTM model utilized both the raw time-series data $\mathbf{X}$ and the predicted ion channel count from the Cluster Count NN as inputs. For each sample, the ion channel count n_{channels} from the CCNN was concatenated with the LSTM output to enhance prediction accuracy.

The dataset consisted of:

- $\mathbf{X}$: A sequence of $2 \cdot N_{\text{POINTS}} + 1$ time points representing the patch clamp signal.
- n_{channels}: The ion channel count predicted by the Cluster Count NN.
- y: A one-hot encoded vector representing the ion channel state at the midpoint of the signal.

The input $\mathbf{X}$ was reshaped as a 3-dimensional tensor with shape $(n_{\text{samples}}, 2 \cdot N_{\text{POINTS}} + 1, 1)$, representing the number of samples, time points, and signal dimensions, respectively. The ion channel count n_{channels} was reshaped to have shape $(n_{\text{samples}}, 1)$.

2.4.2.3 LSTM Model Architecture

The LSTM model consists of two main components:

1. A Long Short-Term Memory (LSTM) layer, which processes the time-series data $\mathbf{X}$.

2. A dense layer that concatenates the output of the LSTM layer with the predicted ion channel count from the CCNN, followed by a softmax layer for classification.

The architecture of the LSTM model is as follows:

- **Input 1**: The predicted ion channel count n_{channels} from the Cluster Count NN, with shape (1).
- **Input 2**: The time-series signal $\mathbf{X}$, with shape $(2 \cdot N_{\text{POINTS}} + 1, 1)$.

- **LSTM Layer**: A single LSTM layer with $n_{\text{lstm}} = 64$ units, processing the time-series input. The LSTM computes a hidden state $\mathbf{h}_t$ at each time step based on the previous hidden state and the input at that time [51]:

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_h \cdot \mathbf{X}_t + \mathbf{U}_h \cdot \mathbf{h}_{t-1} + \mathbf{b}_h\right),$$

where $\mathbf{W}_h$, $\mathbf{U}_h$, and $\mathbf{b}_h$ are the weight matrices and bias vector, and σ is the activation function (typically tanh or ReLU).

The LSTM layer maintains both a hidden state $\mathbf{h}_t$ and a cell state $\mathbf{c}_t$, allowing it to retain long-term dependencies:

$$\mathbf{c}_t = f_t \cdot \mathbf{c}_{t-1} + i_t \cdot \tilde{\mathbf{c}}_t,$$

where f_t, i_t, and $\tilde{\mathbf{c}}_t$ are the forget, input, and candidate cell state activations, respectively.

- **Concatenation Layer**: The output of the LSTM layer $\mathbf{h}_t$ is concatenated with the predicted ion channel count n_{channels} from the Cluster Count NN:

$$\mathbf{z} = \text{concat}([\mathbf{h}_t, n_{\text{channels}}]),$$

forming a combined feature vector $\mathbf{z}$.

- **Dense Layer**: The concatenated feature vector $\mathbf{z}$ is passed through a fully connected dense layer with 50 neurons and ReLU activation:

$$\mathbf{z}' = \text{ReLU}(\mathbf{W}_d \cdot \mathbf{z} + \mathbf{b}_d)$$

where $\mathbf{W}_d$ and $\mathbf{b}_d$ are the weight matrix and bias vector for the dense layer.

- **Output Layer**: A final softmax layer with 10 neurons is used for classification, predicting the ion channel state at the midpoint of the signal. The softmax function converts the raw scores into probabilities:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}$$

where z_i represents the output score for class i.

The model was trained using categorical cross-entropy loss:

$$L = -\sum_{i=1}^{10} y_i \log(\hat{y}_i),$$

where y_i is the true class label (one-hot encoded), and $\hat{y}_i$ is the predicted probability for class i. The Adam optimizer was used to update the model's weights:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L,$$

where θ represents the model's parameters, and η is the learning rate. The chosen learning rate was $\eta = 0.001$ [52].

2.4.2.4 Training and Evaluation

The dataset was split into training and test sets using an 80/20 ratio. The model was trained for 200 epochs, with a batch size of 32, and performance was evaluated using accuracy on the validation set. The architecture's depth, number of LSTM units, and other hyperparameters were carefully assessed in the results section to select the best performing values.

Chapter 3

Results

In this chapter, we first present a detailed analysis of the results obtained from classical algorithms, examining their respective strengths and limitations. Following this, we evaluate the performance of deep learning-based algorithms, offering a similar critical assessment. Ultimately, we provide a comprehensive head-to-head comparison of these approaches, initially using synthetic data and subsequently validating their performance on real-world patch clamp datasets.

3.1 Classical Algorithms

Our initial investigation centered on classical clustering algorithms, which group data points based on similarity in an unsupervised manner. We explored two primary approaches: Density-Based Spatial Clustering of Applications with Noise (DBSCAN), a density-based algorithm renowned for its capability to identify clusters of arbitrary shapes [34], and a hybrid method combining DBSCAN with Bayesian Gaussian Mixture Modeling (BGMM), designed to leverage BGMM's probabilistic framework to address certain limitations of DBSCAN [53].

3.1.1 DBSCAN

As outlined in Section 2.3.1, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that operates based on the density of data points. The core idea behind DBSCAN is to identify dense regions of data points, forming clusters, while classifying regions of low density as noise. This density-based approach makes DBSCAN particularly effective for discovering clusters of arbitrary shapes, in contrast to algorithms such as k-means, which assume spherical clusters. Additionally, DBSCAN does not require the number of clusters to be specified in advance, making it a flexible tool for exploratory data analysis.

The DBSCAN algorithm relies on two key parameters: ϵ and $n_{\min}$. The ϵ parameter (which we defined and discussed in detail in Section 2.3.1) represents the maximum distance between two points for them to be considered part of the same neighborhood. Essentially, ϵ controls the size of the neighborhoods, which, in turn, determines how clusters are formed. The $n_{\min}$ parameter defines the minimum number of points required to form a dense region, thus identifying the core points around which clusters are built. Points within the ϵ-neighborhood of a core point are classified as part of the same cluster, while points that do not meet this criterion are classified as noise or

outliers.

For our study, we applied DBSCAN to a synthetic dataset designed to simulate ion channel recordings. The generation of this dataset was based on the model described in Section 2.1, where we created synthetic data representing ion channel states, such as open and closed conformations.

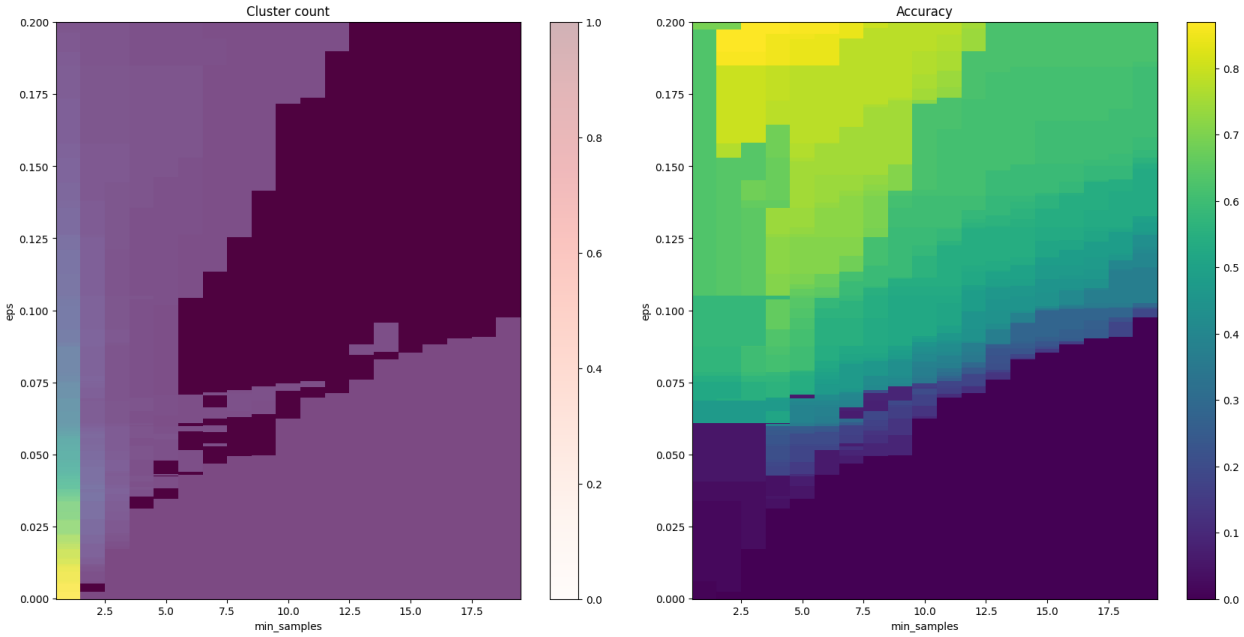
3.1.1.1 Parameter Sensitivity and Ion Count Analysis

DBSCAN operates with two key parameters: ϵ (the neighborhood radius) and $n_{\min}$ (the minimum number of points required to form a cluster) [34]. Additionally, the complexity of the ion channel signals varies with the number of ion channels present, ranging from one to six ion channels in our synthetic datasets. As the number of ion channels increases, the noise level in the signal also rises due to each ion introducing its own unique noise distribution. This escalation in noise presents a challenge for clustering algorithms, making it essential to evaluate how changes in DBSCAN's parameters impact its ability to accurately detect the number of ion channels.

To thoroughly assess the performance of DBSCAN, we systematically varied the parameters ϵ and $n_{\min}$ while fixing the number of ion channels for each iteration. We began with a signal containing one ion channel and incrementally increased the number of ion channels in subsequent tests. By doing so, we aimed to observe how parameter adjustments influence the algorithm's accuracy in identifying the correct number of clusters (corresponding to the number of ion channels).

In the one-ion channel scenario, we present the 2D colormap representation (Figure 3.1a), which shows the relationship between ϵ, $n_{\min}$, and the number of detected clusters. This visualization provides insight into the parameter regions where DBSCAN accurately detects the correct number of clusters. In this case, dark violet areas indicate where the correct number of clusters is identified. Alongside this, we introduce an accuracy plot (Figure 3.2b), showcasing the impact of the same parameters on state classification accuracy. By comparing these two plots side by side, we can assess how well cluster detection aligns with classification performance.

Next, we present two plots that examine the relationships between DBSCAN's parameters, ϵ and $n_{\min}$, and the number of detected clusters for a signal containing one ion. The first plot (Figure 3.2a) explores the impact of varying ϵ while holding $n_{\min}$ constant. This visualization clearly demonstrates how even small changes in ϵ significantly influence DBSCAN's ability to correctly identify the number of clusters, with ϵ playing a dominant role in determining the algorithm's



(a) 2D colormap of the number of detected clusters as a function of ϵ and $n_{\min}$ for a signal containing one ion.

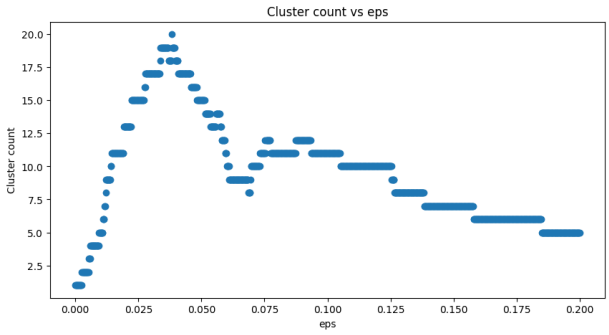
(b) Accuracy as a function of ϵ and $n_{\min}$ for a signal containing one ion. The highest achieved accuracy is 87%.

Figure 3.1: Side-by-side comparison of cluster detection and state classification accuracy for the one-ion scenario. This illustrates that while DBSCAN can correctly identify the number of clusters in certain parameter regions, this does not always result in optimal classification accuracy.

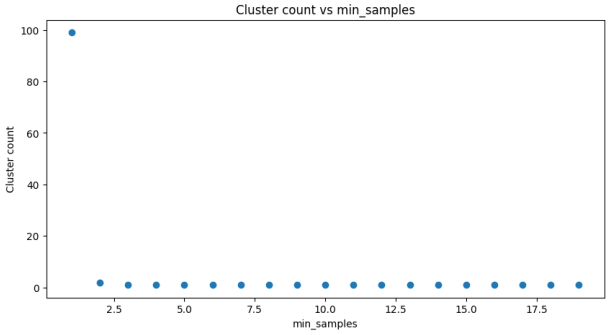
clustering performance.

In contrast, the second plot (Figure 3.2b) examines the effect of changing $n_{\min}$ while keeping ϵ constant. The number of detected clusters remains largely unaffected by variations in $n_{\min}$, indicating that this parameter has minimal influence on cluster detection compared to ϵ [54]. This result can be explained by the nature of DBSCAN, which is primarily designed for spatial datasets rather than time series data. While $n_{\min}$ is critical for defining dense regions in spatial clustering, its role diminishes in the time series context.

This analysis highlights the importance of ϵ over $n_{\min}$ in this specific application of DBSCAN, emphasizing that while DBSCAN is not naturally designed for time series datasets, it can still be adapted with proper transformation techniques, albeit with some limitations on the influence of certain parameters.



(a) The number of detected clusters as a function of ϵ for a signal containing one ion, with $n_{\min}$ held constant. This plot demonstrates how the ϵ parameter impacts cluster detection.



(b) The number of detected clusters as a function of $n_{\min}$ for a signal containing one ion, with ϵ held constant. This plot demonstrates how the ϵ parameter impacts cluster detection.

Figure 3.2: Comparison of DBSCAN cluster detection for a one-ion signal as a function of ϵ and $n_{\min}$. (Left) The number of detected clusters as a function of ϵ, with $n_{\min}$ held constant, demonstrating that changes in ϵ have a significant impact on cluster detection. (Right) The number of detected clusters as a function of $n_{\min}$, with ϵ held constant, showing that variations in $n_{\min}$ have minimal effect on cluster count compared to ϵ. Together, these plots highlight the dominant role of ϵ in determining DBSCAN's clustering performance.

In summary, the one-ion channel analysis reveals that identifying the correct number of clusters does not always correspond to achieving the highest accuracy in state classification as evident in Figure 3.2. This reinforces the importance of tuning DBSCAN's parameters to not only detect clusters but also optimize classification accuracy.

3.1.1.2 Analysis for Two Ion Channels

Continuing the analysis, we now extend the evaluation to a signal containing two ion channels. In this case, there are three possible states: both closed, one open, and both open. The presence of an additional ion introduces further complexity, as each ion channel contributes its own noise distribution. This makes it essential to assess how changes in DBSCAN's parameters, ϵ and $n_{\min}$, impact its ability to detect the correct number of ion channel states.

Similar to the one-ion scenario, we varied ϵ and $n_{\min}$ to observe their influence on cluster detection and state classification accuracy [55]. The 2D colormap (Figure 3.3a) shows the relationship between these parameters and the number of detected clusters for the two-ion signal. Dark violet regions indicate areas where DBSCAN correctly identifies the expected three clusters. To complement this, an accuracy plot (Figure 3.3b) illustrates the classification accuracy across the same parameter space.

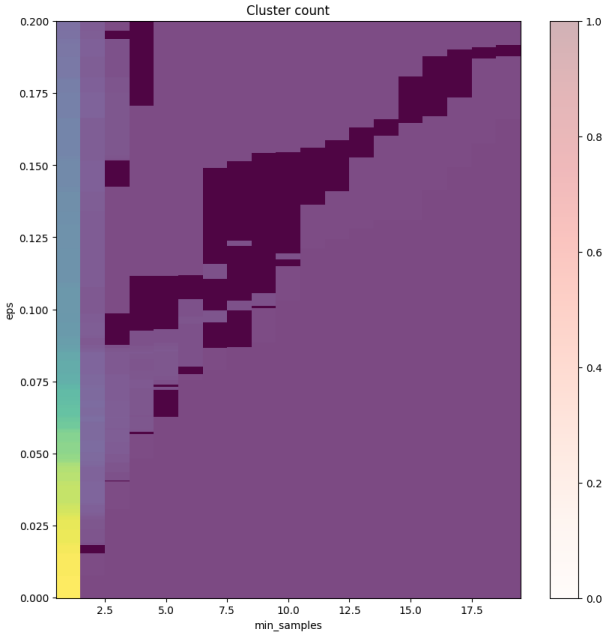
By comparing these two plots side by side, we can see that DBSCAN's ability to detect the correct number of clusters does not always align with optimal state classification performance, as was also observed in the one-ion channel case. However, there is a strong relationship between them.

To further illustrate the sensitivity of DBSCAN to ϵ and $n_{\min}$, Figure 3.4a shows how the number of detected clusters varies with ϵ when $n_{\min}$ is held constant, while Figure 3.4b shows the effect of varying $n_{\min}$ with a constant ϵ. These plots confirm that ϵ plays a far more significant role than $n_{\min}$ in determining the number of detected clusters, reinforcing the conclusions drawn from the one-ion case [56].

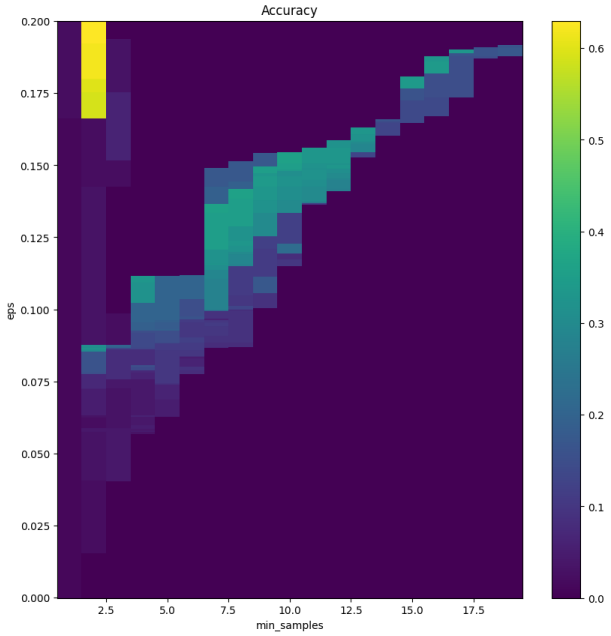
These results for the two-ion case further confirm the dominant influence of ϵ on DBSCAN's ability to correctly identify clusters. While DBSCAN can still be adapted for this application, the challenge of noise and overlapping states becomes more pronounced as the complexity of the signal increases, particularly when more than one ion is present.

3.1.1.3 Analysis for Three Ion Channels

Building on the analysis from the one- and two-ion channel scenarios, we now evaluate the performance of DBSCAN on a signal containing three ion channels. This case introduces four possible

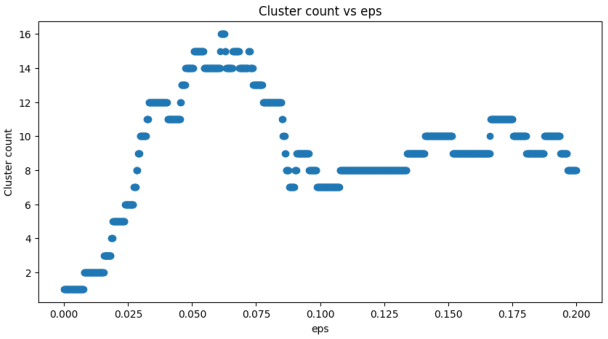


(a) 2D colormap of the number of detected clusters as a function of ϵ and $n_{\min}$ for a signal containing two ion channels.

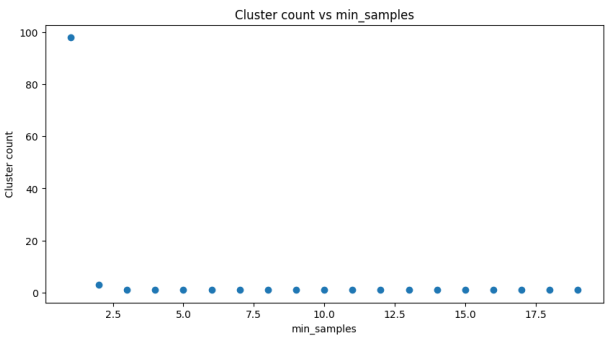


(b) Accuracy as a function of ϵ and $n_{\min}$ for a signal containing two ion channels. The highest achieved accuracy is 63%.

Figure 3.3: Side-by-side comparison of cluster detection and state classification accuracy for the two-ion scenario. As with the one-ion case, DBSCAN correctly identifies the number of clusters in some regions, but these do not always correspond to the highest classification accuracy.



(a) The number of detected clusters as a function of ϵ for a signal containing two ion channels, with $n_{\min}$ held constant.

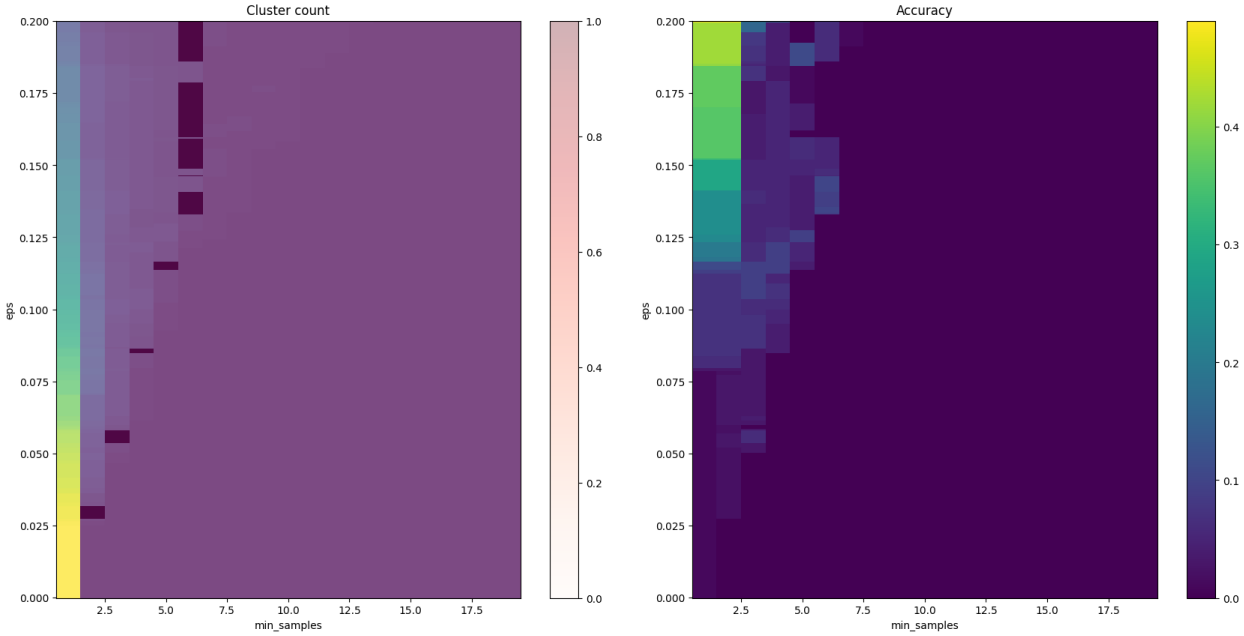


(b) The number of detected clusters as a function of $n_{\min}$ for a signal containing two ion channels, with ϵ held constant.

Figure 3.4: Comparison of DBSCAN cluster detection for a two-ion signal as a function of ϵ and $n_{\min}$. (Left) The number of detected clusters as a function of ϵ, with $n_{\min}$ held constant, showing ϵ's significant impact on cluster detection. (Right) The number of detected clusters as a function of $n_{\min}$, with ϵ held constant, demonstrating that $n_{\min}$ has minimal effect compared to ϵ.

states: all ion channels closed, one ion open, two ion channels open, and all ion channels open. With the increase in the number of ion channels, the noise becomes more complex, and DBSCAN's ability to accurately detect the correct number of states diminishes.

As seen in the previous cases, we varied ϵ and $n_{\min}$ to study their effects on cluster detection and classification accuracy [57]. Figure 3.5a presents the 2D colormap for the number of detected clusters across the parameter space. However, compared to the two-ion channel scenario, the number of parameter pairs that correctly identify the expected four clusters has reduced significantly, as indicated by the limited dark violet regions. Correspondingly, the classification accuracy drops to a maximum of 49% (Figure 3.5b), reflecting the increased difficulty in distinguishing between overlapping states in the presence of more ion channels.



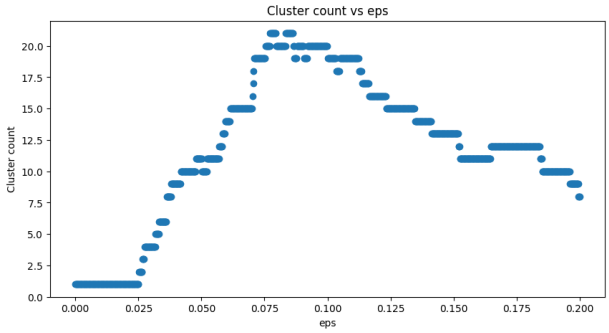
(a) 2D colormap of the number of detected clusters as a function of ϵ and $n_{\min}$ for a signal containing three ion channels.

(b) Accuracy as a function of ϵ and $n_{\min}$ for a signal containing three ion channels. The highest achieved accuracy is 49%.

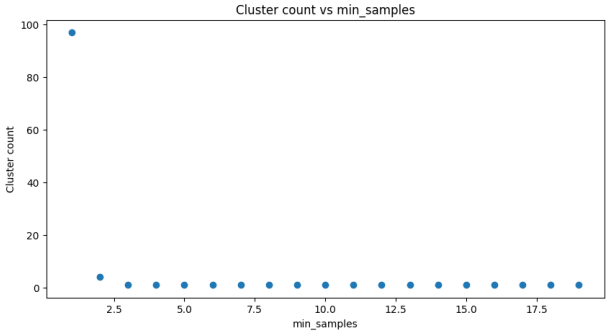
Figure 3.5: Side-by-side comparison of cluster detection and state classification accuracy for the three-ion scenario. As the complexity of the signal increases, the number of correct cluster detections and classification accuracy both decrease significantly.

In Figure 3.6a and Figure 3.6b, we further investigate how varying ϵ and $n_{\min}$ affects the number of detected clusters for this three-ion signal. As with the previous cases, ϵ plays a dominant role in determining the number of clusters detected [58]. However, it becomes evident that finding the

right balance of parameters becomes more challenging as the number of ion channels increases, with fewer parameter pairs yielding the correct number of clusters.



(a) The number of detected clusters as a function of ϵ for a signal containing three ion channels, with $n_{\min}$ held constant.



(b) The number of detected clusters as a function of $n_{\min}$ for a signal containing three ion channels, with ϵ held constant.

Figure 3.6: Comparison of DBSCAN cluster detection for a three-ion signal as a function of ϵ and $n_{\min}$. (Left) The number of detected clusters as a function of ϵ, with $n_{\min}$ held constant, showing ϵ's dominant role in cluster detection. (Right) The number of detected clusters as a function of $n_{\min}$, with ϵ held constant, showing a similar pattern to previous cases.

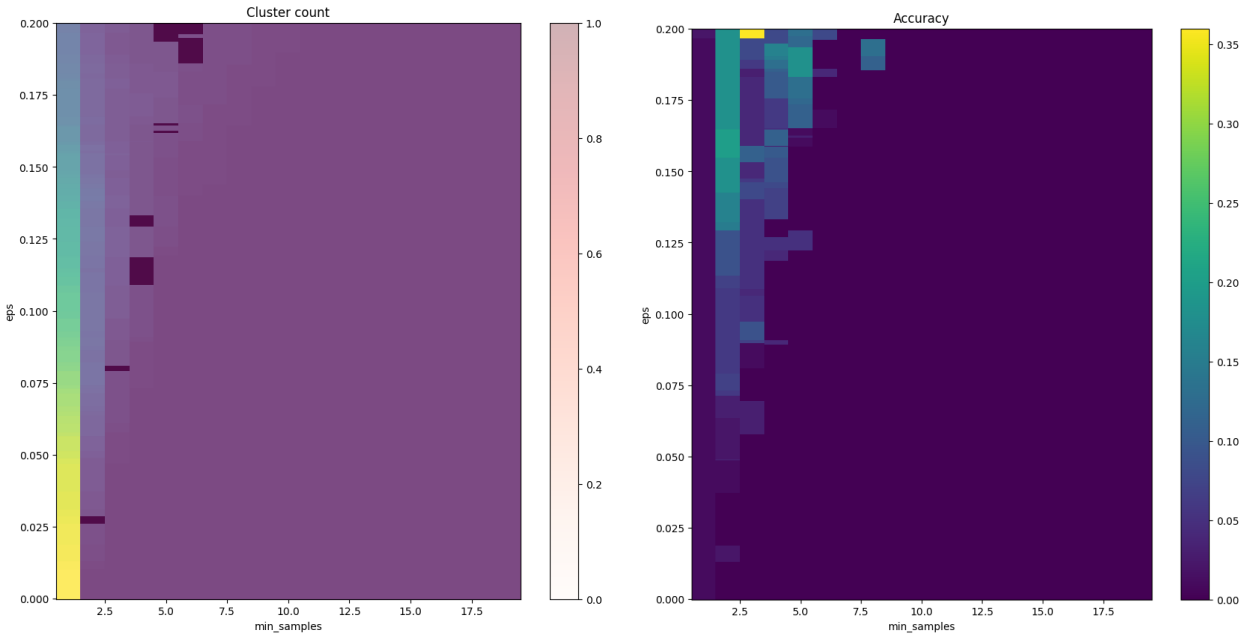
As observed in the two-ion case, the trend continues with three ion channels: both accuracy and the number of parameter pairs that correctly identify the number of clusters decrease as the number of ion channels increases. This suggests that DBSCAN's sensitivity to noise and overlapping states becomes more pronounced with increasing signal complexity, highlighting the limitations of the algorithm for larger systems of ion channels.

3.1.1.4 Analysis for Four Ion Channels

Continuing with the trend observed in the one-, two-, and three-ion cases, we now analyze the performance of DBSCAN on a signal containing four ion channels. This scenario introduces five possible states: all ion channels closed, one ion open, two ion channels open, three ion channels open, and all ion channels open. The increased complexity and overlap between states present even greater challenges for DBSCAN in accurately detecting the correct number of clusters.

As before, we varied ϵ and $n_{\min}$ to explore their impact on cluster detection and classification accuracy [59]. Figure 3.7a shows the 2D colormap for the number of detected clusters across the parameter space. Similar to the three-ion case, the number of parameter pairs that correctly identify the expected five clusters has reduced significantly, with fewer regions showing the desired

cluster count. The classification accuracy further decreases, with a maximum of 36% (Figure 3.7b), indicating the growing difficulty of distinguishing between states as the number of ion channels increases.



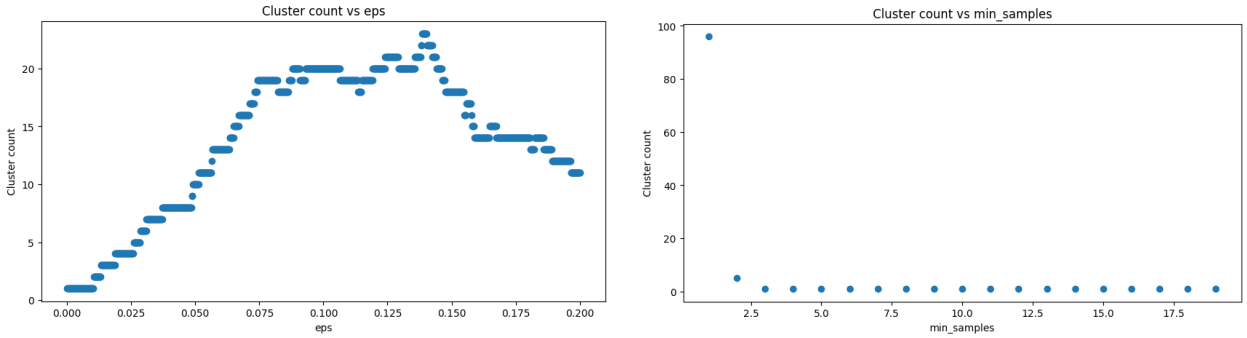
(a) 2D colormap of the number of detected clusters as a function of ϵ and $n_{\min}$ for a signal containing four ion channels.

(b) Accuracy as a function of ϵ and $n_{\min}$ for a signal containing four ion channels. The highest achieved accuracy is 36%.

Figure 3.7: Side-by-side comparison of cluster detection and state classification accuracy for the four-ion scenario. As the complexity of the signal increases, both the number of correct cluster detections and classification accuracy drop significantly.

In Figure 3.8a and Figure 3.8b, we further analyze how varying ϵ and $n_{\min}$ affects the number of detected clusters for the four-ion signal. As in the previous cases, ϵ continues to play a dominant role, but the trend of diminishing parameter pairs yielding correct cluster counts is even more pronounced in this four-ion scenario.

As expected, the trend continues: both the classification accuracy and the number of parameter pairs that correctly identify the number of clusters decrease with the four-ion case [60]. DBSCAN's sensitivity to overlapping states and noise becomes more apparent as the signal complexity increases, further illustrating the limitations of the algorithm when applied to larger systems of ion channels.



(a) The number of detected clusters as a function of ϵ for a signal containing four ion channels, with $n_{\min}$ held constant.

(b) The number of detected clusters as a function of $n_{\min}$ for a signal containing four ion channels, with ϵ held constant.

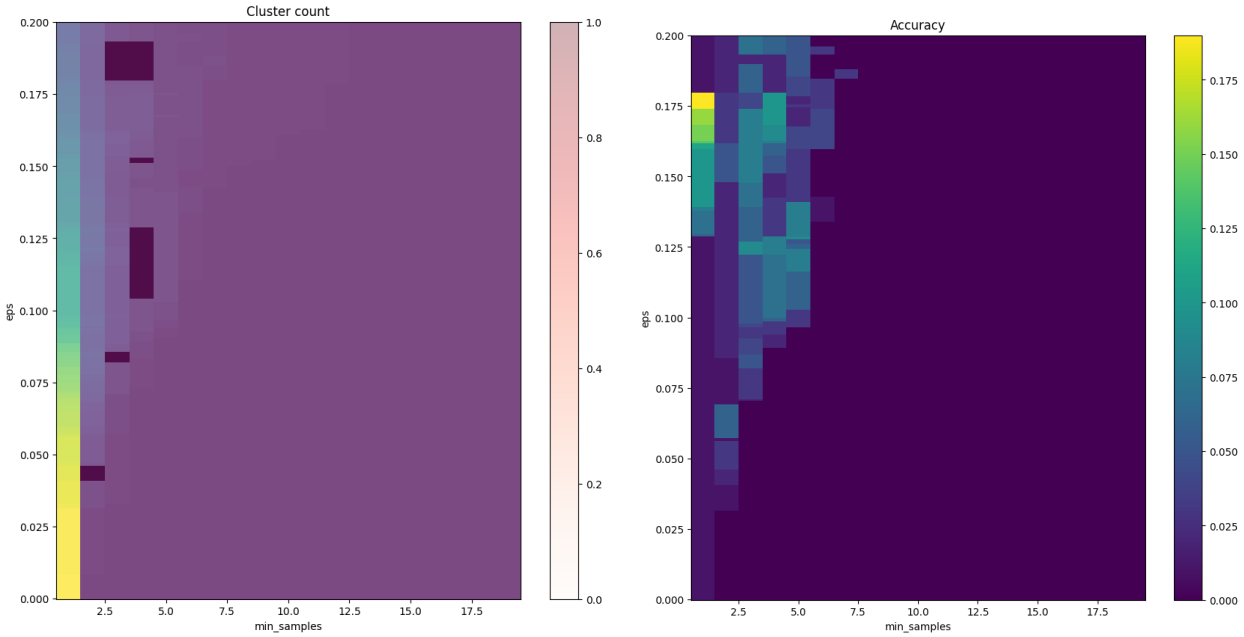
Figure 3.8: Comparison of DBSCAN cluster detection for a four-ion signal as a function of ϵ and $n_{\min}$. (Left) The number of detected clusters as a function of ϵ, with $n_{\min}$ held constant, showing ϵ's dominant role in cluster detection. (Right) The number of detected clusters as a function of $n_{\min}$, with ϵ held constant, showing a similar pattern to previous cases.

3.1.1.5 Analysis for Five Ion Channels

Continuing with the trend observed in the one-, two-, three-, and four-ion cases, we now analyze the performance of DBSCAN on a signal containing five ion channels. This scenario introduces six possible states: all ion channels closed, one ion open, two ion channels open, three ion channels open, four ion channels open, and all ion channels open. The increased complexity and overlap between states make it increasingly difficult for DBSCAN to accurately detect the correct number of clusters.

As before, we systematically varied ϵ and $n_{\min}$ to explore their impact on cluster detection and classification accuracy [59]. Figure 3.9a provides a 2D colormap visualization of the relationship between ϵ, $n_{\min}$, and the number of detected clusters for this five-ion signal. The dark violet regions indicate parameter pairs where DBSCAN correctly identifies the expected number of clusters (six). Compared to the four-ion scenario, the number of parameter pairs that result in accurate cluster detection is significantly reduced, continuing the trend as the ion count increases.

Figures 3.10a and 3.10b provide additional insight into the effects of ϵ and $n_{\min}$ on cluster detection. The first plot examines the impact of varying ϵ while holding $n_{\min}$ constant, whereas the second plot explores the effect of changing $n_{\min}$ while keeping ϵ fixed. As seen in previous cases, ϵ plays a more critical role in determining DBSCAN's performance, with $n_{\min}$ having minimal impact

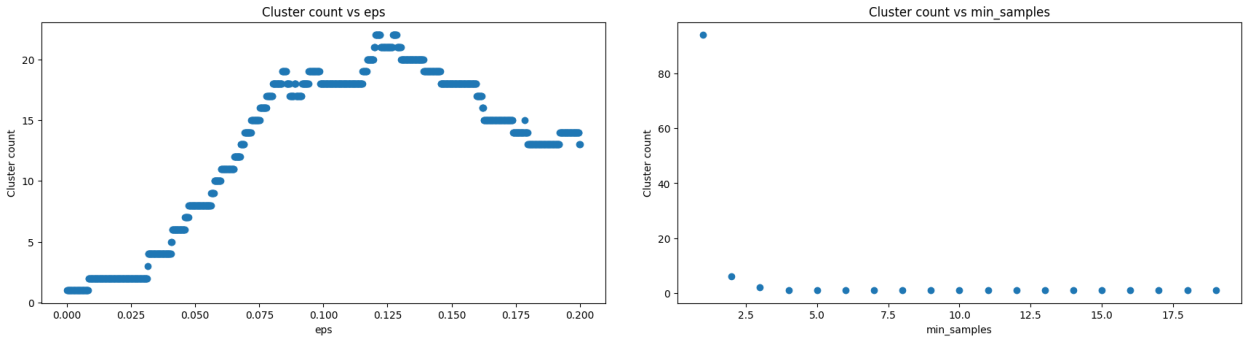


(a) 2D colormap of the number of detected clusters as a function of ϵ and $n_{\min}$ for a signal containing five ion channels.

(b) Accuracy as a function of ϵ and $n_{\min}$ for a signal containing five ion channels. The highest achieved accuracy is 19%.

Figure 3.9: Side-by-side comparison of cluster detection and state classification accuracy for the five-ion scenario. Compared to the four-ion case, the number of parameter pairs that correctly identify the number of clusters continues to decrease, and the classification accuracy drops to 19%.

on the number of detected clusters [39].



(a) The number of detected clusters as a function of ϵ for a signal containing five ion channels, with $n_{\min}$ held constant.

(b) The number of detected clusters as a function of $n_{\min}$ for a signal containing five ion channels, with ϵ held constant.

Figure 3.10: Comparison of DBSCAN cluster detection for a five-ion signal as a function of ϵ and $n_{\min}$. (Left) The number of detected clusters as a function of ϵ, with $n_{\min}$ held constant, demonstrating the dominant role of ϵ in cluster detection. (Right) The number of detected clusters as a function of $n_{\min}$, with ϵ held constant, showing minimal effect of $n_{\min}$ on cluster detection.

This analysis confirms that as the number of ion channels increases, the challenge of identifying the correct number of clusters and achieving high classification accuracy becomes progressively harder. For the five-ion signal, the highest accuracy reached is 19%, which is barely better than random guessing. Due to this sharp decline in performance, we conclude that further increasing the number of ion channels is not meaningful, as the model fails to handle five ion channels effectively. Additionally, the parameter regions that correctly identify the number of clusters continue to shrink, further reinforcing the trend seen in the previous cases.

3.1.1.6 Conclusion

This analysis of the DBSCAN algorithm across multiple ion signals reveals both its strengths and limitations. As demonstrated in Fig. 3.11, the accuracy of DBSCAN steadily declines as the number of ion channels increases, with a significant drop in performance for signals containing four and five ion channels. The highest accuracy achieved for the five-ion signal was only 19%, barely better than random guessing. This is due to the increasing complexity of the signal, where overlapping states and noise make it challenging for classical algorithms to differentiate between multiple ion channels accurately. The challenge lies in tuning the hyperparameters of DBSCAN, particularly ϵ and $n_{\min}$, to match the number of clusters [61]. However, even when the correct number of clusters

is identified, the overall classification accuracy remains far from ideal, especially as the complexity of the signal grows.

In addition to the accuracy trends, Fig. 3.12 presents a superimposed plot of the number of clusters detected across a range of ϵ values for all ion channels. The plot highlights how the challenge of parameter tuning intensifies as the number of ion channels increases. For simpler signals (e.g., one or two ion channels), DBSCAN successfully identifies clusters over a broader range of ϵ values. However, as the number of ion channels grows, the range of effective ϵ values narrows, and the clustering process becomes more sensitive to small changes in this parameter [39]. This is particularly evident in the four- and five-ion signals, where the parameter space for meaningful clustering is significantly reduced. Additionally, the fluctuations in cluster count across different ϵ values show that achieving a stable clustering solution becomes increasingly difficult with higher signal complexity.

This visualization reinforces the observation that DBSCAN struggles to maintain accuracy and effectiveness as the signal complexity increases, with the overlap in optimal parameter regions shrinking accordingly. While DBSCAN's simplicity and interpretability make it a valuable tool, these results suggest that for more complex signals, particularly those with four or more ion channels, DBSCAN requires extremely precise tuning to work effectively. This sets the stage for the next chapter, where we extend DBSCAN by incorporating distributional information to create a more robust clustering method capable of handling these complex signals, all while remaining within the domain of classical algorithms.

3.1.2 DBSCAN and BGMM

In this section, we present the results of applying a hybrid approach that combines distribution-based peak detection with DBSCAN and Bayesian Gaussian Mixture Models (BGMM) for classifying ion channel states. The focus here is on the key parameters that influence the accuracy and robustness of this method, particularly:

- The number of bins used in the histogram for peak detection.
- The ϵ and $n_{\min}$ parameters for DBSCAN, applied to the peak points detected in the histogram.
- The number of clusters derived from DBSCAN, used as input for the BGMM.

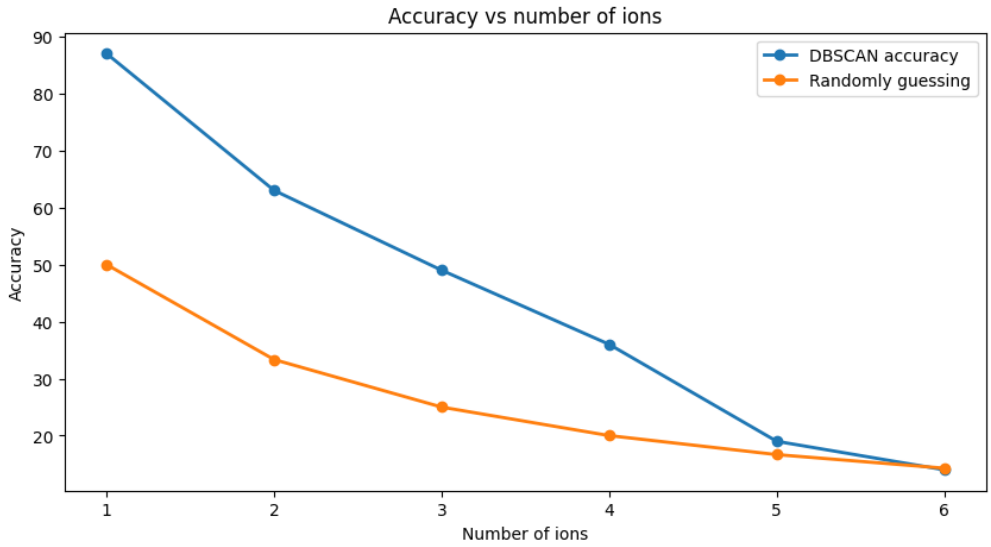


Figure 3.11: Accuracy vs. number of ion channels for DBSCAN and random guessing.

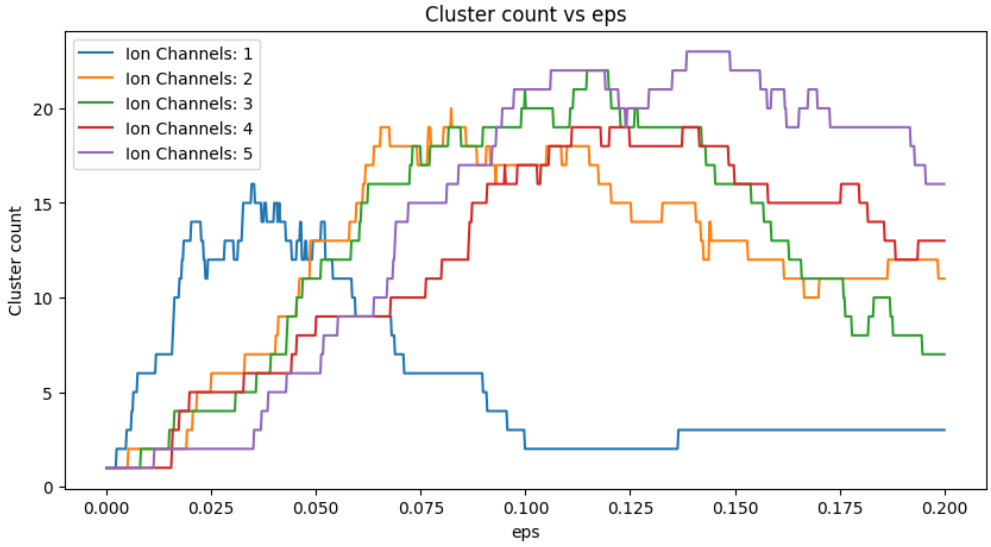


Figure 3.12: Superimposed ϵ values for all ion channels.

The following subsections analyze how variations in these parameters affect the method's performance, including the accuracy of ion channel detection and the stability of the clustering results. The primary focus is on understanding the interaction between these parameters, identifying optimal settings, and observing how the combined approach handles increasing signal complexity as the number of ion channels grows. The results demonstrate whether the hybrid method can mitigate the limitations of standalone DBSCAN, particularly when dealing with complex ion channel signals.

3.1.2.1 Impact of Histogram Bin Count on Peak Detection

Peak detection plays a pivotal role in determining the number of ion channels present in the signal before applying DBSCAN. After applying histogram-based peak detection, the number of peaks is identified based on the bin count, where the number of bins is a key parameter that directly affects the detection of peaks.

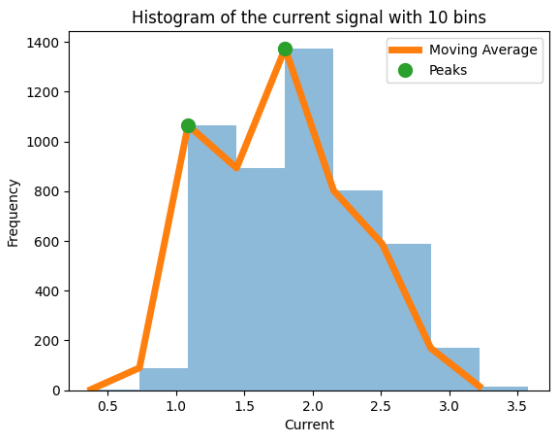
The primary parameter that influences this process is the number of bins in the histogram. The number of bins determines how finely the signal's distribution is divided. This parameter must be carefully tuned to ensure accurate peak detection for further clustering with DBSCAN.

Figure 3.13 illustrates the effect of varying the number of bins in the histogram. With fewer bins (e.g., 10), the histogram under-samples the signal, resulting in missed peaks. As the number of bins increases, more peaks are detected, but an excessively high number of bins (e.g., 60) introduces false peaks, which can distort the DBSCAN clustering. The optimal bin count needs to balance these two competing factors: ensuring significant peaks are captured without introducing too many false peaks.

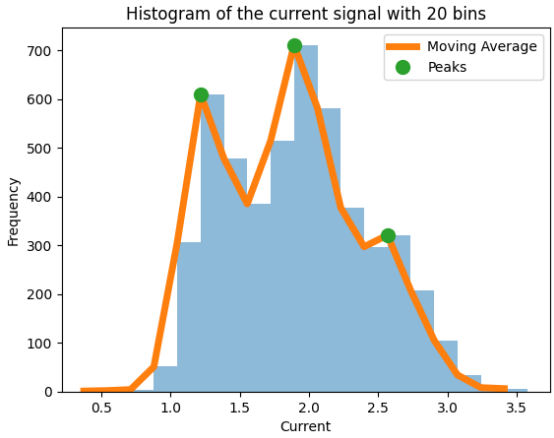
Figure 3.14 shows the relationship between the number of histogram bins and the number of detected peaks for the synthetic signal. The number of peaks detected increases steadily as the bin count rises. This analysis focuses solely on the impact of bin count on peak detection, illustrating that careful tuning of the bin count is crucial for accurate detection and subsequent DBSCAN clustering.

In analyzing the peak detection from the histogram, it is important to note that the presence of noise in the signal can lead to numerous small fluctuations at both the beginning and end of the histogram. These fluctuations may erroneously be counted as peaks, which can skew the results. To mitigate this issue, we establish a threshold value, as illustrated in Figures 3.15a and 3.15b, 3.15c and 3.15d, and 3.15e and 3.15f. By setting this threshold, we filter out minor fluctuations and only consider peaks that exceed this predefined value. This approach enhances the accuracy of peak detection and ensures that only significant peaks contribute to the subsequent analysis.

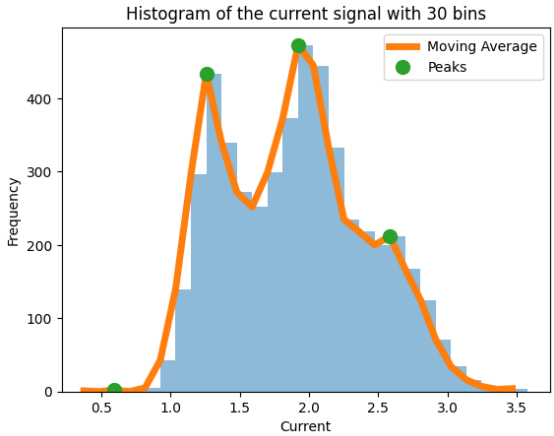
To further improve peak detection and minimize the influence of noise in the histogram, we implement a thresholding technique. By calculating the mean value of the signal, we set a threshold at 5% of this mean. This criterion effectively filters out peaks that are insignificant or attributed



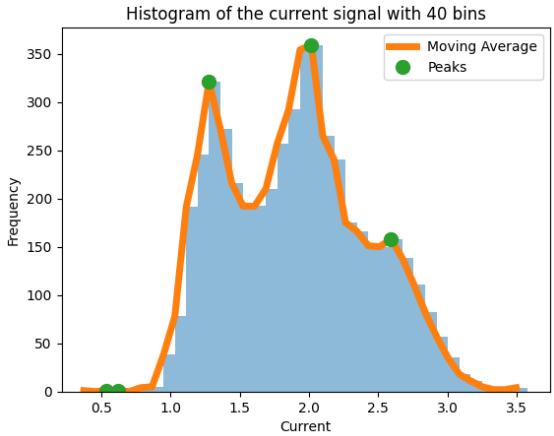
(a) 10 bins



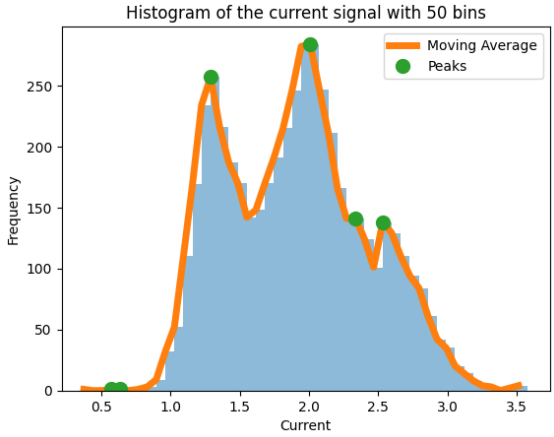
(b) 20 bins



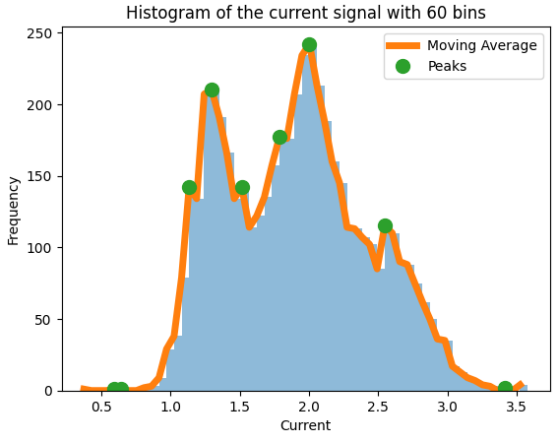
(c) 30 bins



(d) 40 bins



(e) 50 bins



(f) 60 bins

Figure 3.13: Effect of varying the number of bins on peak detection on a synthetic signal containing two ion channels (three states). As the number of bins increases, the number of detected peaks also increases. However, an excessively high number of bins may lead to false peaks, while too few bins may miss important peaks.

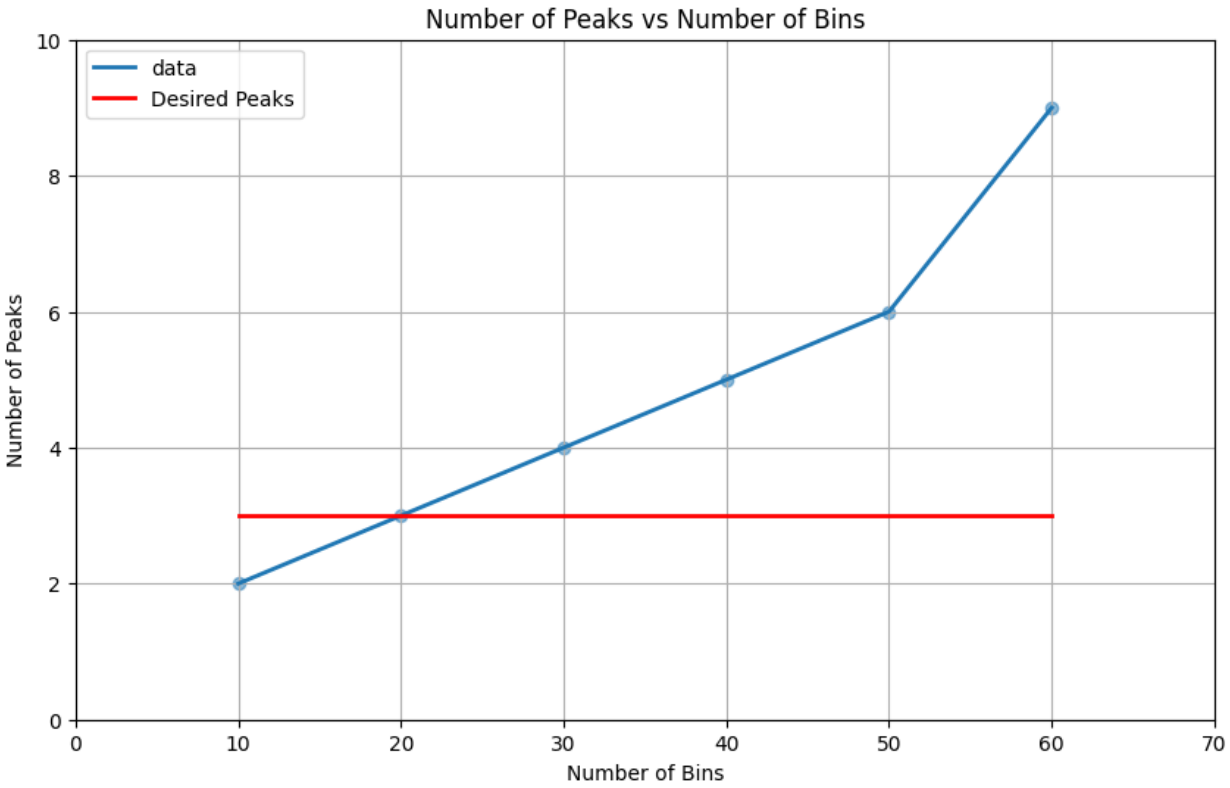


Figure 3.14: Number of bins vs. Number of peaks detected for the same synthetic signal used in 3.13. As the number of bins increases, the number of detected peaks rises, highlighting the sensitivity of peak detection to bin count.

to noise, particularly the small fluctuations observed at the beginning and end of the histogram.

By applying this threshold, we ensure that only peaks exceeding 5% of the mean signal value are considered valid for further analysis. This approach not only simplifies the peak detection process but also improves the robustness of the clustering results by focusing solely on significant peaks that are more likely to correspond to actual ion channel states. Consequently, this enhances the overall accuracy of the subsequent classification using DBSCAN and BGMM.

In order to select the optimal number of bins for peak detection, we explored the relationship between the number of bins and the detected number of states. As shown in the generated images, for each ion channel, the number of states was plotted against the number of bins. The plots also include the mean number of states detected for each bin count, along with the expected number of states for the ion channel (represented by the red dashed line).

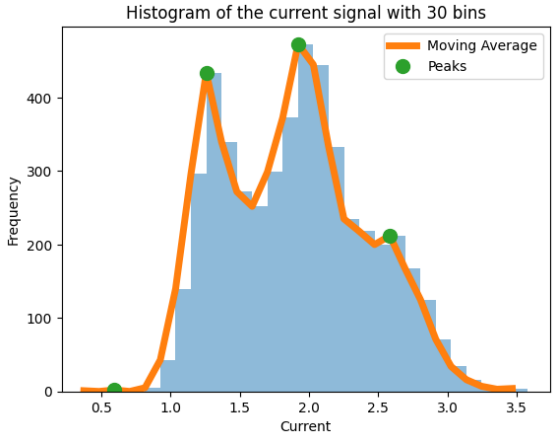
To avoid underestimation of the number of states, we implemented the following selection criterion: the optimal number of bins is chosen when the mean number of detected states is at least one more than the expected number of states. This ensures that we do not miss any actual states while tolerating a slight overestimation to account for minor noise-induced fluctuations.

The logic behind this criterion is to provide a buffer against underestimating the true number of states, which could result in missed states during peak detection. By allowing for a mean state count that exceeds the expected number of states by one, we ensure that the significant peaks corresponding to ion channel states are captured, while also avoiding an excessive number of false peaks due to noise.

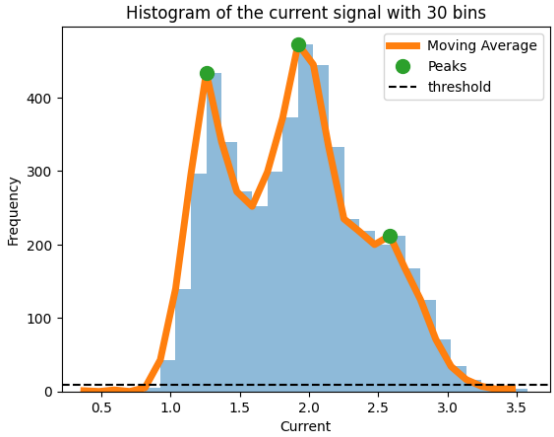
This approach is demonstrated in the figures, where we can observe that the mean state count increases with the number of bins. By selecting a bin count that satisfies our criterion (mean state count $\geq$ expected states $+1$), we optimize the peak detection process for accuracy and robustness.

In addition to analyzing the relationship between the number of bins and the detected states for each ion channel, we also investigated the trend in the optimal number of bins required to correctly detect the expected number of states as the number of ion channels increases. Figure 3.17 shows the mean number of bins that give the correct number of states for ion channels with different numbers of ion channels.

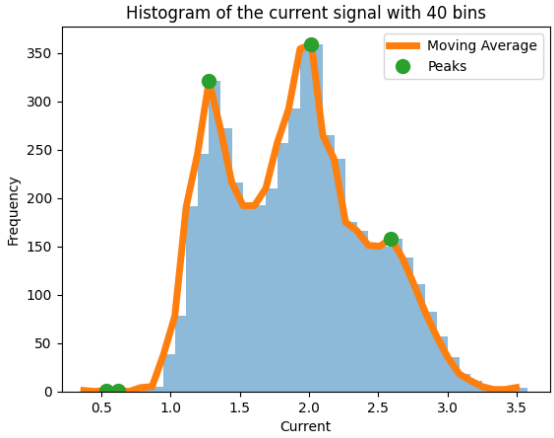
This figure helps validate the bin selection approach by illustrating that, as the number of ion channels increases, the number of bins required to achieve accurate peak detection also increases.



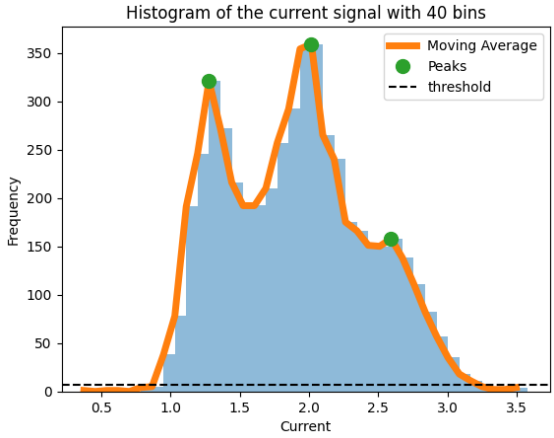
(a) 30 bins - Before Threshold



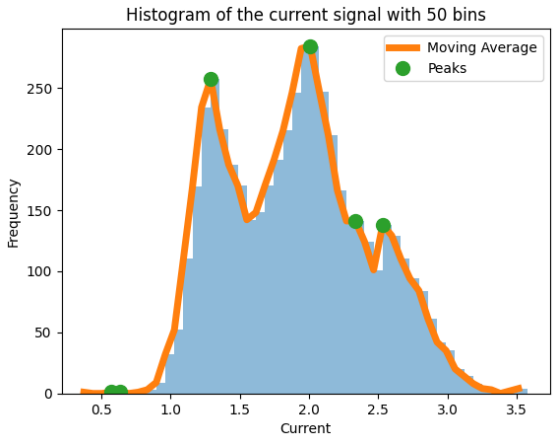
(b) 30 bins - After Threshold



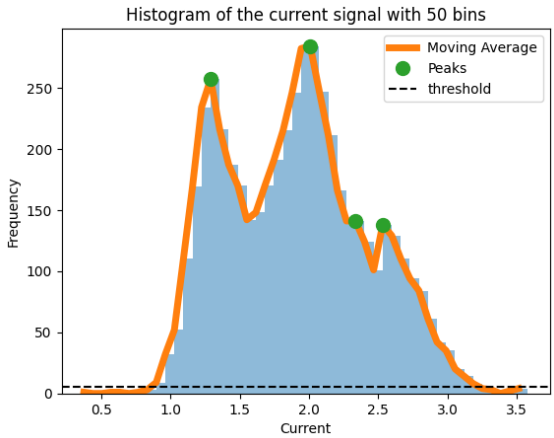
(c) 40 bins - Before Threshold



(d) 40 bins - After Threshold

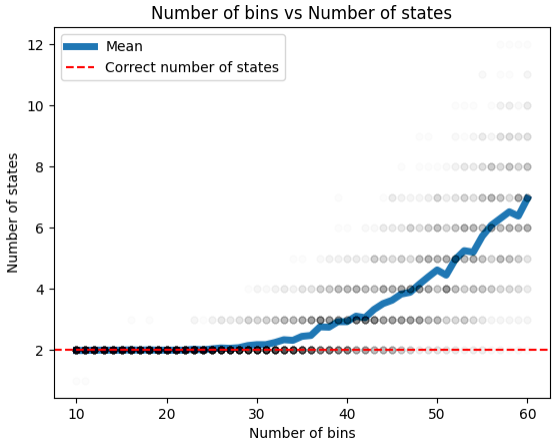


(e) 50 bins - Before Threshold

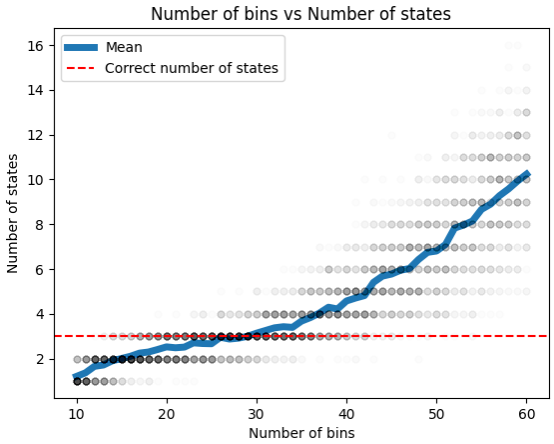


(f) 50 bins - After Threshold

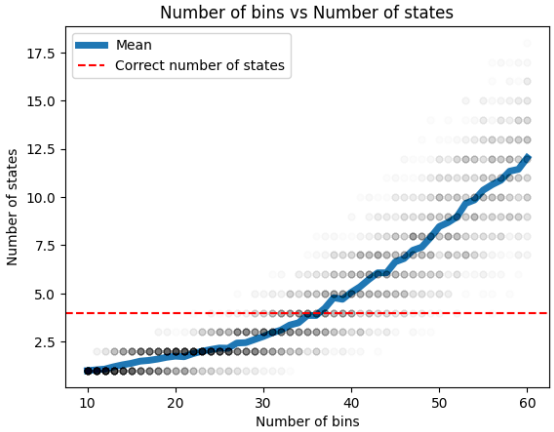
Figure 3.15: Comparison of peak detection histograms before and after applying a threshold for signals with 30, 40, and 50 bins. The application of a threshold significantly reduces noise-induced false peaks, enhancing the accuracy of peak detection for subsequent clustering.



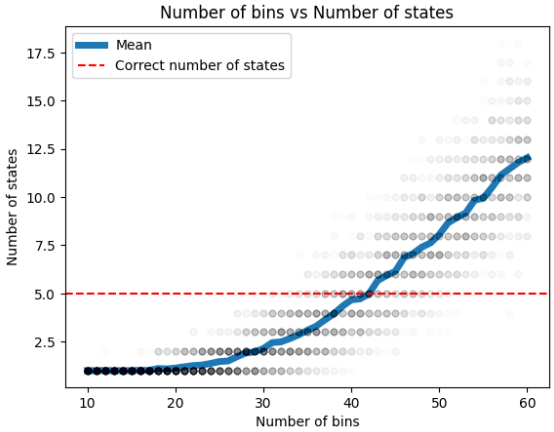
(a) 1 ion channel



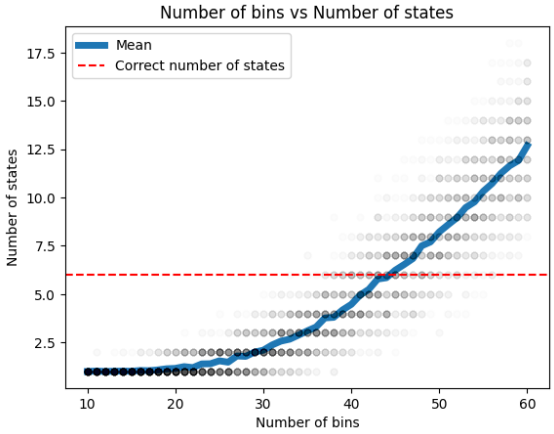
(b) 2 ion channels



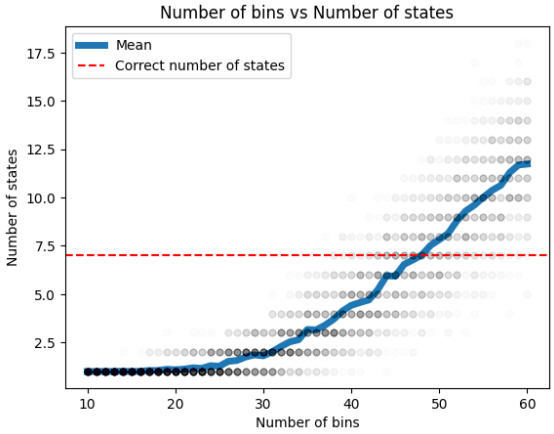
(c) 3 ion channels



(d) 4 ion channels



(e) 5 ion channels



(f) 6 ion channels

Figure 3.16: Number of bins vs number of states for each ion channel dataset. Each plot shows the detected number of states as a function of the number of bins used in the histogram. The dark blue line represents the mean number of detected states, while the red dashed line indicates the correct (expected) number of states for the given ion channel. This comparison highlights the relationship between binning resolution and state detection accuracy, demonstrating how increasing the number of bins leads to an overestimation of states in the presence of noise.

The trend is relatively consistent, with the required number of bins generally rising as the complexity of the signal (i.e., the number of ion channels) increases.

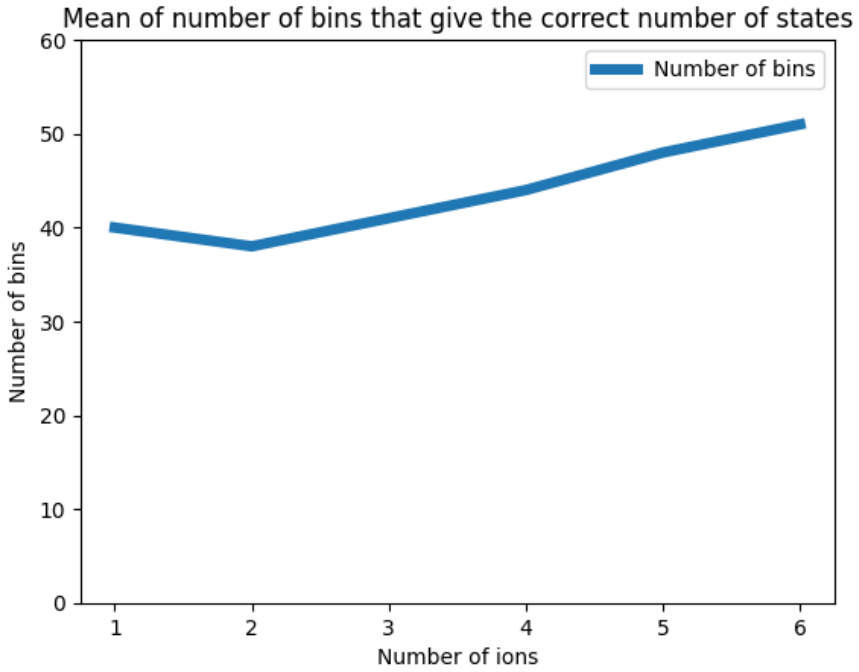


Figure 3.17: Mean number of bins required to detect the correct number of states as a function of the number of ion channels in the signal. The trend shows a gradual increase in the required number of bins as the number of ion channels increases, indicating that signals with more complexity need more bins for accurate peak detection.

In cases where the number of ion channels is unknown, selecting an optimal bin number requires a balanced approach that ensures sensitivity to significant peaks while mitigating noise. Based on the analysis presented earlier, a bin number in the range of 40 to 50 has been shown to provide reliable results for signals of varying complexity. By considering the mean number of bins required for correct peak detection across different ion channels, as shown in Figure 3.17, we propose selecting a general bin number of 45. This value offers a compromise between detecting the correct number of states and minimizing false peaks due to noise. Moreover, the selection of 40 bins aligns well with the empirical results from various signals, ensuring robust and accurate peak detection even when the number of ion channels is unknown.

3.1.2.2 Clustering Peaks Using DBSCAN

In the context of peak detection for ion channel data, when using DBSCAN to cluster the peaks, the parameter $n_{\min}$ is set to 1. The reasoning behind this choice stems from the nature of the peaks. Since each peak corresponds to a distinct state of the ion channel, and in an ideal, noise-free signal, we expect a clean separation of these states, where each state would be represented by a single peak. Therefore, setting $n_{\min}$ to any value greater than 1 might discard valid peaks that represent actual ion channel states, especially when dealing with clean data.

On the other hand, the parameter ϵ becomes the critical variable to determine. ϵ defines the maximum distance between two points for them to be considered as part of the same cluster [58]. Given that $n_{\min}$ is fixed, the correct choice of ϵ is essential for properly grouping points into clusters while separating the noise. The value of ϵ needs to be chosen carefully to ensure that it captures peaks that are part of the same state while not merging distinct peaks, especially in signals with a certain degree of noise.

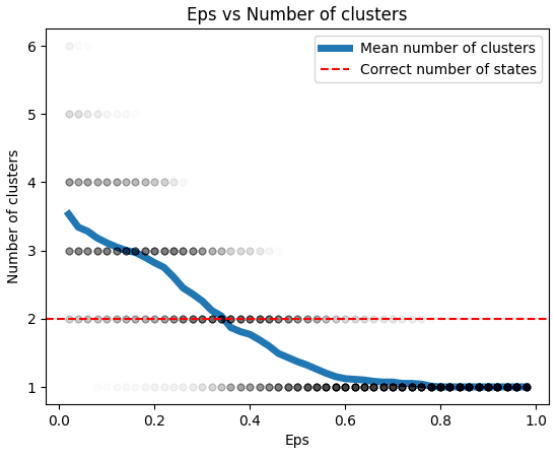
Thus, for this application, the main focus will be on tuning the ϵ parameter to achieve optimal clustering of peaks corresponding to ion channel states. In order to determine the optimal value of the ϵ parameter for DBSCAN, we analyze the relationship between ϵ and the number of clusters formed. For each ion channel, we examine the value of ϵ that yields the number of clusters closest to the expected number of states. This analysis is visualized in the six graphs generated for ion channels 1 through 6, similar to the bin analysis conducted previously.

Figure 3.18 illustrates the mean number of clusters as a function of ϵ, with the correct number of states indicated by a red dashed line. By identifying the point where the mean number of clusters intersects the expected number of states, we can determine the optimal ϵ value for each ion channel.

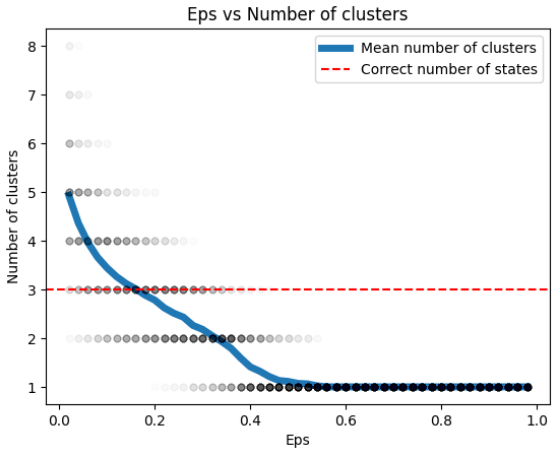
This approach allows us to systematically choose the best ϵ parameter that minimizes the impact of noise while ensuring accurate clustering of the peaks corresponding to ion channel states.

Based on the analysis for each ion channel, we calculated the optimal ϵ values that yield the correct number of states. By averaging the ϵ values across all six ion channels, we obtained the results presented in Figure 3.19.

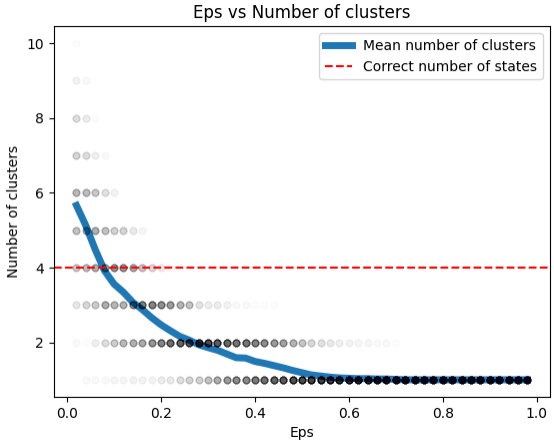
The graph illustrates a downward trend in the mean ϵ as the number of ion channels increases, suggesting that smaller ϵ values are required for larger ion channels to accurately detect the number



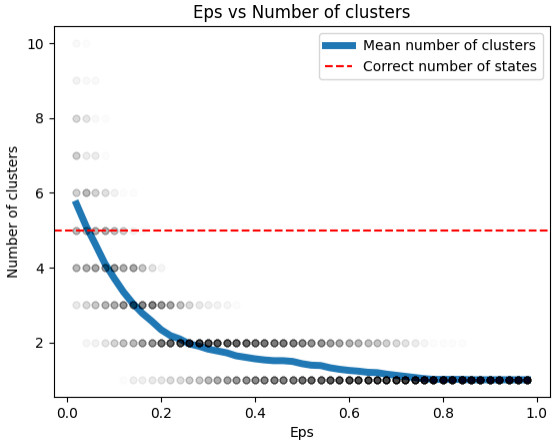
(a) 1 ion channel



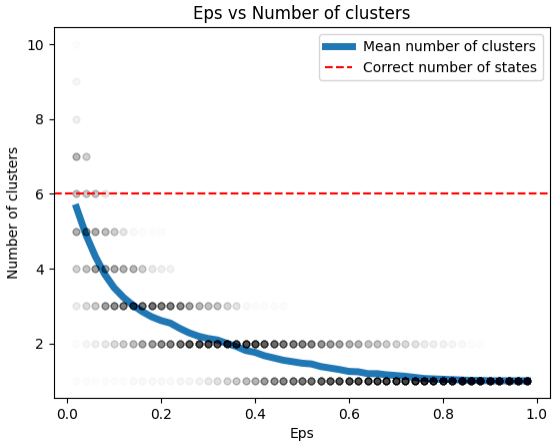
(b) 2 ion channels



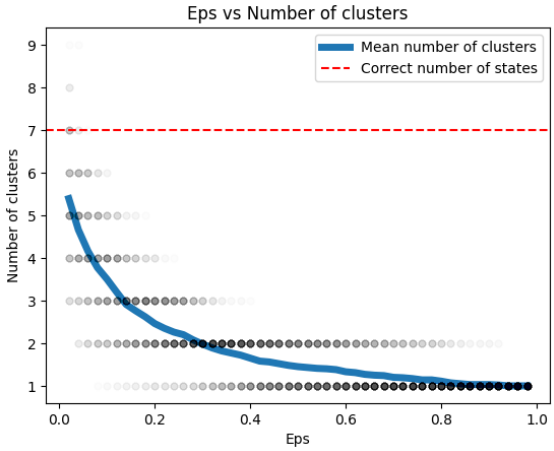
(c) 3 ion channels



(d) 4 ion channels



(e) 5 ion channels



(f) 6 ion channels

Figure 3.18: Mean number of clusters as a function of ϵ, with the correct number of states for ion channel 1 highlighted by the red dashed line. Similar graphs for ion channels 2 through 6 follow the same pattern.

of states. This trend reflects the diminishing distance between peaks as the number of ion channels grows, thus requiring finer distinctions in the clustering process.

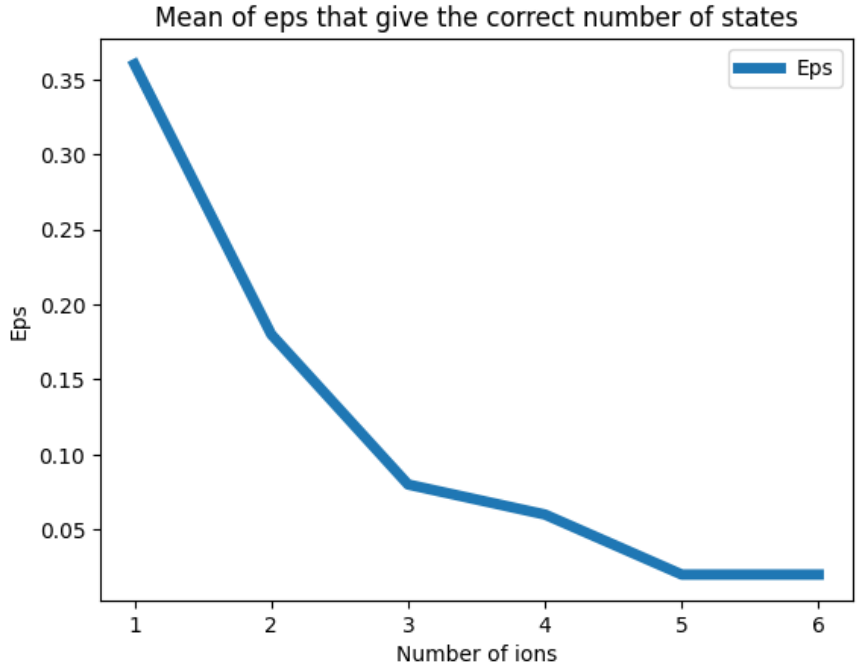


Figure 3.19: Mean ϵ values that result in the correct number of clusters for each ion channel.

As we observed, the primary challenge is selecting the appropriate ϵ for a given signal. The complexity and noise level of the signal often correlate with the number of ion channels present. Consequently, for noisier or more complex signals, the ϵ value needs to be smaller to accurately detect peaks and cluster them appropriately.

To aid in selecting the correct ϵ, we can create a table that associates peak ranges with optimal ϵ values based on the signal complexity. The table below presents a heuristic approach for choosing the ϵ based on the number of peaks identified in the signal:

Peak Range	**Recommended ϵ**
1–2	0.30–0.35
3–4	0.15–0.20
5–6	0.05–0.10
7+	0.01–0.05

Table 3.1: Recommended ϵ values based on peak ranges in the signal.

This table serves as a guide for selecting the ϵ value depending on the number of peaks observed

in the signal. For signals with fewer peaks, a larger ϵ is appropriate, while for signals with more peaks and potentially more noise, smaller ϵ values provide more accurate clustering.

3.1.2.3 Using the Number of Clusters from DBSCAN as Input for BGMM

After obtaining the clusters from DBSCAN, the next step involves using the number of clusters identified by DBSCAN as the input parameter for the Bayesian Gaussian Mixture Model (BGMM) [53]. This allows us to classify each point in the signal based on the clusters formed by DBSCAN.

The process can be described as follows:

1. First, the signal is processed to detect peaks.
2. The DBSCAN algorithm is applied to the peaks, where the ϵ parameter is tuned based on the signal complexity, as discussed in the previous section.
3. DBSCAN outputs the number of clusters, representing the grouping of peaks.
4. The number of clusters derived from DBSCAN is then used as the input parameter for the Bayesian Gaussian Mixture Model (BGMM).

The BGMM is responsible for classifying each point in the signal to one of the clusters, allowing for probabilistic assignment and uncertainty quantification [62]. By leveraging DBSCAN for clustering and feeding the cluster count into BGMM, we can effectively capture the underlying structure of the signal and perform accurate classification for further analysis.

We perform two analyses to evaluate the performance of our approach:

1. **Accuracy over 1 to 6 ion channels**: This analysis assesses the accuracy of the clustering and classification method across different signals containing between 1 and 6 ion channels. We aim to determine how well the method detects the correct state of states in each signal.
2. **Average Accuracy of Points over 1 to 6 Ion Channels**: The second analysis evaluates the average accuracy of correctly assigning each datapoint to its true cluster, as derived from the BGMM model. This index provides insight into how well the model detects the correct cluster for each point across signals containing different numbers of ion channels.

3.1.2.4 Average Identified Number of Ion Channels vs Actual Value

In this analysis, we assess the model's ability to detect the correct number of ion channels in signals containing between 1 to 6 ion channels [63]. For each signal, the DBSCAN algorithm identifies clusters, and the number of clusters is interpreted as the number of ion channels. These results are compared with the expected number of ion channels present in the signal.

As shown in Figure 3.20, the orange line represents the expected number of ion channels, while the blue line shows the average number of states (clusters) identified by the model. The model performs relatively well, with the identified number of states closely matching the actual ion channels. However, as the number of ion channels increases to 6, there is a noticeable divergence, with the model detecting fewer states than expected. This suggests that the model may struggle with more complex signals.

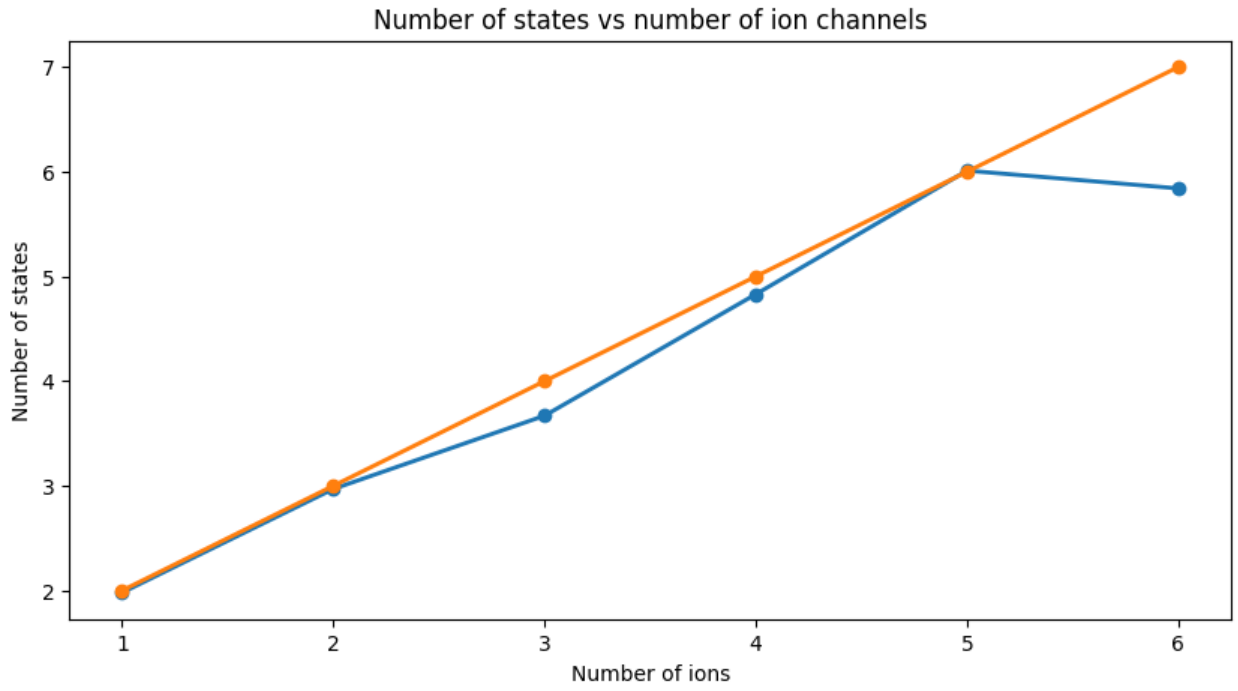


Figure 3.20: Average identified number of ion channels (states) vs actual ion channel count

3.1.2.5 Conclusion

The model demonstrates a strong performance in detecting the number of ion channels for simpler signals with fewer ion channels (1 to 4). However, its performance decreases as the signal complexity

increases, particularly for signals with 5 or more ion channels, where it tends to underestimate the number of states. This underestimation could be due to limitations in clustering methods like DBSCAN when dealing with higher noise levels and overlapping data points.

3.1.2.6 Average Accuracy of Points Over 1 to 6 Ion Channels

For each ion channel count, we calculate the average accuracy of points being correctly assigned to their respective clusters. This accuracy is given by:

$$\text{Avg. Accuracy} = \frac{\text{Number of Correctly Classified Points}}{\text{Total Number of Points}} \times 100$$

This index provides a straightforward assessment of how effectively the model identifies the correct clusters for individual datapoints, with the goal of maximizing classification accuracy across varying signal complexities (1 to 6 ion channels).

Figure 3.21 shows the results of this comparison. The blue line represents the accuracy achieved using DBSCAN alone, while the orange line corresponds to the accuracy achieved using the DBSCAN + BGMM hybrid model [64]. For reference, the green line represents the expected accuracy when guessing randomly.

From the figure, it is evident that the hybrid model consistently outperforms DBSCAN alone in terms of accuracy. For signals containing one ion channel, both models perform similarly, with an accuracy of over 80%. As the number of ion channels increases, the performance of both models declines, but the hybrid model maintains higher accuracy throughout. Notably, at six ion channels, the hybrid model achieves an accuracy of approximately 40%, compared to 14% for DBSCAN alone.

The DBSCAN + BGMM hybrid model clearly offers a substantial improvement in classification accuracy over DBSCAN alone, particularly as the complexity of the signal (number of ion channels) increases. This is due to the hybrid model's ability to leverage the initial clustering of DBSCAN while further refining the classification using BGMM's probabilistic approach, making it more robust against the increasing complexity and noise of the signal. Therefore, for more complex signals with multiple ion channels, the hybrid approach is recommended for optimal clustering and classification accuracy.

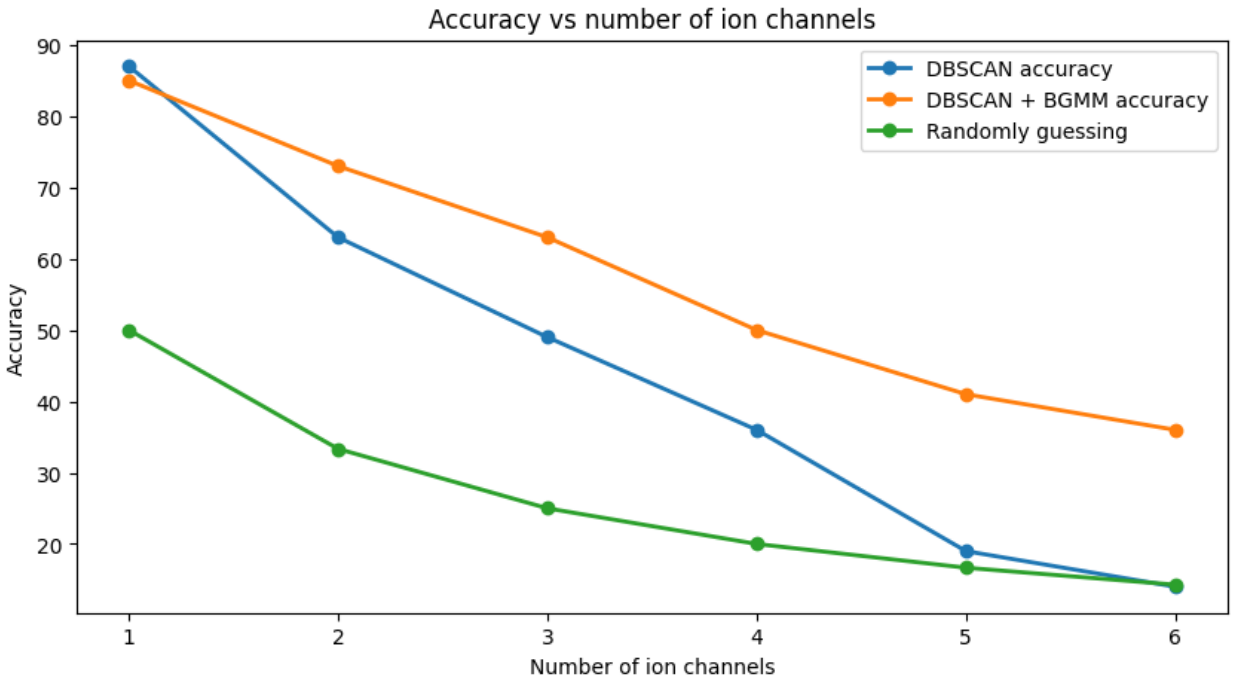


Figure 3.21: Comparison of average accuracy between DBSCAN and DBSCAN + BGMM hybrid models across different ion channel counts

3.2 Machine Learning Algorithms

In this section, we transition from classical clustering algorithms to machine learning-based approaches for detecting the correct number of ion channels and predicting the states of individual data points. While classical algorithms like DBSCAN provide a solid foundation for simpler signals, they encounter limitations when handling more complex and noisy datasets [34].

To address these challenges, we first develop a machine learning model designed to accurately estimate the correct number of clusters (states) for each signal. This model improves on the classical methods by leveraging deeper patterns within the data that may not be captured by traditional clustering techniques alone.

Following this, we utilize a Long Short-Term Memory (LSTM) model, which builds upon the cluster information from the initial model to classify each data point into its respective state [65]. The LSTM model's ability to incorporate temporal dependencies makes it particularly well-suited for this task, as it can learn the dynamic behaviors of ion channels over time and improve state prediction accuracy.

Together, these machine learning methods form a more comprehensive approach to ion channel state detection, allowing us to achieve higher accuracy and more reliable predictions in the presence of complex and noisy signals.

3.2.1 Cluster Detection Model

The first step in our machine learning approach involves developing a model to accurately detect the number of clusters (ion channels) present in the signal. The structure of the model is a fully connected neural network (FCNN), designed to learn from the input signal and predict the correct number of clusters. In this subsection, we evaluate the impact of several key parameters on the model's performance:

- **Input Size (Number of Bins)**: The input size is determined by the number of bins used to discretize the signal data. We analyze how varying the number of bins affects the model's ability to accurately predict the number of clusters.
- **Depth of the Model**: The depth refers to the number of hidden layers in the fully connected network. By increasing or decreasing the depth, we aim to determine the optimal number of layers that balance model complexity and performance.
- **Size of Each Layer**: The size of each layer corresponds to the number of neurons in the hidden layers. We investigate the effect of increasing or decreasing the layer size while keeping other parameters fixed, to understand how the model's capacity influences its performance.

For each of these parameters, we conduct experiments by varying one parameter at a time while keeping the others fixed. This allows us to observe the specific influence of each parameter on the performance of the cluster detection model, providing insights into the optimal configuration for accurate state detection.

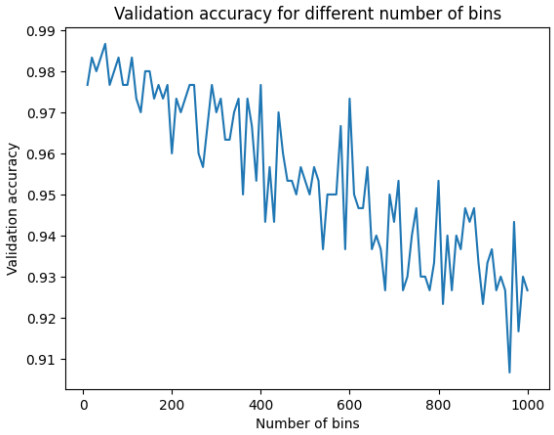
3.2.1.1 Effect of Input Size (Number of Bins) on Performance

The input size, defined by the number of bins, plays a significant role in the performance of the clustering model. We conducted experiments varying the number of bins from 1 to 1000, with a step size of 10, to understand the impact of input size on validation accuracy. The results, as

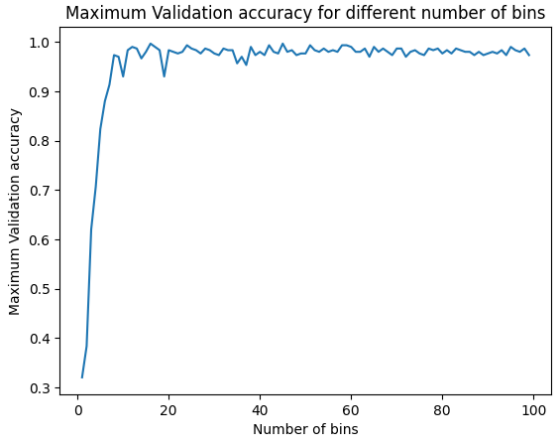
shown in Figure 3.22a, suggest that the optimal number of bins lies around 50. However, to further investigate this range, we ran an additional experiment focusing on the 1 to 100 bins range, with a step size of 1.

Interestingly, as depicted in Figure 3.22b, we observed that 16 bins resulted in the best validation performance within this smaller range. This could be because 16 bins provide a balance between capturing enough detail in the distribution of the input data without introducing excessive noise or overfitting, allowing the model to generalize better during validation. This observation prompted us to analyze the validation accuracy across epochs for these experiments, which is displayed in Figures 3.23. Both experiments demonstrate a significant increase in performance after 10 epochs, followed by gradual convergence.

These results underscore the importance of selecting an optimal input size. Based on these findings, an input size of 16 bins appears to provide the best performance for our clustering task.



(a) Maximum validation accuracy for different number of bins (1 to 1000 bins with step size of 10).

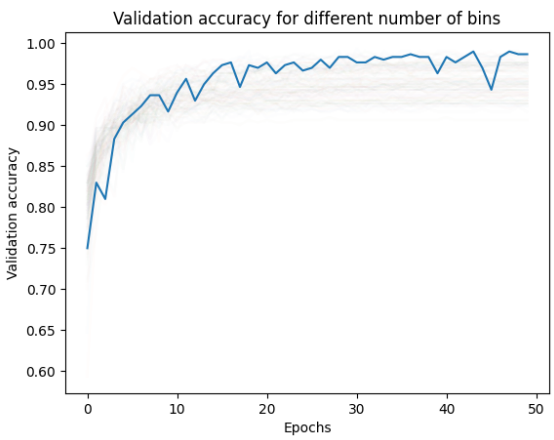


(b) Maximum validation accuracy for different number of bins (1 to 100 bins with step size of 1).

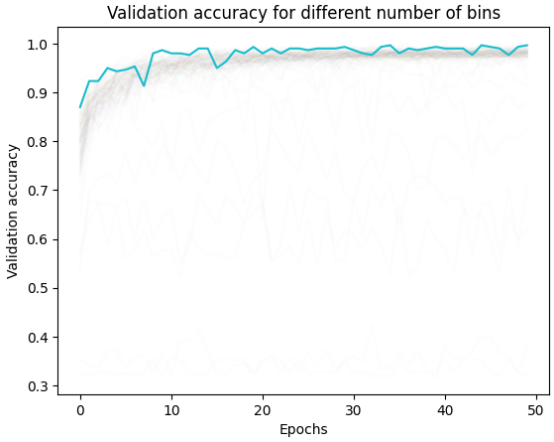
Figure 3.22: Comparison of maximum validation accuracy for different bin configurations, highlighting the impact of varying bin sizes on model performance.

3.2.1.2 Model Depth

After determining the optimal input size, we shifted our focus to assessing the effect of model depth, defined by the number of layers in the architecture. We conducted experiments varying the number of layers from 1 to 50, with all other hyperparameters remaining constant, to evaluate how



(a) Validation accuracy over epochs for different number of bins (1 to 1000 bins with step size of 10). The graph corresponding to 50 bins is highlighted in blue, showing the highest validation accuracy at the end of training, while other bin configurations are represented in light grey.



(b) Validation accuracy over epochs for different number of bins (1 to 100 bins with step size of 1). The graph corresponding to 16 bins is highlighted in blue, showing the highest validation accuracy at the end of training, while other bin configurations are represented in light grey.

Figure 3.23: Comparison of validation accuracy over epochs for different bin configurations, highlighting performance trends across different bin sizes in two ranges.

the depth of the model impacts performance in terms of loss and validation accuracy.

As illustrated in Figure 3.24, the model with 13 layers, highlighted in the graph, achieved the lowest loss by the end of training. This suggests that increasing the depth up to a certain point improves the model's ability to minimize loss.

Furthermore, we analyzed the validation accuracy across epochs for each model depth, as shown in Figure 3.25. Similar to the loss trends, the 13-layer model, highlighted in the figure, achieved the highest validation accuracy.

Finally, we evaluated the maximum validation accuracy achieved for each number of layers, presented in Figure 3.26. The 13-layer model, marked by a red dot for visibility, achieved the highest validation accuracy overall. This result reinforces the observation that 13 layers strike an optimal balance between model complexity and performance, offering the best validation accuracy without introducing excessive overfitting or diminishing returns from added layers.

3.2.1.3 Number of Neurons per Layer

After determining the optimal model depth, we extended our analysis to investigate the effect of the number of neurons per layer on model performance. We experimented with models having between

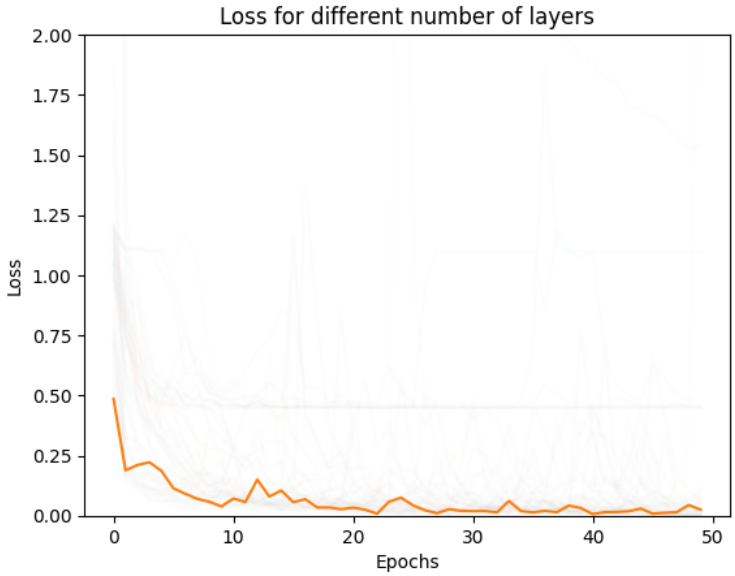


Figure 3.24: Loss over epochs for different number of layers (1 to 50 layers). The model with 13 layers, highlighted, achieved the lowest loss at the end of training.

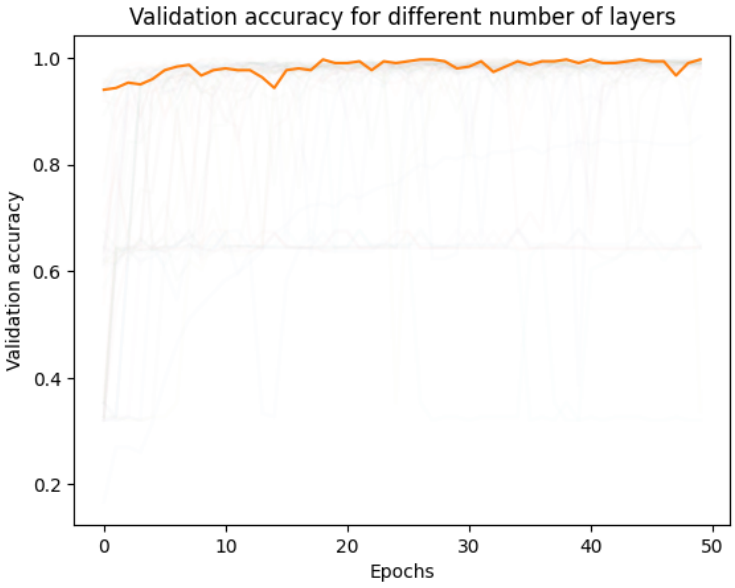


Figure 3.25: Validation accuracy over epochs for different number of layers (1 to 50 layers). The 13-layer model, highlighted, achieved the highest validation accuracy during training.

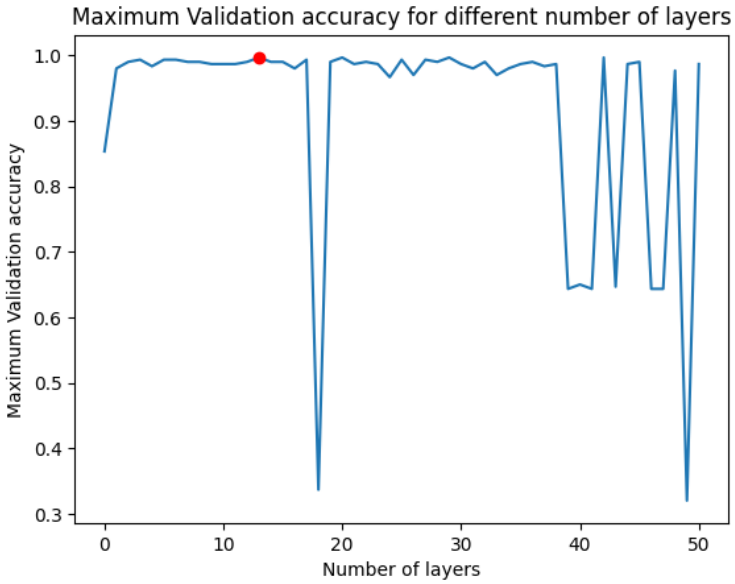


Figure 3.26: Maximum validation accuracy for different number of layers (1 to 50 layers). The 13-layer model, highlighted with a red dot, achieved the highest overall validation accuracy.

1 and 500 neurons per layer, with increments of 10 neurons, while keeping all other hyperparameters constant. The goal was to evaluate how varying the number of neurons influences the model's loss and validation accuracy.

As shown in Figure 3.27, the model with 170 neurons per layer, highlighted in the graph, achieved the lowest loss by the end of training. This suggests that having an optimal number of neurons allows the model to effectively capture the underlying patterns in the data without overfitting.

Additionally, we analyzed the validation accuracy across epochs for the different neuron configurations, as depicted in Figure 3.28. The model using 170 neurons, highlighted in the graph, achieved the highest validation accuracy.

Finally, Figure 3.29 presents the maximum validation accuracy achieved for each configuration of neurons per layer. The model with 170 neurons, marked by a red dot for visibility, achieved the highest overall validation accuracy. This result confirms that 170 neurons per layer offer the best balance between model complexity and performance, without incurring diminishing returns from using more neurons.

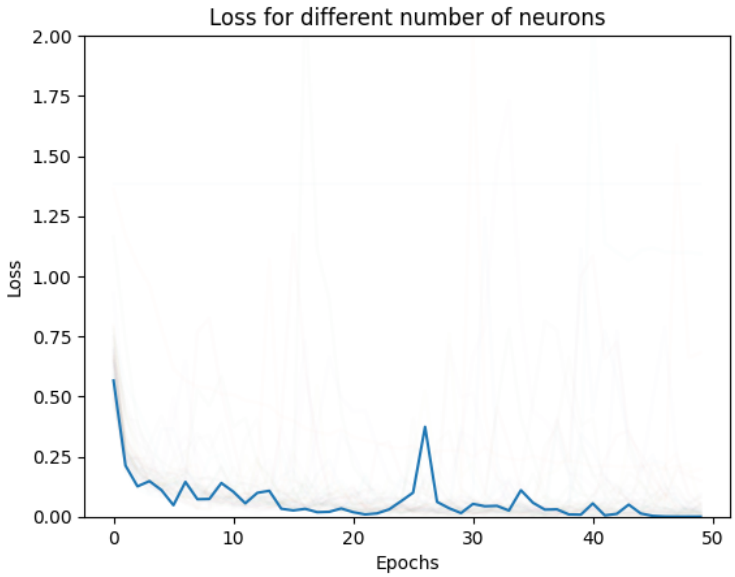


Figure 3.27: Loss over epochs for different number of neurons per layer (1 to 500 neurons, with steps of 10). The model with 170 neurons, highlighted, achieved the lowest loss at the end of training.

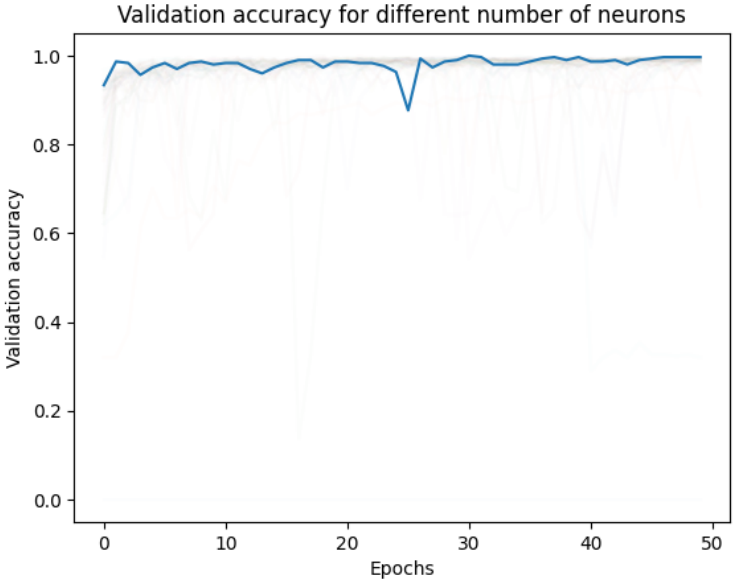


Figure 3.28: Validation accuracy over epochs for different number of neurons per layer (1 to 500 neurons, with steps of 10). The model with 170 neurons, highlighted, achieved the highest validation accuracy during training.

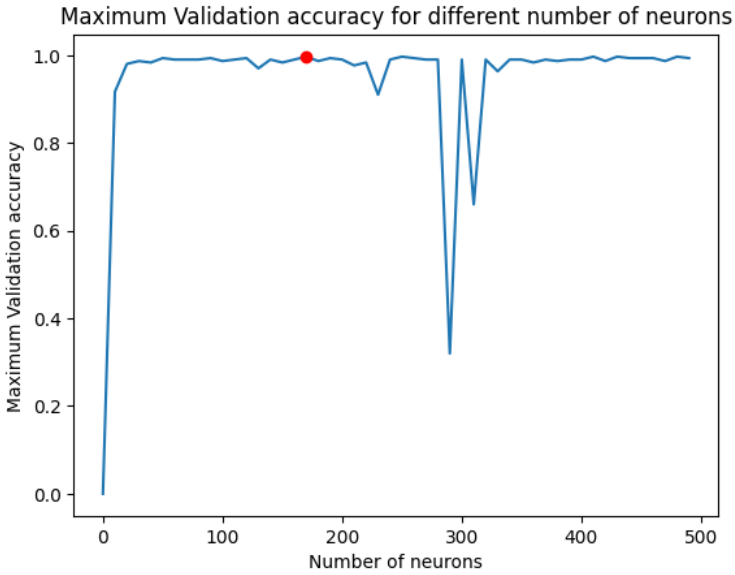


Figure 3.29: Maximum validation accuracy for different number of neurons per layer (1 to 500 neurons, with steps of 10). The model with 170 neurons, highlighted with a red dot, achieved the highest overall validation accuracy.

3.2.2 Combining Clustering Detection Model with LSTM for State Detection

After developing the clustering detection neural network (NN) model, we proceeded to combine it with a Long Short-Term Memory (LSTM) network to detect the correct state of the signal for a given point [66]. The key variables in this phase of experimentation were the input window size of the LSTM model and the number of neurons in the LSTM layer. The window size determines how many time steps before and after the given point are used as input to the LSTM, where the model predicts the state of the middle point.

3.2.2.1 Window Size

We first experimented with different window sizes: 10, 20, 30, 40, and 50. The goal was to assess the effect of window size on the model's performance, specifically its ability to correctly detect the signal's state. As the old saying in the community goes, "garbage in, garbage out"—the more irrelevant signal we pass to the model, the harder it becomes for the LSTM to detect the correct state.

For each window size, we produced both loss and accuracy plots, superimposing the values for the training and testing phases to assess the model's generalization performance. The results for

loss and accuracy are shown separately in Figure 3.30 and Figure 3.31, respectively.

As can be seen from the figures, the model with a window size of 10 yielded the best overall performance, achieving the lowest loss and highest accuracy. Interestingly, as the window size increased, the loss values for the training and validation phases began to diverge significantly, indicating a growing gap in performance between training and testing, which suggests overfitting. The same phenomenon is observed in the accuracy plots, where the model's accuracy on the validation set starts to degrade while training accuracy remains high. This divergence suggests that the model is starting to memorize patterns specific to the training data instead of generalizing from it. Additionally, as the window size grows, the accuracy decreases and the loss increases, further supporting the idea that larger windows introduce irrelevant or noisy information, making it harder for the LSTM to detect the correct state of the signal.

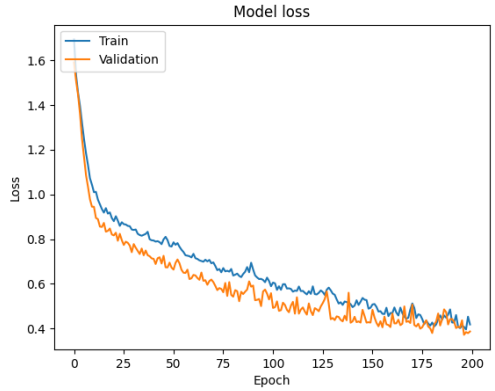
These findings highlight the importance of selecting an appropriate window size, with a size of 10 being optimal for this task, as it strikes the best balance between capturing useful context and avoiding unnecessary noise.

3.2.2.2 LSTM Layer Size

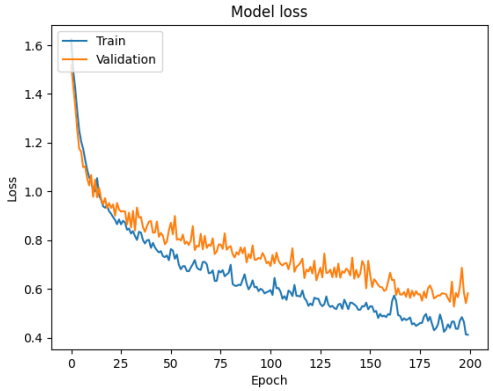
We conducted another set of experiments where we fixed the window size to the previously determined optimal value of 10 and varied the number of neurons in the LSTM layer. For each LSTM size, we generated both loss and accuracy plots for the training and testing phases. The results for the loss are shown in Figure 3.32, and the results for accuracy are shown in Figure 3.33.

From these figures, we can observe that the LSTM size does not have a significant impact on the overall results as long as the number of neurons is greater than 4. For LSTM sizes larger than 4, the performance stabilizes, and validation accuracy consistently remains above 80%. The best performance was achieved with an LSTM size of 64 neurons, resulting in an accuracy of 88.9%. Furthermore, as shown in both the loss and accuracy plots, increasing the LSTM size beyond 64 neurons did not yield any substantial improvements and, in some cases, led to slight overfitting as indicated by the divergence between training and validation curves.

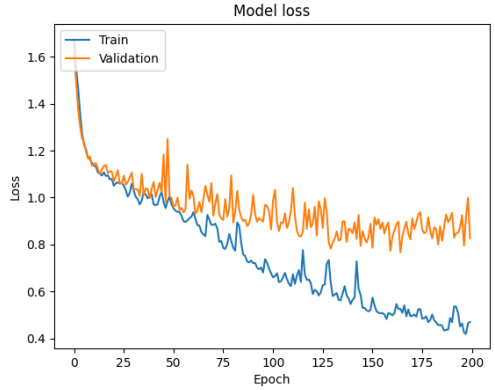
These findings suggest that while the number of LSTM neurons is an important parameter, once a certain threshold is reached (around 64 neurons), increasing the number of neurons further offers diminishing returns in terms of performance.



(a) Window size 10: Training and testing loss.

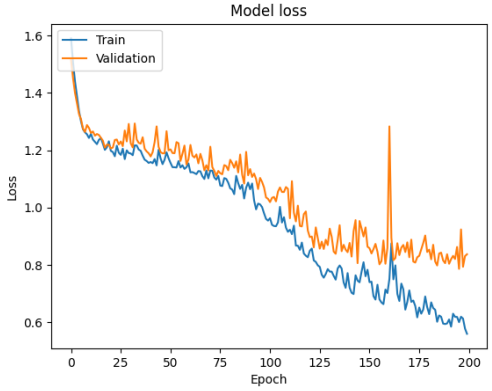


(b) Window size 20: Training and testing loss.



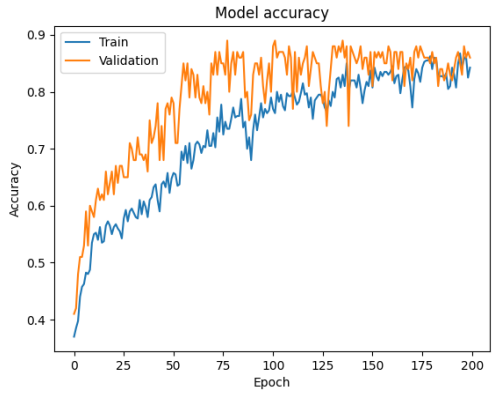
(c) Window size 30: Training and testing loss.

(d) Window size 40: Training and testing loss.

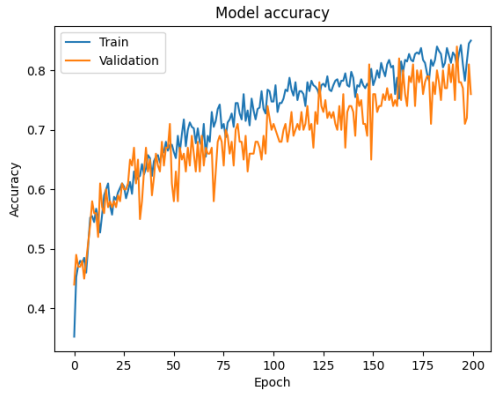


(e) Window size 50: Training and testing loss.

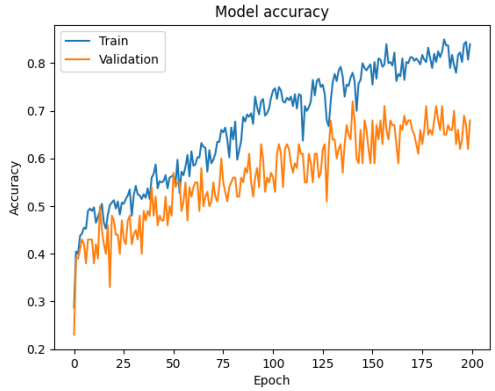
Figure 3.30: Training and testing loss for different window sizes (10, 20, 30, 40, 50). The graphs show the impact of the input window size on model performance in terms of loss. The best result was achieved with a window size of 10, which had the lowest loss.



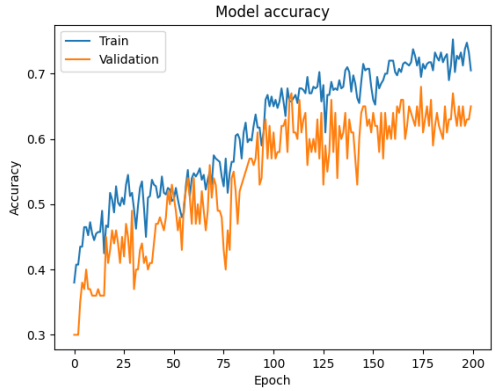
(a) Window size 10: Training and testing accuracy.



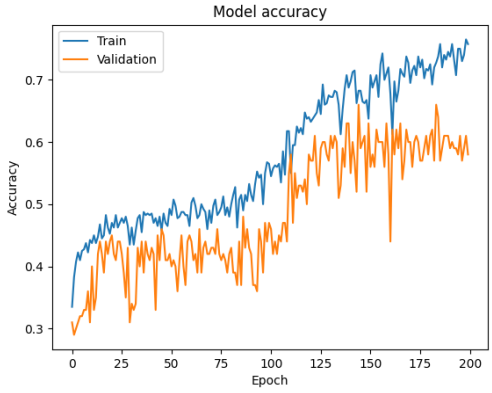
(b) Window size 20: Training and testing accuracy.



(c) Window size 30: Training and testing accuracy.

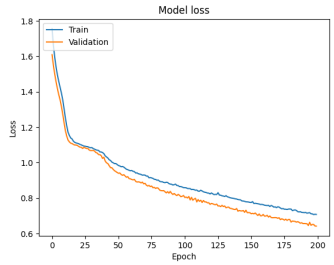


(d) Window size 40: Training and testing accuracy.

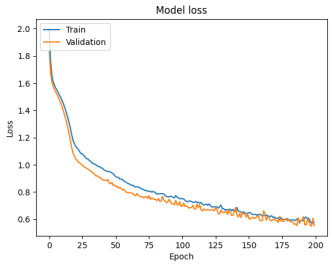


(e) Window size 50: Training and testing accuracy.

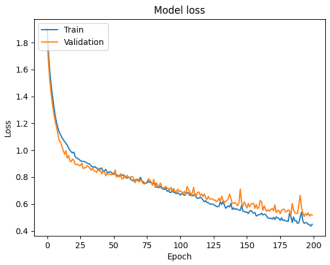
Figure 3.31: Training and testing accuracy for different window sizes (10, 20, 30, 40, 50). The graphs show how accuracy improves or degrades depending on the window size. Window size 10 provided the highest accuracy.



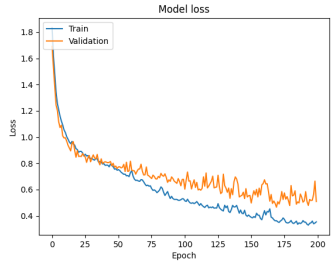
(a) LSTM with 4 neurons.



(b) LSTM with 8 neurons.

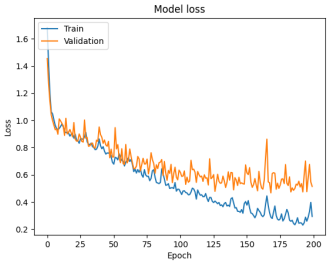


(c) LSTM with 16 neurons.

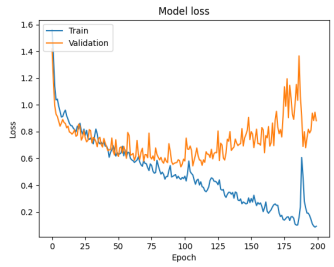


(d) LSTM with 32 neurons.

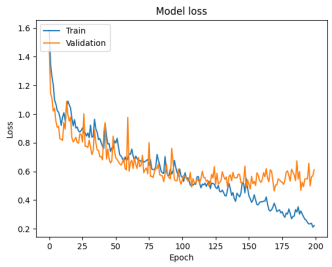
(e) LSTM with 64 neurons.



(f) LSTM with 128 neurons.

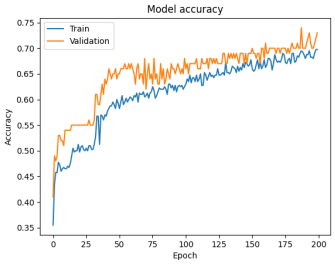


(g) LSTM with 256 neurons.

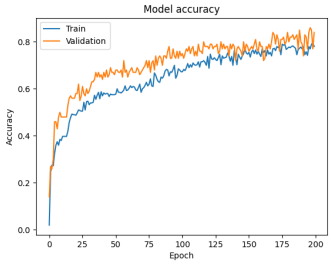


(h) LSTM with 512 neurons.

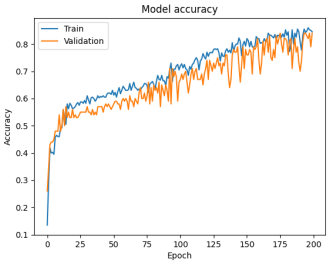
Figure 3.32: Loss vs. Epochs for different LSTM neuron sizes, with a fixed window size of 10.



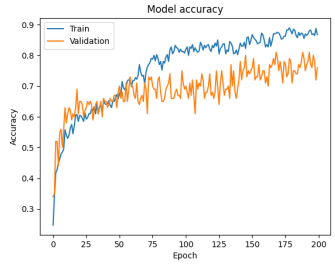
(a) LSTM with 4 neurons.



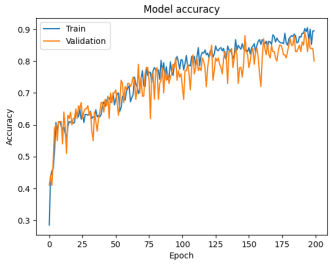
(b) LSTM with 8 neurons.



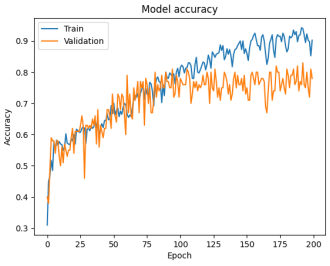
(c) LSTM with 16 neurons.



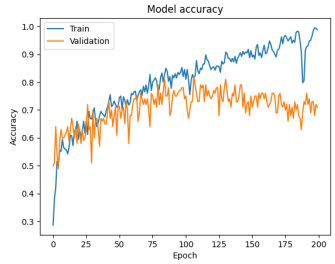
(d) LSTM with 32 neurons.



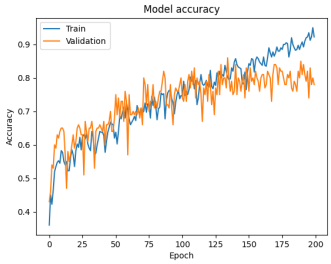
(e) LSTM with 64 neurons.



(f) LSTM with 128 neurons.



(g) LSTM with 256 neurons.



(h) LSTM with 512 neurons.

Figure 3.33: Accuracy vs. Epochs for different LSTM neuron sizes, with a fixed window size of 10.

The results of our experiments demonstrate that the LSTM-based approach consistently outperforms traditional algorithms across various ion channel types and datasets. This improvement is particularly noticeable when dealing with a large number of ion channels as evident in Fig 3.34. Also it's worth noting that as expected for fewer number of ion channels the accuracy is higher as evident in Figure 3.35

The superior performance of the LSTM can be attributed to its recurrent architecture, which allows it to maintain information about past inputs and use it to inform future predictions [67]. This is essential for understanding the complex dynamics of ion channels, which can exhibit long-term dependencies in their electrical activity.

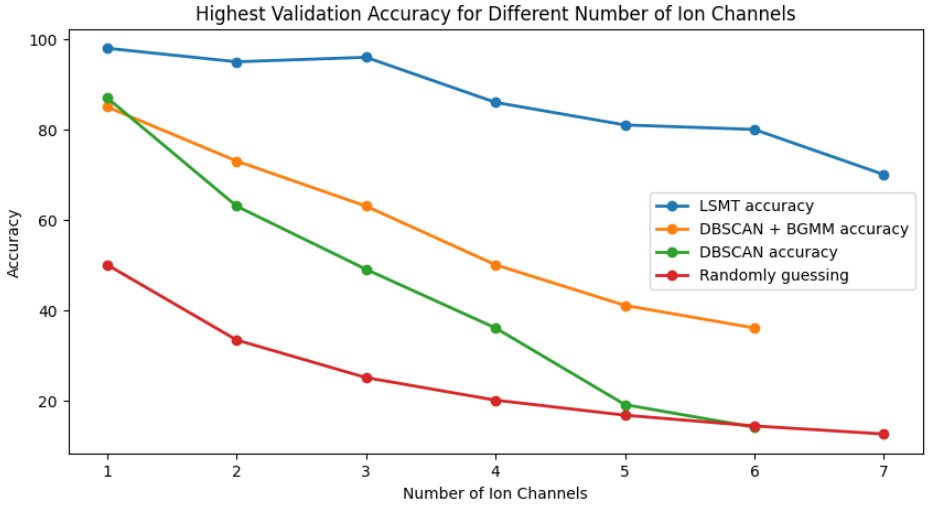


Figure 3.34: Figure demonstrates LSTM state classification accuracy on validation data with varying numbers of ion channels. The results highlight the LSTM's ability to effectively classify ion channel states, even when dealing with complex datasets containing a large number of ion channels.

3.3 Head-to-head Comparison on Synthetic Data

In this section, we perform a head-to-head comparison of the BGMM+DBSCAN approach and the LSTM model on synthetic data. We provide examples for one, two, and three ion channels. These examples effectively demonstrate the strengths and limitations of each algorithm.

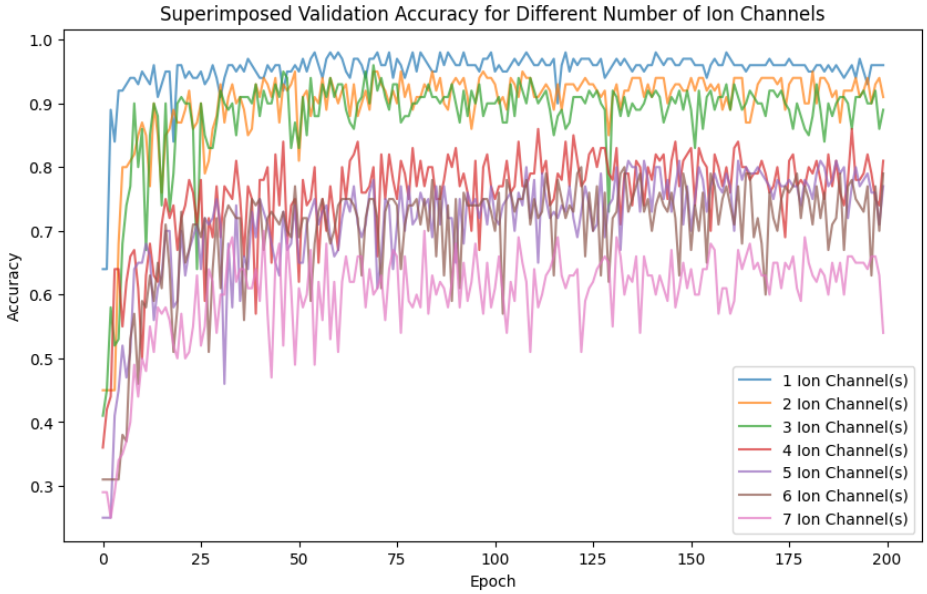
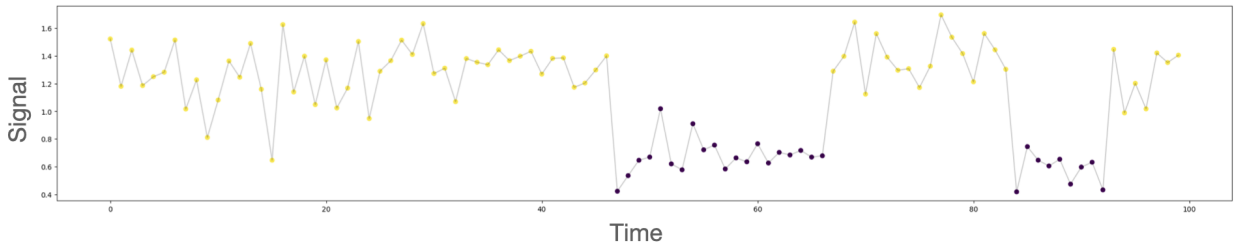


Figure 3.35: Validation accuracy across epochs for LSTM models with varying numbers of ion channels. The graph shows the performance of models with 1 to 7 ion channels over 200 epochs. Models with fewer ion channels generally achieve higher and more stable accuracy, while those with more channels exhibit lower and more volatile accuracy patterns.

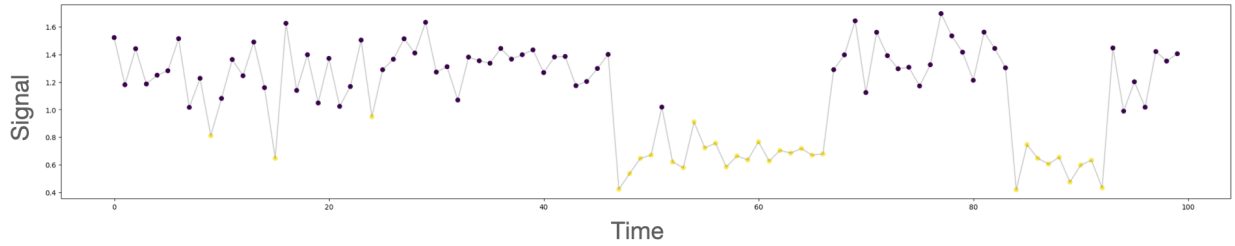
3.3.0.1 One Ion Channel

We start by showcasing two examples with one ion channel. For each example, we provide three subplots: the original labels, the classification results of BGMM+DBSCAN, and the classification results of the LSTM model. Note that the LSTM requires a window of data to make predictions, resulting in the first 10 and last 10 points remaining unclassified.

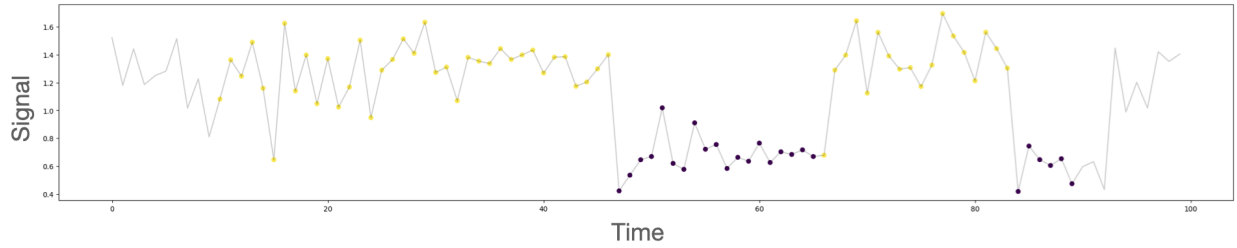
As seen in Figure 3.36 and Figure 3.37, when analyzing the examples of one ion channel, BGMM+DBSCAN struggles with handling noise spikes. In particular, during a spike of noise, BGMM+DBSCAN misclassifies the signal, interpreting it as a transition to a different state. On the other hand, the LSTM model, which takes the surrounding context into account, correctly identifies that this noise spike is still part of the same state. Additionally, it is important to note that the classification colors are not consistent between the different methods' graphs, though this does not affect the actual classification performance. The LSTM performed with 100% accuracy in this example, successfully classifying every point despite the noise.



(a) Correct labels for Example 1.

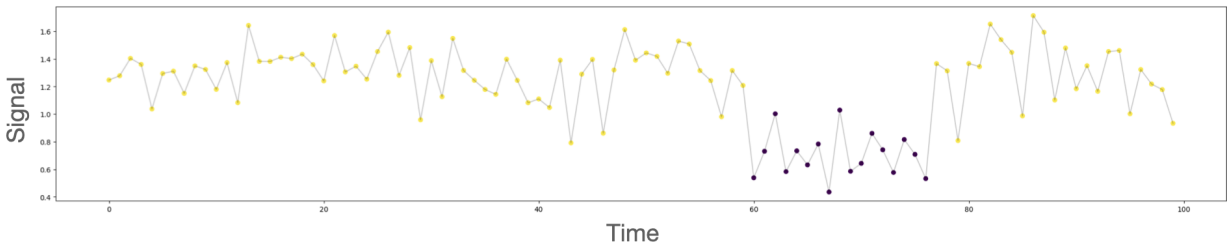


(b) BGMM+DBSCAN classification for Example 1.

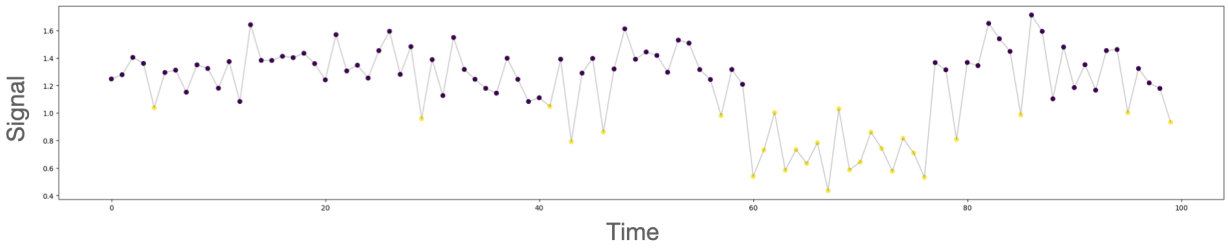


(c) LSTM classification for Example 1.

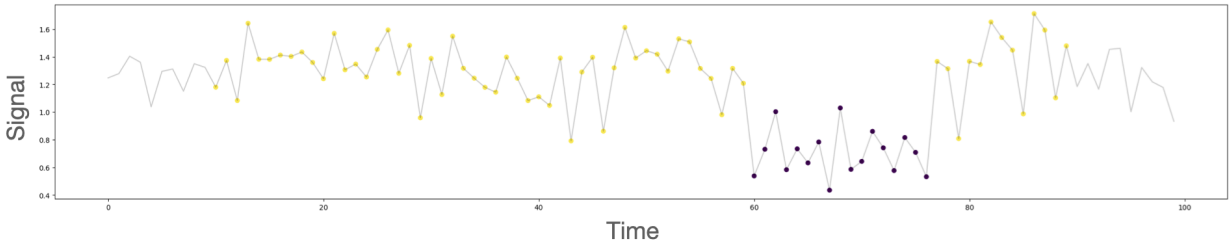
Figure 3.36: Example 1: Comparison between BGMM+DBSCAN and LSTM on one ion channel.



(a) Correct labels for Example 2.



(b) BGMM+DBSCAN classification for Example 2.



(c) LSTM classification for Example 2.

Figure 3.37: Example 2: Comparison between BGMM+DBSCAN and LSTM on one ion channel.

3.3.0.2 Two Ion Channels

Next, we present two examples with two ion channels. Similarly, we provide the original labels, BGMM+DBSCAN classification results, and LSTM classification results for each example.

In the two ion channels examples (Figures 3.38 and 3.39), it is evident that BGMM+DBSCAN struggles particularly with the middle states. These states are prone to misclassification, often being assigned to a neighboring state above or below, leading to inaccurate results. BGMM+DBSCAN has difficulty distinguishing these transitions and understanding the nuances of the signal's intermediate behavior. Moreover, in Figure 3.39, BGMM+DBSCAN even fails to correctly identify that there are two ion channels in the signal, further demonstrating its limitations.

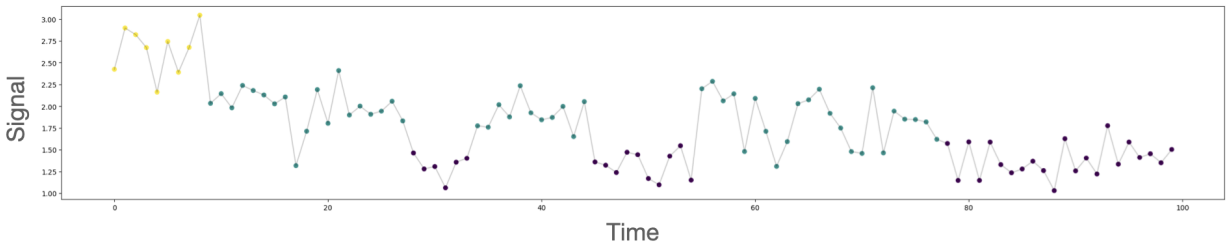
On the other hand, the LSTM model performs significantly better, accurately recognizing and respecting the middle states by assigning them their proper classification. The only issue the LSTM demonstrates is when transitioning into a new state, occasionally misclassifying the datapoint where the new state begins. Despite this minor issue, the LSTM consistently outperforms BGMM+DBSCAN in these examples.

3.3.0.3 Three Ion Channels

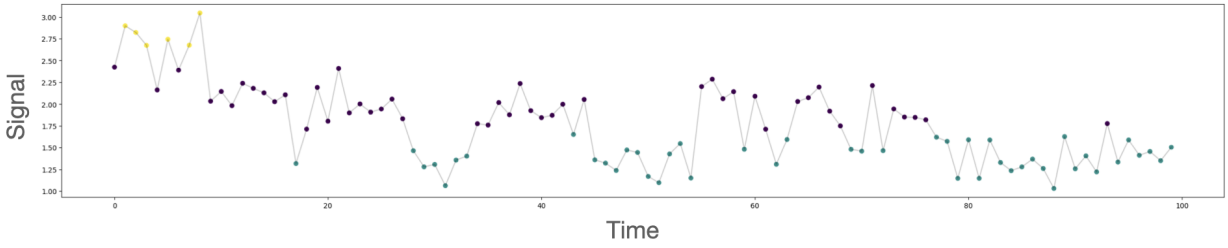
Lastly, we provide two examples with three ion channels to further illustrate the performance of both algorithms.

In the three ion channels examples (Figures 3.40 and 3.41), BGMM+DBSCAN once again demonstrates a high level of inconsistency in assigning the correct labels. It frequently misclassifies the data, failing to maintain any clear distinction between the different states of the ion channels. This inconsistency highlights the limitations of BGMM+DBSCAN as the number of ion channels increases.

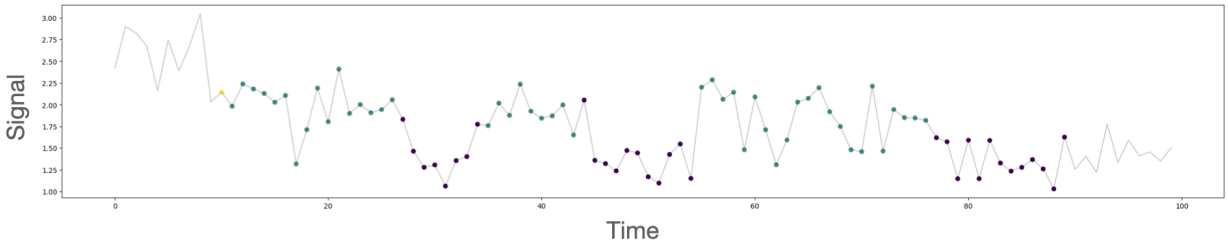
In contrast, the LSTM model performs very well in these cases, correctly identifying and classifying the majority of the signal states. However, the LSTM still shows a recurring issue with misclassifying the initial points of a new state, as it occasionally struggles to detect the exact moment of state transition. As observed in previous examples, the classification colors between different models remain inconsistent. Nonetheless, the LSTM continues to significantly outperform BGMM+DBSCAN in handling these complex signals.



(a) Original labels for Example 3.

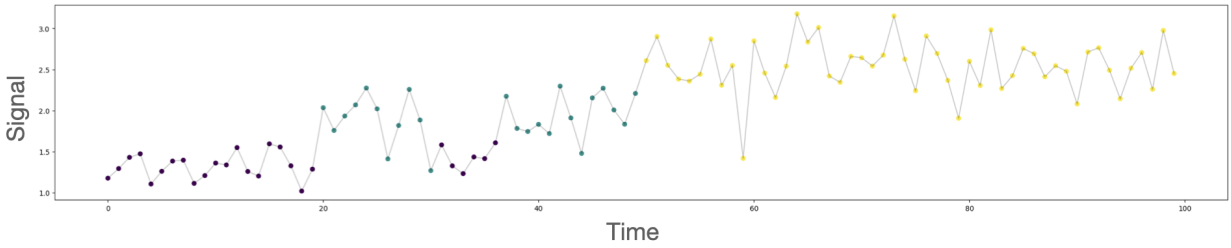


(b) BGMM+DBSCAN classification for Example 3.

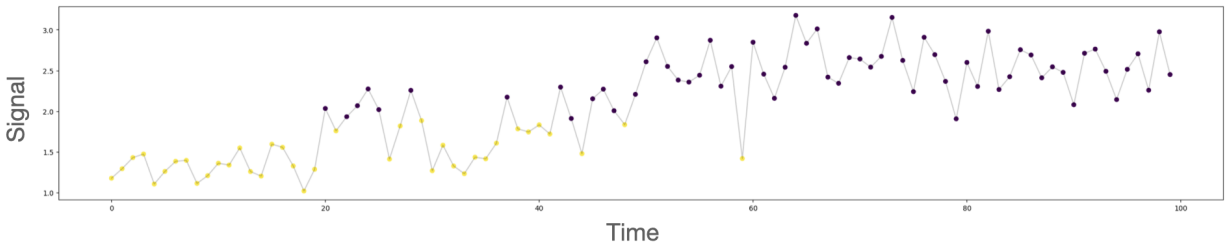


(c) LSTM classification for Example 3.

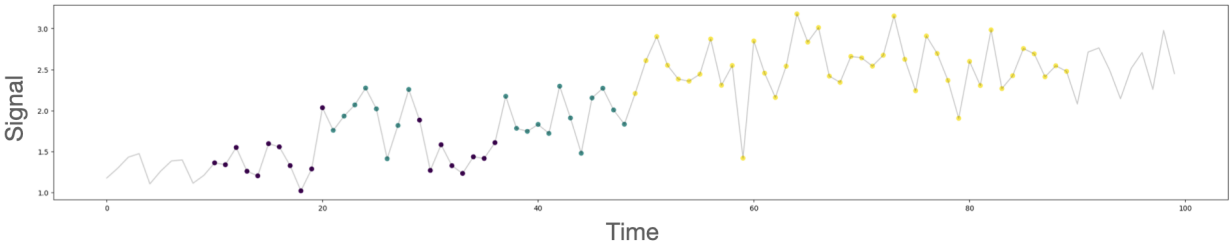
Figure 3.38: Example 3: Comparison between BGMM+DBSCAN and LSTM on two ion channels.



(a) Original labels for Example 4.

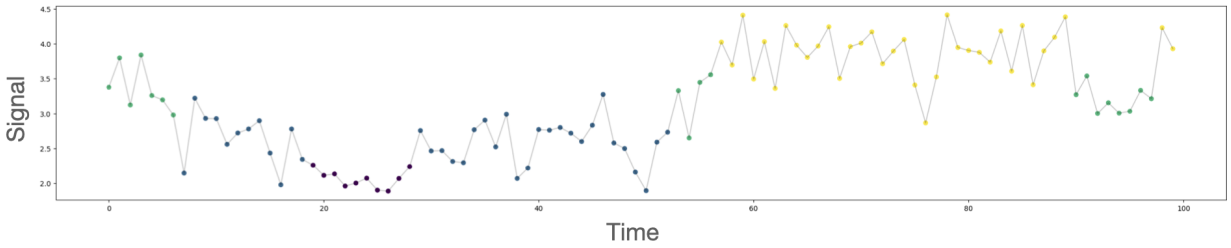


(b) BGMM+DBSCAN classification for Example 4.

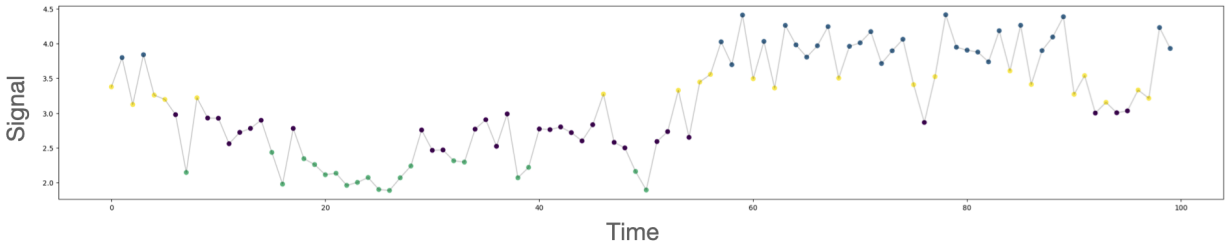


(c) LSTM classification for Example 4.

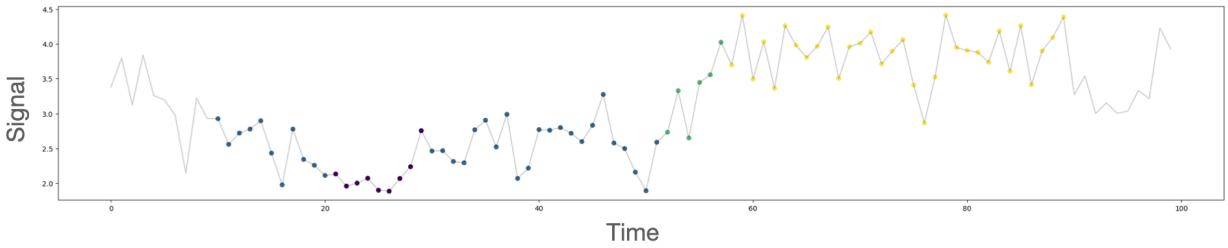
Figure 3.39: Example 4: Comparison between BGMM+DBSCAN and LSTM on two ion channels.



(a) Original labels for Example 5.

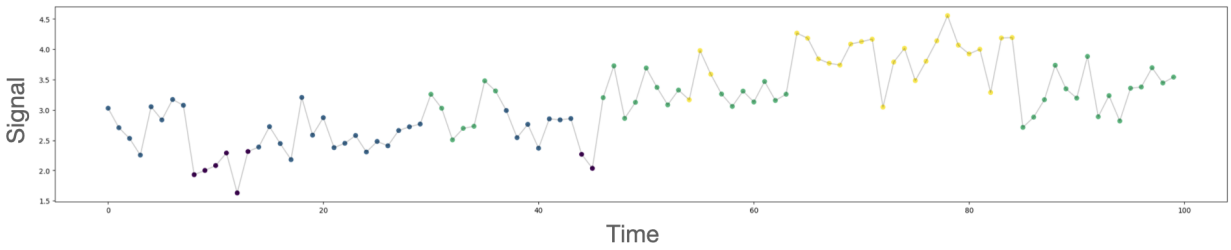


(b) BGMM+DBSCAN classification for Example 5.

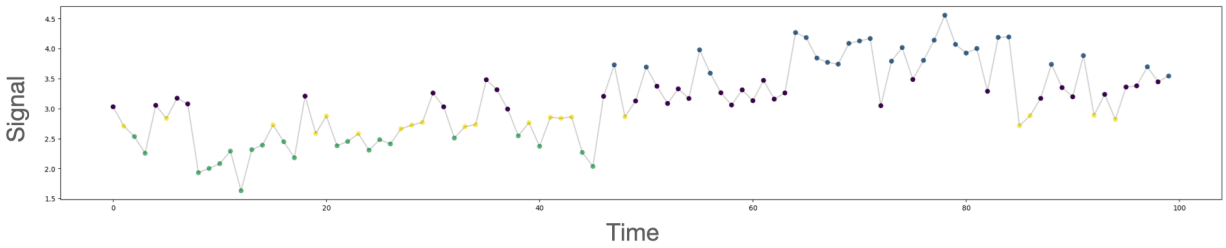


(c) LSTM classification for Example 5.

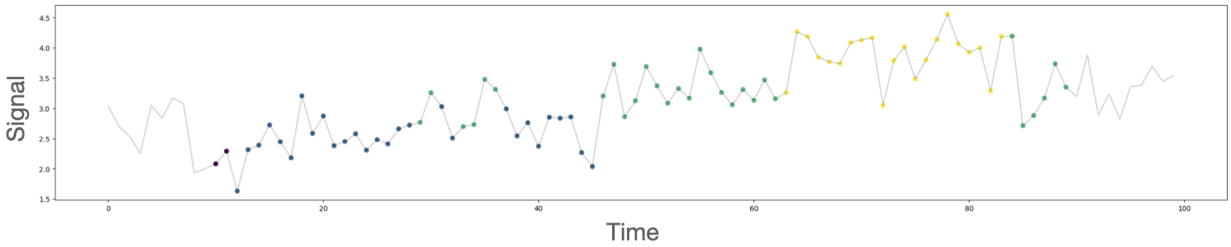
Figure 3.40: Example 5: Comparison between BGMM+DBSCAN and LSTM on three ion channels.



(a) Original labels for Example 6.



(b) BGMM+DBSCAN classification for Example 6.



(c) LSTM classification for Example 6.

Figure 3.41: Example 6: Comparison between BGMM+DBSCAN and LSTM on three ion channels.

3.3.0.4 Conclusion

In this section, we compared the performance of BGMM+DBSCAN and LSTM models for classifying synthetic ion channel data, focusing on scenarios with 1, 2, and 3 ion channels. The comparison highlights several key insights:

For the 1 ion channel case, BGMM+DBSCAN struggles with noise spikes, frequently misclassifying noisy data, while the LSTM model, leveraging its ability to analyze the context around the signal, performs flawlessly and provides a 100% accurate classification. This demonstrates the robustness of the LSTM model in dealing with noisy data.

When analyzing the 2 ion channel examples, BGMM+DBSCAN shows difficulties in identifying the correct states, particularly in middle states, often classifying points as belonging to adjacent states. It also fails to recognize that there are two distinct ion channels in one of the examples (Figure 3.39). On the other hand, the LSTM model correctly handles the middle states but struggles with the first few points during state transitions, occasionally misclassifying the initial points of new states. Despite this issue, the LSTM's performance is far superior to that of BGMM+DBSCAN.

Finally, for the 3 ion channel cases, BGMM+DBSCAN's performance deteriorates further, with significant inconsistency in assigning correct labels, making it unreliable in distinguishing between states. The LSTM model, while consistently outperforming BGMM+DBSCAN, continues to exhibit minor difficulties with state transitions but maintains a high level of accuracy overall.

In conclusion, while BGMM+DBSCAN can be a viable option for simpler cases with fewer ion channels, it becomes unreliable as the complexity of the signal increases. The LSTM model, by contrast, demonstrates strong performance across all cases, making it a far more robust choice for ion channel state classification, despite minor limitations in detecting precise state transitions. These findings align with our previous accuracy comparison between LSTM, BGMM+DBSCAN (hybrid), and DBSCAN models, as shown in Figure 3.42. The LSTM model consistently outperforms both DBSCAN and BGMM+DBSCAN, reinforcing its ability to accurately classify ion channel states, especially in more complex scenarios.

Overall, the results of the experiments are highly promising, especially for scenarios with multiple ion channels, where the LSTM model has consistently shown strong performance. Given that the LSTM can accurately handle state classification even in more complex datasets, the results

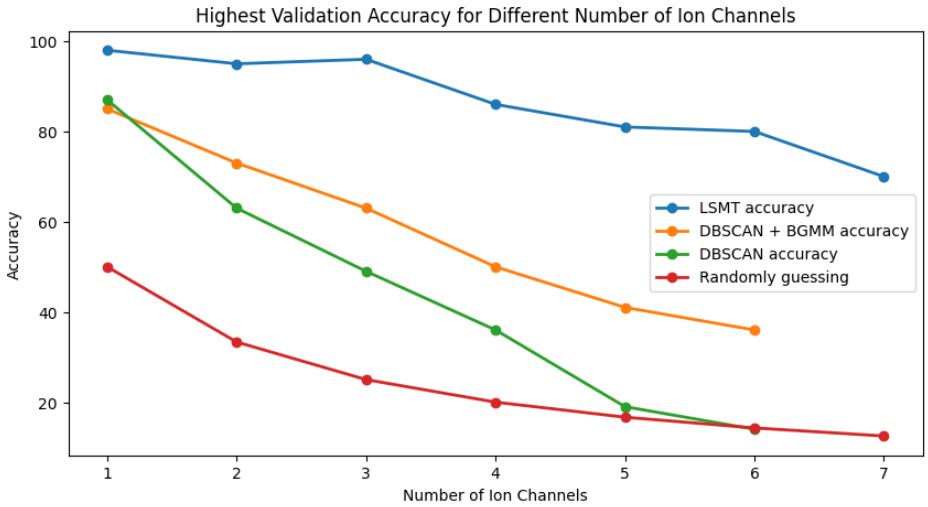


Figure 3.42: Figure demonstrates LSTM state classification accuracy on validation data with varying numbers of ion channels. The results highlight the LSTM's ability to effectively classify ion channel states, even when dealing with complex datasets containing a large number of ion channels.

would likely be satisfactory for a wide range of foreseen applications, including those involving noisy or multi-state ion channel signals.

One surprising aspect of the data is how well the LSTM model handles contextual information, particularly in cases where BGMM+DBSCAN failed to identify correct states or even distinguish between different ion channels. Despite the LSTM model's occasional issues with state transitions, it continues to excel in complex tasks, providing a reliable option for state classification in ion channels.

3.4 Related Work

While the presented NN-LSTM approach demonstrates promising results, several alternative methods exist for analyzing ion channel data. This section briefly outlines these alternative approaches, discussing their potential strengths and weaknesses and explaining why they may be less suitable or out of scope for the specific application of multi-channel CFTR patch-clamp analysis presented in this thesis.

3.4.1 Hidden Markov Models (HMMs)

HMMs are a widely used statistical approach for analyzing time-series data with hidden states, making them a natural candidate for ion channel analysis. They can model the probabilistic transitions between different channel states and infer the most likely state sequence given the observed current. However, traditional HMMs face challenges when applied to multi-channel recordings due to the increased complexity of state transitions and the difficulty in disentangling the activity of individual channels. While variations like factorial HMMs or coupled HMMs attempt to address multi-channel scenarios, they often involve computationally intensive training and parameter estimation, particularly as the number of channels increases [68].

3.4.2 Non-Negative Matrix Factorization (NMF)

NMF is a dimensionality reduction technique that can decompose a complex signal into a linear combination of basis functions. In ion channel analysis, NMF can be used to identify distinct channel states by representing the observed current as a weighted sum of basis currents corresponding to different open and closed states. However, NMF typically requires prior knowledge of the number of states, which may not be readily available in all experimental settings. Additionally, NMF does not explicitly model the temporal dynamics of channel transitions, which can be crucial for accurate state classification [69].

3.4.3 Independent Component Analysis (ICA)

ICA is another blind source separation technique used to decompose a mixed signal into its statistically independent source signals. In multi-channel recordings, ICA can potentially separate the activity of individual ion channels. However, ICA assumes that the source signals are linearly mixed and statistically independent, which may not hold true for ion channel data, where channels can exhibit correlated behaviour. Furthermore, ICA does not directly address the issue of state classification [70].

3.4.4 Wavelet Transform-Based Methods

Wavelet transforms offer time-frequency analysis capabilities, allowing for the detection of transient events like channel openings and closings. While wavelets can be effective in denoising and feature extraction, they do not directly address the challenge of clustering or classifying channel states, particularly in multi-channel scenarios. They often serve as a preprocessing step before applying other clustering algorithms [71].

3.4.5 Change-Point Detection Methods

Change-point detection algorithms aim to identify abrupt changes or transitions in time-series data. In the context of ion channel analysis, these methods can be used to detect the opening and closing events of channels. However, these methods primarily focus on detecting the timing of transitions rather than clustering similar states or estimating the number of channels [72].

3.4.6 State-Space Models

State-space models provide a general framework for modeling dynamical systems with hidden states. While powerful, fitting state-space models to ion channel data can be challenging due to the complex nature of channel gating and the presence of noise. Furthermore, they often require sophisticated parameter estimation techniques [73].

The proposed NN-LSTM approach addresses several of the limitations of these alternative methods. The NN effectively estimates the number of active channels, which is crucial information that many other methods require as input. The LSTM architecture explicitly models temporal dependencies, capturing the dynamic nature of ion channel transitions more accurately than methods like NMF or ICA. While HMMs also model temporal dependencies, the NN-LSTM approach offers greater flexibility and scalability, especially for multi-channel recordings, and avoids the computational burden of HMM parameter estimation. Finally, the use of simulated data allows for robust training and validation of the NN-LSTM model, overcoming the limitations of limited labeled data that often hinder the performance of supervised learning methods.

This research focuses specifically on multi-channel CFTR recordings. Some of the mentioned techniques, particularly single-channel analysis methods, are inherently out of scope for this ap-

plication. While some techniques could potentially be adapted or combined with the proposed approach, the NN-LSTM framework offers a compelling balance of accuracy, robustness, and computational efficiency for the targeted application. Future research could explore integrating some of these techniques, such as wavelet-based denoising or change-point detection, as preprocessing steps to further enhance the performance of the NN-LSTM model.

Chapter 4

Conclusion

4.1 Conclusion

In this thesis, we presented a comprehensive study on clustering ion channel activity using both classical and deep learning approaches. Our objective was to design, implement, and evaluate clustering algorithms that can accurately determine the number of active ion channels from patch clamp data. Throughout the study, we developed two primary approaches: classical clustering algorithms (DBSCAN and Bayesian Gaussian Mixture Model combined with DBSCAN) and machine learning models (Cluster Count Neural Network and Long Short-Term Memory network).

4.1.1 Summary of Findings

We began our analysis by investigating classical clustering methods, specifically focusing on DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and a hybrid approach involving Bayesian Gaussian Mixture Models (BGMM) combined with DBSCAN. These methods were evaluated on synthetic and real patch clamp data, which were generated by simulating ion channel behavior.

The DBSCAN algorithm demonstrated robustness in identifying clusters where the number of ion channels was relatively low, and noise was minimal. However, as the number of ion channels increased and noise levels became significant, the performance of DBSCAN alone declined. The introduction of BGMM to estimate the number of clusters before applying DBSCAN showed marked improvement in high-noise scenarios, yielding more accurate clustering results by mitigating the limitations of DBSCAN's density estimation.

Next, we explored machine learning models, starting with a Cluster Count Neural Network, which was designed to directly predict the number of active ion channels based on features derived from the patch clamp data. This model demonstrated remarkable efficiency in capturing the underlying trends and predicting ion channel counts with high accuracy. The neural network approach proved especially valuable in handling noisy data, where classical methods struggled.

Building on this, we introduced a Long Short-Term Memory (LSTM) network to incorporate temporal dependencies within the patch clamp data. By leveraging the sequential nature of the input signals, the LSTM model was able to learn time-dependent patterns and further refine the ion channel state predictions. The combination of the Cluster Count Neural Network and LSTM model

resulted in a highly effective solution, outperforming classical methods in both low and high-noise conditions.

4.1.2 Discussion of Results

Our results showed that classical clustering algorithms like DBSCAN, while effective in certain scenarios, are inherently limited in their ability to handle complex, noisy data. The BGMM+DBSCAN hybrid model partially addressed these challenges by introducing probabilistic clustering before density-based refinement. However, even this approach struggled in the presence of significant temporal dependencies in the data.

On the other hand, the machine learning models demonstrated superior performance across all conditions. The Cluster Count Neural Network effectively predicted the number of ion channels with minimal computational overhead, and the LSTM model further refined these predictions by incorporating the temporal characteristics of the data. This highlights the advantage of deep learning models, which can learn complex, non-linear relationships in the data that classical methods are unable to capture.

The training of the machine learning models was facilitated by the large amount of simulated data, which allowed us to fine-tune model parameters and achieve optimal performance. In particular, the choice of hyperparameters, such as the number of LSTM units, dense layer configurations, and learning rates, was crucial in maximizing the models' accuracy. These hyperparameters were carefully selected through cross-validation and detailed experimentation, as described in the results section.

4.1.3 Limitations and Future Work

Despite the promising results, several limitations in the current work point towards areas for future improvement. One major challenge is the generalization of the trained models to real-world data, which can differ significantly from the synthetic data used during training. Although we validated our models on real patch clamp data, further testing on a more diverse set of real-world conditions is necessary to fully evaluate the robustness of the machine learning models.

Another limitation is the computational cost associated with training deep learning models, particularly the LSTM network. The LSTM model, while effective, requires significant computa-

tional resources for training, especially when dealing with long time sequences. Future research could explore more efficient model architectures, such as convolutional neural networks (CNNs) or attention-based mechanisms like transformers, which could reduce the computational burden while maintaining or improving model performance.

Moreover, the models presented in this thesis are trained in a supervised fashion, meaning that they require labeled data. In practice, obtaining labeled patch clamp data can be difficult and time-consuming. Future work could explore unsupervised or semi-supervised learning techniques that would allow the models to learn from unlabeled or partially labeled data. This would be particularly useful for large-scale studies where manually labeling data is not feasible.

Another direction for future research could involve incorporating additional domain knowledge into the models. For instance, biophysical models of ion channel kinetics could be integrated into the deep learning framework, allowing the models to leverage known physiological constraints when making predictions. This could improve the interpretability and accuracy of the models, especially in cases where the data is noisy or incomplete.

Beyond these technical considerations, this research holds significant promise for clinical applications, particularly in the context of cystic fibrosis research. The crucial role of CFTR channel parameters in understanding disease mechanisms and evaluating the efficacy of drug therapies has been demonstrated. The clustering algorithms developed in this thesis offer a powerful new tool for analyzing the effect of drugs on CFTR channel activity in multi-channel recordings. By providing more accurate and robust estimations of channel parameters, these algorithms could enable more precise assessment of drug efficacy and personalized treatment strategies. Future work could explore incorporating these algorithms into drug development pipelines, facilitating the identification of novel therapeutic interventions for cystic fibrosis and other related conditions. This could involve analyzing patch-clamp data from patients with different CFTR mutations and assessing the impact of various drugs on their channel activity, ultimately leading to more targeted and effective treatments.

4.1.4 Conclusion

In conclusion, this thesis has demonstrated the utility of both classical clustering algorithms and machine learning models in the task of ion channel clustering. While classical methods like DB-

SCAN and BGMM+DBSCAN are useful in certain conditions, they are limited in their ability to handle noisy, complex data. In contrast, machine learning models, particularly the combination of the Cluster Count Neural Network and LSTM network, provide a powerful and flexible solution to this problem.

The findings of this thesis suggest that deep learning models offer significant advantages over classical methods, particularly in terms of handling noise and capturing temporal dependencies in the data. These models have the potential to revolutionize the analysis of ion channel data, offering new insights into ion channel behavior and enabling more accurate interpretations of patch clamp experiments.

Looking forward, the integration of more advanced machine learning techniques and the application of these models to larger, real-world datasets will be crucial steps in further advancing the field. With continued development and refinement, these models could provide valuable tools for researchers and clinicians studying ion channels, ultimately leading to a better understanding of their role in health and disease.

Bibliography

[1] "Patch clamp electrophysiology," May 2023.

[2] A. S. Moffett *et al.*, "Permissive and nonpermissive channel closings in cftr revealed by a factor graph inference algorithm," *Biophysical Reports*, vol. 2, no. 4, p. 100083, 2022.

[3] Z. Zeng, F. Jiang, Y. Yue, D. Han, L. Lin, S. Zhao, Y. bo Zhao, Z. Pan, C. Li, G. Nyström, and J. Wang, "Flexible and ultrathin waterproof cellular membranes based on high-conjunction metal-wrapped polymer nanofibers for electromagnetic interference shielding," *Advanced Materials*, vol. 32, 2020.

[4] R.-H. Fang, W. Gao, and L. Zhang, "Targeting drugs to tumours using cell membrane-coated nanoparticles," *Nature Reviews Clinical Oncology*, vol. 20, pp. 33–48, 2022.

[5] S. Lodetti, J. Bruna, and J. J. Melero, "Methods for the evaluation of new power quality parameters: A review of rapid voltage changes and supraharmonics," *undefined*, 2019.

[6] E. McCleskey and M. S. Gold, "Ion channels of nociception," *Annual Review of Physiology*, vol. 61, pp. 835–856, 2020.

[7] H. Zhang, X. Li, J. Hou, L. Jiang, and H. Wang, "Angstrom-scale ion channels towards single-ion selectivity," *Chemical Society Reviews*, 2022.

[8] D. T. Infield, K. M. Strickland, A. Gaggar, and N. A. McCarty, "The molecular evolution of function in the cftr chloride channel," *Journal of General Physiology*, vol. 153, no. 12, p. e202012625, 2021.

[9] C. G. Ziegler, S. J. Allon, S. K. Nyquist, I. M. Mbano, V. N. Miao, Y. Cao, X. Yuan, J. Xu, S. I. Liin, C. M. Finnerty, *et al.*, "Sars-cov-2 receptor ace2 is an interferon-stimulated gene

in human airway epithelial cells and is detected in specific cell subsets across tissues," *Cell*, vol. 181, no. 5, pp. 1016–1035, 2020.

[10] V. Scotet, C. L'Hostis, and C. Férec, "The changing epidemiology of cystic fibrosis: Incidence, survival and impact of the cftr gene discovery," *Genes*, vol. 11, no. 4, p. 439, 2020.

[11] L. Hanssens, J. Duchateau, and G. Casimir, "Cftr protein: Not just a chloride channel?," *Cells*, vol. 10, no. 12, p. 3576, 2021.

[12] M. Lopes-Pacheco, "Cftr modulators: The changing face of cystic fibrosis in the era of precision medicine," *Frontiers in Pharmacology*, vol. 10, p. 1662, 2020.

[13] J.-P. Mornon, B. Hoffmann, S. Jonić, P. Lehn, and I. Callebaut, "Full-open and closed cftr channels, with lateral tunnels from the cytoplasm and an alternative position of the f508 region, as revealed by molecular dynamics," *Cellular and Molecular Life Sciences*, vol. 72, pp. 1377–1403, 2015.

[14] Q. Han, Y. Yang, J. Zhang, and B. Dong, "Electrochemical characterization of chloride ion transport in rubber concrete," *Journal of Building Engineering*, 2023.

[15] K. Fiedorczuk and J. Chen, "Molecular structures reveal synergistic rescue of 508 cftr by trikafta modulators," *Science*, vol. 378, no. 6619, pp. 284–290, 2022.

[16] L. Jacobo-Albavera, M. Domínguez-Pérez, D. Medina-Leyte, A. González-Garrido, and T. Villarreal-Molina, "The role of the atp-binding cassette a1 (abca1) in human disease," *International Journal of Molecular Sciences*, vol. 22, no. 4, p. 1666, 2021.

[17] C. Bouysset and S. Fiorucci, "Prolif: a library to encode molecular interactions as fingerprints," *Journal of Cheminformatics*, vol. 13, no. 1, pp. 1–14, 2021.

[18] M. Tooley, "Electronics and biological signal processing," *BJA education*, vol. 23, no. 4, pp. 122–127, 2023.

[19] T. J. Cavicchi and U. Spagnolini, "Digital signal processing," in *8th International Multitopic Conference, 2004. Proceedings of INMIC 2004*, 2018.

[20] Y.-y. Chen *et al.*, "Characterizations of the urate transporter, glut9, and its potent inhibitors by patch-clamp technique," *SLAS Discovery*, vol. 26, pp. 450–459, 2020.

[21] D. H. Craighead *et al.*, "Time-efficient, high-resistance inspiratory muscle strength training for cardiovascular aging," *Experimental Gerontology*, vol. 154, 2021.

[22] A. U. Kokhan *et al.*, "Patch-clamp technique for studying ion channels in activated platelets," *System Biology and Physiology Reports*, 2021.

[23] P. D. W. Eckford, C. Li, M. Ramjeesingh, and C. Bear, "Cystic fibrosis transmembrane conductance regulator (cftr) potentiator vx-770 (ivacaftor) opens the defective channel gate of mutant cftr in a phosphorylation-dependent but atp-independent manner," *The Journal of Biological Chemistry*, vol. 287, pp. 36639–36649, 2012.

[24] M. Dean, K. Moitra, and R. Allikmets, "The human atp-binding cassette (abc) transporter superfamily," *Human Mutation*, vol. 43, pp. 1162–1182, 2022.

[25] L. Moesgaard, M. L. Pedersen, C. U. Nielsen, and J. Kongsted, "Structure-based discovery of novel p-glycoprotein inhibitors targeting the nucleotide binding domains," *Scientific Reports*, vol. 13, 2023.

[26] G. Veit, A. Roldán, M. Hancock, D. F. Da Fonte, H. Xu, M. H. Hussein, *et al.*, "Allosteric folding correction of f508del and rare cftr mutants by elexacaftor-tezacaftor-ivacaftor (trikafta) combination," *JCI Insight*, vol. 5, 2020.

[27] S. Bickers, S. Benlekbir, J. Rubinstein, and V. Kanelis, "Structure of ycf1p reveals the transmembrane domain tmd0 and the regulatory region of abcc transporters," *Proceedings of the National Academy of Sciences*, vol. 118, 2021.

[28] D. J. Ramms, F. Raimondi, N. Arang, F. Herberg, S. S. Taylor, and J. Gutkind, "Gs–protein kinase a (pka) pathway signalopathies: The emerging genetic landscape and therapeutic potential of human diseases driven by aberrant gs-pka signaling," *Pharmacological Reviews*, vol. 73, 2021.

[29] X. Chen, "Regional disturbances by the [delta]f508 mutation on the cystic fibrosis transmembrane conductance regulator," 2016.

[30] K. Locher, "Mechanistic diversity in atp-binding cassette (abc) transporters," *Nature Structural & Molecular Biology*, vol. 23, pp. 487–493, 2016.

[31] C. Ng, J. Farr, P. E. Young, M. Windley, M. Perry, A. Hill, and J. Vandenberg, "Heterozygous kcnh2 variant phenotyping using flp-in hek293 and high-throughput automated patch clamp electrophysiology," *Biology Methods Protocols*, vol. 6, 2021.

[32] Y. Yamamoto, S. Hirose, Y. Wuriyanghai, D. Yoshinaga, and T. Makiyama, "Electrophysiological analysis of hipsc-derived cardiomyocytes using a patch-clamp technique," *Methods in Molecular Biology*, vol. 2320, pp. 121–133, 2021.

[33] J. Nowicka-Zagrajek and R. Weron, "Modeling electricity loads in california: Arma models with hyperbolic noise," *Signal Processing*, vol. 82, no. 12, pp. 1903–1915, 2002.

[34] S. Chowdhury, N. Helian, and R. C. d. Amorim, "Feature weighting in dbscan using reverse nearest neighbours," *Pattern Recognition*, vol. 137, p. 109314, 2023.

[35] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise," *ArXiv*, vol. abs/1706.03825, 2017.

[36] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D.-k. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations (ICLR)*, 2018.

[37] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. J. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. A. Cuadros, R. Kim, R. Raman, P. Nelson, J. Mega, and D. Webster, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Journal of the American Medical Association (JAMA)*, vol. 316, no. 22, 2016.

[38] P. Li, X. He, M. Qiao, X. Cheng, J. Li, X. Guo, T. Zhou, D. Song, M. Chen, D. Miao, Y. Jiang, and Z. Tian, "Exploring label probability sequence to robustly learn deep convolutional neural networks for road extraction with noisy datasets," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–18, 2022.

[39] X. Bai, Z. Xie, X. Xu, and Y. Xiao, "An adaptive threshold fast dbscan algorithm with preserved trajectory feature points for vessel trajectory clustering," *Ocean Engineering*, 2023.

[40] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *Proceedings of the ACM Web Conference 2022*, 2022.

[41] M. Miller, F. Lamb, A. Dittmann, S. Bogdanov, Z. Arzoumanian, K. Gendreau, S. Guillot, W. Ho, J. Lattimer, M. Loewenstein, S. Morsink, P. Ray, M. Wolff, C. Baker, T. Cazeau, S. Manthripragada, C. Markwardt, T. Okajima, S. Pollard, I. Cognard, H. Cromartie, E. Fonseca, L. Guillemot, M. Kerr, A. Parthasarathy, T. Pennucci, S. Ransom, and I. Stairs, "The radius of psr j0740+6620 from nicer and xmm-newton data," *The Astrophysical Journal Letters*, vol. 918, 2021.

[42] Z. Zhou, G. Si, H. Sun, K. Qu, and W. Hou, "A robust clustering algorithm based on the identification of core points and knn kernel density estimation," *Expert Systems with Applications*, vol. 195, p. 116573, 2022.

[43] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[44] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, "Hoeffding trees with nmin adaptation," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 70–79, 2018.

[45] C. Wang, H. Liu, C. Li, Y. Sun, W. Wang, and B. Wang, "Robust intrusion detection for industrial control systems using improved autoencoder and bayesian gaussian mixture model," *Mathematics*, 2023.

[46] S. Chhabra, H. Venkateswara, and B. Li, "Generative alignment of posterior probabilities for source-free domain adaptation," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4114–4123, 2023.

[47] F. Khoroshevsky, S. Khoroshevsky, and A. Bar-Hillel, "Parts-per-object count in agricultural images: Solving phenotyping problems via a single deep neural network," *Remote Sensing*, vol. 13, p. 2496, 2021.

[48] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Neural Information Processing Systems*, pp. 802–810, 2015.

[49] C. Stoean, M. Zivkovic, A. Bozovic, N. Bačanin, R. Strulak-Wójcikiewicz, M. Antonijevic, and R. Stoean, "Metaheuristic-based hyperparameter tuning for recurrent deep learning: Application to the prediction of solar energy generation," *Axioms*, vol. 12, p. 266, 2023.

[50] P. Singh, P. Sahoo, D. Fujita, A. Bandyopadhyay, K. Ray, and S. Ghosh, "Reducing the dimension of a patch-clamp to the smallest physical limit using a coaxial atom probe," *Progress in Electromagnetics Research B*, vol. 89, pp. 29–44, 2020.

[51] H. Zhang, Z. Mo, J. Wang, and Q. Miao, "Nonlinear-drifted fractional brownian motion with multiple hidden state variables for remaining useful life prediction of lithium-ion batteries," *IEEE Transactions on Reliability*, vol. 69, pp. 768–780, 2020.

[52] F. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size," *arXiv preprint arXiv:1602.07360*, vol. abs/1602.07360, 2016.

[53] W. Falah and I. J. Mohammed, "Hybrid cnn-smote-bgmm deep learning framework for network intrusion detection using unbalanced dataset," *Iraqi Journal of Science*, 2023.

[54] P. Nanayakkara, M. A. Smart, R. Cummings, G. Kaptchuk, and E. M. Redmiles, "What are the chances? explaining the epsilon parameter in differential privacy," *arXiv preprint arXiv:2303.00738*, vol. abs/2303.00738, 2023.

[55] A. Greco, G. Valenza, J. Lázaro, J. M. Garzón-Rey, J. Aguiló, C. de la Cámara, R. Bailón, and E. Scilingo, "Acute stress state classification based on electrodermal activity modeling," *IEEE Transactions on Affective Computing*, vol. 14, pp. 788–799, 2023.

[56] K.-J. Chen, Y.-K. Chuang, B.-Y. Yu, and S.-Y. Fang, "Minimizing cluster number with clip shifting in hotspot pattern classification," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[57] R. Masuda and R. Inoue, "Point event cluster detection via the bayesian generalized fused lasso," *ISPRS International Journal of Geo-Information*, vol. 11, p. 187, 2022.

[58] S. Aibar, C. B. González-Blas, T. Moerman, V. A. Huynh-Thu, H. Imrichová, G. J. Hulselmans, F. Rambow, J. Marine, P. Geurts, J. Aerts, J. van den Oord, Z. Atak, J. Wouters, and S. Aerts, "Scenic: Single-cell regulatory network inference and clustering," *Nature Methods*, vol. 14, pp. 1083–1086, 2017.

[59] K. Blin, S. Shaw, A. M. Kloosterman, Z. Charlop-Powers, G. P. van Wezel, M. Medema, and T. Weber, "antismash 6.0: improving cluster detection and comparison capabilities," *Nucleic Acids Research*, vol. 49, pp. W29–W35, 2021.

[60] J. Šimaitis, S. Allen, and C. Vagg, "Are future recycling benefits misleading? prospective life cycle assessment of lithium-ion batteries," *Journal of Industrial Ecology*, vol. 27, pp. 1291–1303, 2023.

[61] M. T. Jatipaningrum, S. E. Azhari, and K. Suryowati, "Pengelompokan kabupaten dan kota di provinsi jawa timur berdasarkan tingkat kesejahteraan dengan metode k-means dan density-based spatial clustering of applications with noise," *Jurnal Derivat: Jurnal Matematika dan Pendidikan Matematika*, 2022.

[62] A. P, A. Sharma, A. Singla, N. Sharma, and D. Gowda V, "Iot group key management using incremental gaussian mixture model," in *2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 469–474, 2022.

[63] I. Szabò and A. Szewczyk, "Mitochondrial ion channels," *Annual Review of Biophysics*, vol. 52, 2023.

[64] G. Li, X. Wang, R. Hu, H. Zhang, and S. Ke, "Normal-to-lombard speech conversion by lstm network and bgmm for intelligibility enhancement of telephone speech," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2020.

[65] N. K. Thapa K and N. Duraipandian, "Malicious traffic classification using long short-term memory (lstm) model," *Wireless Personal Communications*, vol. 119, pp. 2707–2724, 2021.

[66] K. Greff, R. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2222–2232, 2015.

[67] L. Cheng, H. Zang, Y. Xu, Z.-n. Wei, and G.-q. Sun, "Augmented convolutional network for wind power prediction: A new recurrent architecture design with spatial-temporal image inputs," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 6981–6993, 2021.

[68] L. Venkataramanan *et al.*, "Applying hidden markov models to the analysis of single ion channel activity," *Biophysical Journal*, vol. 82, no. 4, pp. 1930–1942, 2002.

[69] L. Xu, M. E. Chavez-Echeagaray, and V. Berisha, "Unsupervised eeg channel selection based on nonnegative matrix factorization," *Biomedical Signal Processing and Control*, vol. 76, p. 103700, 2022.

[70] A. Hyvärinen, *Topographic Independent Component Analysis*, pp. 3446–3449. New York, NY: Springer New York, 2022.

[71] Z. Chen, X. Guo, X. Zhang, W. Cram, and Z. Li, "A novel method for analysis of single ion channel signal based on wavelet transform," *Computers in Biology and Medicine*, vol. 37, no. 4, pp. 559–562, 2007. Wavelet-based Algorithms for Medical Problems.

[72] A. Y. Kaplan and S. L. Shishkin, *Application of the Change-Point Analysis to the Investigation of the Brain's Electrical Activity*, vol. 509. Springer Science+Business Media, 2000.

[73] L. Paninski, Y. Ahmadian, D. G. Ferreira, S. Koyama, K. Rahnama Rad, M. Vidne, J. Vogelstein, and W. Wu, "A new look at state-space models for neural data," *Journal of computational neuroscience*, vol. 29, pp. 107–126, 2010.