UNIVERSITY OF BIRMINGHAM

# Enhanced Sensing and Electronic Characterisation of Single-Molecules using Machine Learning

By

**Christopher Dennis Weaver**

**A thesis submitted to the University of Birmingham for the degree of**

**DOCTOR OF PHILOSOPHY**

School of Chemistry

College of Engineering and Physical Sciences

University of Birmingham

United Kingdom

September 2023

# Abstract

Due to the complex and inherently stochastic nature of single-molecule science, experimental measurements typically need to produce large datasets to be able to infer meaningful conclusions regarding the properties of single molecules. Due to both the large volume of data and the highly complex nature of single-molecule systems, experimental results are notoriously difficult to analyse effectively by traditional means. Typically, statistical techniques, such as hypothesis testing, are employed such that the average behaviour of single molecules can be inferred. However, these approaches can be ill-suited to these datasets as assumptions are made to allow for statistical inference. These assumptions can result in limited predictive accuracies due to information loss that coincides with holding potentially misleading prior expectations. As an alternative to statistical methods, the techniques provided by the field of machine learning might be better suited to data analysis of complex datasets as they can yield predictions that do not rely on the presence of assumptions.

Therefore, this thesis explores the benefits of the application of machine learning algorithms to the analysis of single-molecule datasets. To this end, scanning tunnelling spectroscopy techniques were implemented to measure the electrical properties of single molecules and acquire challenging datasets. From the desire to measure single molecules, a new current-time technique workflow was developed whereby current-distance spectroscopy was used to experimentally determine the tunnelling decay constant in analyte solutions. This tunnelling decay constant was subsequently used to construct a calibration curve that relates any measured tunnelling current to the corresponding tunnelling distance. It has been

demonstrated that this calibration allows for a tunnelling distance to be optimized for a given molecules physical dimensions, and result in reliable detection of current-time binding events when using an STM in constant-current mode.

Complementary to the development of a reliable current-time technique workflow, advances were made regarding the detection of anomalies in current-time traces. The techniques tested include a simple moving average, a zero-crossing rate detector, and a convolutional autoencoder. It has been demonstrated that current-time events can be detected within a trace with an area under receiver operating characteristic score of 0.97 at a signal-to-noise ratio of only 0.1 when employing a moving average. Whilst exploring the optimisation of these anomaly detection methods it was also discovered that the window size used for each technique directly influences the width of baseline and event distributions as per the standard error of means. As a result, anomaly detection at lower signal-to-noise ratios requires sufficient temporal resolution such that larger window sizes can be implemented. Because of this, it was summarised that effective detection of I(t) events requires high sampling frequencies.

In previous studies, the current-time technique has been extensively applied to the detection of DNA nucleotides for the trialling of a potential next generation sequencing technology. Here, an additional limitation of this technique was revealed where different molecular species produce current-time events of similar appearance. Indeed, the traditionally measured properties of these events demonstrate considerable overlap which makes effective classification not possible. However, it has been demonstrated here that a dataset that possesses the same overlap limitation can be classified with an accuracy of 95%.

Implementations of convolutional neural networks and random forest classifiers have been able to classify these complex datasets when traditional statistical means cannot.

As a side to the enhancements made with the current-time technique, machine learning techniques have also been demonstrated to enhance the analysis within scanning tunnelling microscopy break junction experiments through implementation of the dimensionality reduction techniques principal component analysis, t-distributed stochastic neighbour embedding, and uniform manifold approximation and projection. To ease the cleaning and segmentation of experimental data, dimensionality reduction was used such that trends in the overall shapes of traces could be visualised in low-dimensional space. It has been shown that traces with similar shapes are located adjacent to one another in these embedded spaces which facilitated the clustering of traces. To demonstrate the versatility of this method, dimensionality reduction was also applied to a cyclic voltammetry dataset where the differences in complex voltametric curves could be easily visualised and related to experimental parameters.

Stemming from the development of a machine learning enhanced analysis process of break junctions, a novel pseudo-rotaxane molecule has been characterised. From studying this molecule, multiple molecular conductance values of -1.25, -1.96, and -3.08 $\log(G/G_0)$ have been observed. All of these values are far higher than one would expect given the molecules length and lack of conjugation. This highlights rotaxane-like molecules, in general, as potentially highly conductive molecular wires. However, whilst this molecule was shown to possess high conductivity, it also possessed a large variety of plateau break-off distances which is believed to result from strong sulphur-gold interactions. To address these

displacement variations a novel alignment procedure was developed. Here, change-point detection strategies from the field of anomaly detection were implemented such that the displacement at which molecular plateaus start could be found on a trace-by-trace basis.

Overall, the implementation of machine learning techniques has been shown to improve the analysis processes of single-molecule scanning tunnelling microscopy techniques, such that the previously challenging classification of current-time events has been achieved with relative ease. Additionally, machine learning has enhanced the characterisation of single molecules for nanoscale electronics such that promising molecules have been highlighted.

# Acknowledgements

I would like to take this opportunity to offer thanks to all the individuals around me that helped me throughout my postgraduate studies.

Firstly, I would like to thank my supervisor, Professor Tim Albrecht. Thank you for the opportunity to continue my academic career through my undergraduate masters and into my postgraduate research. Thank you for your constant guidance and patience that you have provided throughout my studies. Without your encouragement, I would have never been able to pursue my passion for data science and machine learning.

Next, I would like to thank all of my colleagues within the Albrecht group. I would like to thank Dr Joseph Hamill for patiently getting me through the challenging process of learning to operate an STM. Without your encouragement, I would have not been able to overcome the overwhelming number of intricate details required for effective use of such a sensitive instrument. To Dr Adrian Fortuin I would like to thank you for helping me get to grips with electrochemistry and more complex mathematics. I would like to thank Dr Oliver Irving for your constant support and ever uplifting attitude. Your encouragement has seen me through the more difficult moments of this journey. To Nashwa Awais, I wish to offer thanks for always being there and distracting me from the many failed experiments with talks about a certain excellent video game. To Lauren Matthews, I would like to thank you for you constant high spirits that has helped make my time in the group an enjoyable one. Lastly, to Jake, Cengiz, and Rand. Although the three of you joined the group near the end of my course, for the time that I have known you, you have all been highly encouraging and supportive. All of you have

been wonderful individuals that have made my time within the Albrecht group an enjoyable one.

To my friends outside of the university. Thank you all for always being there and looking out for me.

To my family I would like to offer a large thank you. To my parents, Stephen and Vivienne Weaver. Thank you always believing in me and encouraging me to be the best that I can be. You have always provided me with all the opportunities possible such that I have been able to grow and prosper with few limitations. Thank you for everything you have done throughout the years.

Lastly, I would like to thank my wonderful girlfriend, Charlotte Cox. Throughout the five years that we have been together, you have always been supportive of me and selflessly caring. Your constant encouragement has seen me grow into a more mature person that I am happy to be now. Without your kind words and occasional tough love, I would have never started this PhD let alone finish it. For everything you have done for me, thank you.

# List of Publications

- "Multivariate Approach to Single-Molecule Thermopower and Electrical Conductance Measurements", J. M. Hamill, C. Weaver and T. Albrecht, *The Journal of Physical Chemistry C*, 2021, **125**, 26256–26262.

- "Unsupervised classification of voltammetric data beyond principal component analysis", C. Weaver, A. C. Fortuin, A. Vladyka and T. Albrecht, *Chemical Communications*, 2022, **58**, 10170–10173.

# Table of Contents

## Contents

# Table of Figures

# Table of Tables

# Table of Abbreviations

| Abbreviation | Definition |
|---|---|
| **Adam** | Adaptive Moment Estimation |
| **AE** | Autoencoder |
| **AI** | Artificial Intelligence |
| **AUC** | Area Under the Curve |
| **Bagging** | Bootstrap Aggregating |
| **BJ** | Break Junction |
| **CNN** | Convolutional Neural Network |
| **CoD** | Curse of Dimensionality |
| **CV** | Cyclic Voltammetry/Cyclic Voltammogram |
| **DL** | Deep Learning |
| **DNA** | Deoxyribose Nucleic Acid |
| **DR** | Dimensionality Reduction |
| **E** | Electron Transfer |
| **EC** | Electron Transfer-Chemical Reaction Coupling |
| **ECE** | Electron Transfer-Chemical Reaction-Electron Transfer Coupling |
| **ECSTM** | Electrochemical Scanning Tunnelling Microscopy |
| **E-tongue** | Electrochemical Tongue |
| **EVA** | ethylene-vinyl acetate |
| **FFNN** | Feed-forward Neural Network |
| **FN** | Fowler-Nordheim |
| **FNR** | False Negative Rate |
| **FPR** | False Positive Rate |
| **I(s)** | Current-Distance |
| **I(t)** | Current-Time |
| **KL** | Kullback-Leibler |
| **KNN** | k-Nearest Neighbour |
| **MAE** | Mean Absolute Error |
| **MCBJ** | Mechanically Controlled Break Junction |
| **MCC** | Matthews Correlation Coefficient |
| **ML** | Machine Learning |
| **MLP** | Multi-layer Perceptron |
| **MPVC** | Multiparameter Vector-based Classification |
| **MSE** | Mean Squared Error |
| **NBC** | Naïve Bayesian Classifier |
| **NEGF** | Non-Equilibrium Green's Function |
| **PCA** | Principal Component Analysis |
| **PID** | Proportional-Integral-Derivative |
| **rAMP** | Ribose Adenosine Monophosphate |
| **rCMP** | Ribose Cytidine Monophosphate |
| **ReLU** | Rectified Linear Unit |
| **RFC** | Random Forest Classifier |
| **RGB** | Red-Green-Blue |

| | |
|---|---|
| **rGMP** | Ribose Guanosine Monophosphate |
| **RMS** | Root Mean Square |
| **RNN** | Recurrent Neural Network |
| **ROC** | Receiver Operating Characteristic |
| **ROI** | Region Of Interest |
| **RPS** | Resistive Pulse Sensing |
| **rTMP** | Ribose Thymidine Monophosphate |
| **SCC** | Sparse Categorical Crossentropy |
| **SGD** | Stochastic Gradient Descent |
| **SM** | Single-Molecule |
| **SNE** | Stochastic Neighbour Embedding |
| **SNR** | Signal-to-Noise Ratio |
| **STM** | Scanning Tunnelling Microscopy |
| **STM-BJ** | Scanning Tunnelling Microscopy Break Junction |
| **SVM** | Support Vector Machine |
| **TL** | Transfer Learning |
| **TNR** | True Negative Rate |
| **TPR** | True Positive Rate |
| **t-SNE** | T-Distributed Stochastic Neighbour Embedding |
| **UMAP** | Uniform Manifold Approximation and Projection |
| **XOR** | Exclusive OR |
| **ZCR** | Zero-Crossing Rate |

# Chapter 1

## Introduction to Machine Learning in Chemistry

Contents

## 1.1. Single-Molecule Science

Single-Molecule (SM) science is a field of study that involves the isolation and measurement of the properties of individual molecules.[1] The ability to capture and measure individual molecules has many applications. For instance, measurement of molecular properties within an ensemble would provide an insight into the types of molecules present. Being able to distinguish the molecules present in a system would make improvements within a biosensing setting where the detection of certain markers would allow effective diagnosis and better insights into the function of biological systems.[2] Another example of SM applications is in the field of nanoelectronics.[3] Here, the ability to measure the electrical properties of specially designed molecules is invaluable. Accurate characterisation of conductive molecules would aid in the development of novel electrical components with smaller component sizes. Ultimately, this would allow for the development of more powerful electrical devices as reducing the size of components in electrical circuits will result in more compact modules that will allow for such devices to possess more processing power and storage than previous. For instance, the size of transistors has reduced considerably which has allowed for the incorporation of more transistors within an individual device.[4,5] Given the ever-decreasing size of these electronic components, researchers now aim to investigate the use of individual molecules as SM electrical components.

However, due to the small size of single molecules, their behaviour cannot be measured with high degrees of certainty. As single molecules are small, their behaviour is greatly influenced by multiple factors. These include aspects such as temperature, pressure, electromagnetic fields, and molecular collisions meaning that unbound molecules are constantly in motion.[6] When capturing these molecules, this means it is difficult to predict their orientation, and how

effectively molecules have been captured. There is also no way to pre-emptively know whether a captured molecule is the desired subject or an unwanted contaminant. This, while less likely, can still occur when studying the purest of environments. In addition to these environmental factors there is also uncertainty caused by the measuring equipment. Due to the small nature of single molecules, the measuring equipment needs to be highly sensitive with large sampling resolutions. Because of this sensitivity more care is required with regards to equipment operation and calibration as even the slightest deviations in equipment parameters can cause substantial error in measurement.[7,8] Lastly, there is also natural variation in molecular properties depending on the property being measured, with this intrinsic uncertainty stemming from the rules of quantum physics. Because of these environmental, instrumental, and intrinsic factors, SM experiments require a large number of repeat measurements to analyse molecular properties more reliably. As such, this field can generate very large amounts of data.[9] This comes with its own challenges as effective treatment of large datasets is then required. These challenges are showcased within the following example techniques.

### 1.1.1. Scanning Tunnelling Microscopy

One of the more common techniques used within SM science is Scanning Tunnelling Microscopy (STM). In 1983, Binnig and Rohrer first devised the STM as an instrument that can measure tunnelling currents across a nanoscale gap. Conventionally, an STM is used to image conductive surfaces with nanoscale spatial resolution using these tunnelling currents. To do this, STM translates the quantum tunnelling current of electrons into topographical changes in a conductive surface.[10,11] For more details on how STMs function, and the quantum tunnelling effects involved, the interested reader is referred to Section 2.1. However, in SM

science STM can additionally be used to produce non-image data that results from direct measurement of molecular properties via implementation of the following techniques.

### *Current-Distance Spectroscopy*

One such example is the current-distance (I(s)) spectroscopy technique. Here, I(s) techniques are employed to both trap and measure single molecules for the purpose of electrical characterisation. Overall, these techniques have been used as an effective test bed for analysing the electrical properties of single molecules.[12–17] Within this field, investigations into molecular behaviour have explored potential use as wires[18–25], diodes[26,27], and even transistors[28].

I(s) techniques were first introduced in 2003 by Xu and Tao as a method for measuring the conductance behaviour of individual molecules.[29] These techniques work by temporarily disabling the active lateral movement of the STM tip such that the z-piezo actuator is solely controlled. Using only movement in the Z axis, an STM probe is repeatedly approached and withdrawn from a conductive sample. During this process, the tunnelling current is sampled at a user-defined frequency to yield withdrawal traces as shown in Figure 1.1. In general, traces start at a user-defined setpoint followed by an exponential decay of current to a noise level. The starting point and any deviations from exponential decay within the trace depend on the nature of the I(s) experiment. Within the original study, the authors were able to present results for the conductance of 4,4’-bipyridine as well as for various alkanedithiols. In addition, they measured the conductance of a single tunnelling channel consisting of a single gold atom contact. Both of these results agreed with theoretical calculations of molecular resistances and decay constants that were available at the time.[30,31]

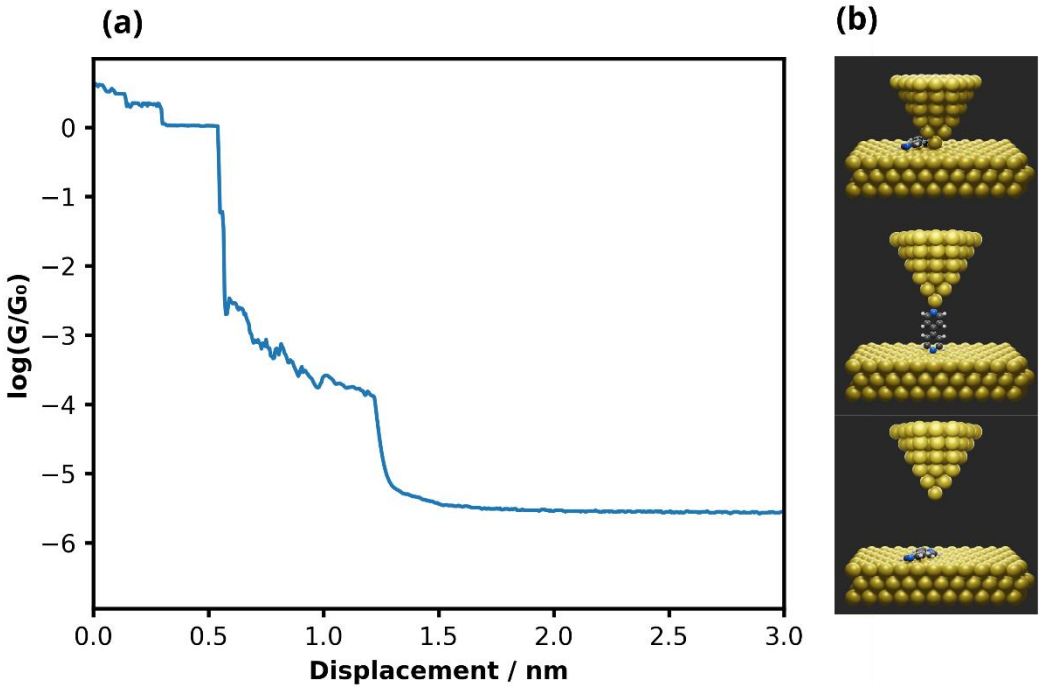



*Figure 1.1*

*An example BJ trace of 4,4'-bipyridine measured between gold substrate and tip. (a) shows one I(s) curve that contains notable features at log(G/G0) values of 0, and between -2.5 and -4. These correspond to gold-gold contact, and a molecular plateau, respectively. (b) shows the physical setup within the STM that results in the curve features. Shown is the gold-gold contact (top), the molecular bridge(middle), and the break after the molecular plateau (bottom).*

There are two main subsets of the I(s) technique. These are the STM break junction (STM-BJ) and the "soft-contact" I(s) techniques.[24] Both techniques function identically apart from having different starting setpoints. For STM-BJs the initial setpoint is set such that the tip and substrate are placed into physical contact. At the start of a withdrawal trace, this physical contact is represented by several conductance plateaus, which correspond to tip-substrate atomic connections. This behaviour can be seen in Figure 1.1a where a plateau manifests at 0 $\log(G/G_0)$, which arises from the formation of a single-atom junction bridging the STM tip and

substrate. The elongated nature of this plateau results from the rearrangement of surface atoms as the junction is pulled. The junction ultimately breaks once the tip has withdrawn sufficiently, which is measured as a sharp conductance decrease back to the exponential decay regime. The end of the $G_0$ plateau can be used as a point of reference for electrode separation distance, as this plateau occurs when the electrode separation is functionally zero.

In the case of "soft contact" I(s) spectroscopy, physical contact between electrodes is normally not reached thus removing the corresponding plateau features. This comes at the cost of losing the distance reference point but can reduce damage to the STM tip and substrate, as physical contact often results in destructive changes to metallic surfaces. Overall, this can help reduce variations in trace measurements as the involved surfaces are more preserved.[24]

For both techniques, the remainder of the withdrawal traces are similar. If a molecule of interest were to be adsorbed onto the substrate, this could lead to additional plateau features within traces such as the plateau shown in Figure 1.1a between -2.5 and -4 $\log(G/G_0)$. This occurs due to the formation of more junctions. In this instance, the junction is formed due to the adsorbed molecule of interest establishing a bridge between conductive surfaces. The conductance and length of these plateaus depend on the electronic and physical composition of the analyte molecule, respectively.[19]

After measurement, the many withdrawal repeats are subsequently analysed. In general, there are two molecular properties that are typically extracted from these results. These are the molecular conductance and the plateau break-off distance. To acquire the molecular conductance, the conductance values measured during the molecular plateaus require extraction. However, the extraction of these values is not a trivial task as the starting point

and length of plateaus are not identical between traces.[29] Indeed, locating and extracting molecular plateaus on a trace-by-trace basis would be difficult to automate due to the variations between repeat traces, and manual extraction would not be appropriate due to the large number of traces within I(s) datasets. In fact, it shouldn't be assumed that the molecular plateaus are present in all of the measured traces. In extreme cases where there are contaminants or large instrumental error, measured traces may contain artefacts that prevent or hide the presence of the desired molecular conductance plateau. In extreme cases such as these, data cleaning and selection may be required, (Section 5.2.2).

To reliably produce a meaningful measurement of the molecular conductance from the varied positions of molecular plateaus, the conductance values of entire traces are presented in a histogram to better visualise the conductance distributions. The more consistent features within the break junction (BJ) traces will cause peaks within these histograms. With enough plateau measurements, an overall distribution of conductance values will be produced such that the average molecular conductance can be determined despite the deviations caused by error and intrinsic variations (Figure 1.2).[32] In addition, the degree of variation can also be determined by studying the width of histogram peaks.

The second property, the break-off distance, corresponds to the length of the measured molecule for the molecule's orientation at the moment of measurement. This value, therefore, provides insight into the binding of the molecule within junctions. To determine the break-off distance, a distance reference point is required. Here, the distance is zeroed to the substrate-tip contact distance which coincides with the rupture of a conductance plateau measured at a value equal to $G_0$. After this, the break-off distance is measured as the distance at which the

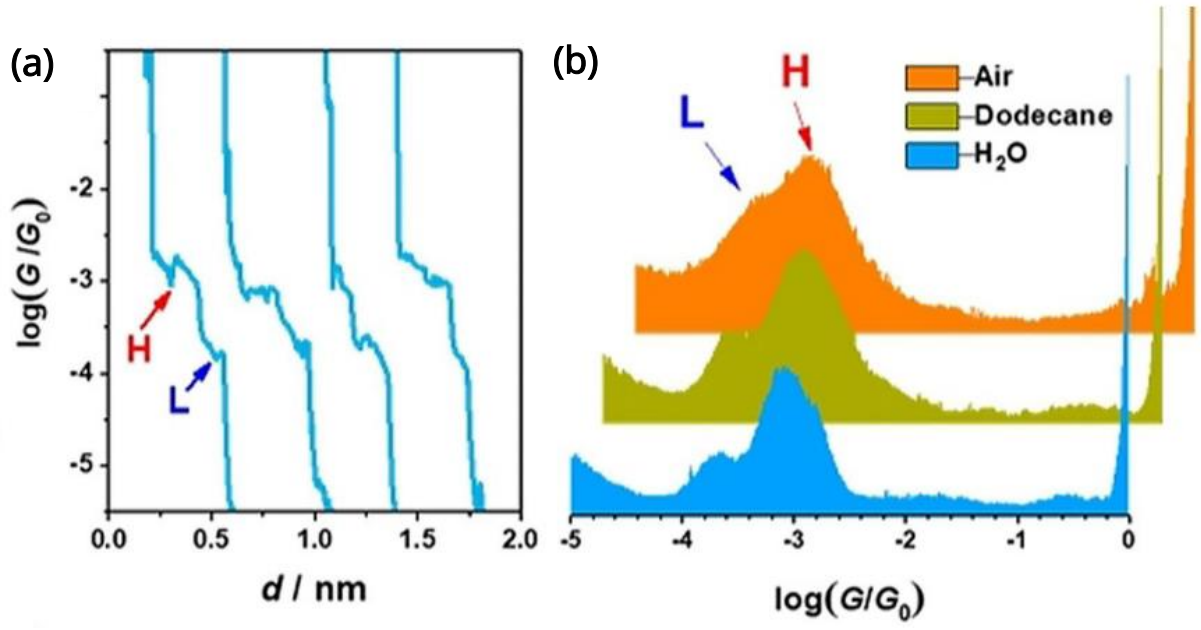



*Figure 1.2*

*Example BJs acquired by sampling 4,4'bipyridine in different mediums adapted from the works of Yu et al.*[32] *(a) shows four example BJ traces. (b) shows how many individual traces are conventionally combined into conductance histograms. These histograms show peaks that coincide with the presence of molecular plateaus.*

conductance falls below a certain value. This value is chosen such that it is lower than the expected molecular conductance, and larger than the noise floor of the exponential decay. Like with the conductance measurements, there are contributing factors that cause uncertainty in the break-off distance. For instance, the break-off distance is influenced by the molecular configuration within the junction and is also affected by the rearrangement of atoms at the conductive surfaces.[33] There are also cases where the break-off conductance threshold is not reached or is reached at distances much shorter than expected due to poor molecular binding. In these cases, data cleaning and thresholding may also be required. Ultimately, the same histogram analysis is generally employed to yield the average break-off distance, and thus the most probable junction length.

This method of analysis, whilst effective at demonstrating the conductance and break-off characteristics of repeated BJ measurements, does have limitations. For instance, consider the case where a bimodal distribution of molecular conductance is presented (Figure 1.3). Here two different values for molecular conductance can be seen. However, this visualisation through histograms does not immediately provide insight into how these two features manifest. The presence of these two peaks can be due to either all traces containing two molecular plateaus (Figure 1.3b) or due to two different populations of BJs that have different molecular conductance (Figure 1.3c). The former case potentially resulting from a chain of two molecules[34,35] or from a change of molecular orientation during a pull[36], and the latter case

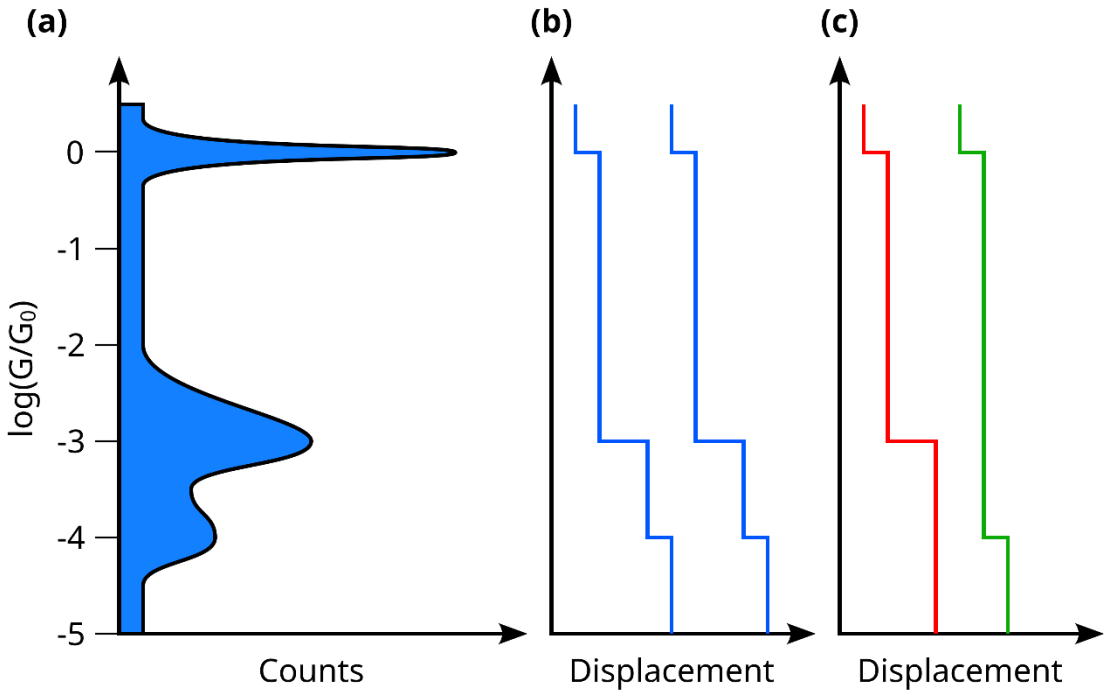



*Figure 1.3*

*Demonstration of the different cases that would produce the conductance distribution presented in (a). (b) shows the case of all traces containing two molecular plateaus, whereas (c) shows the case of two different molecular plateaus that are present in separate traces.*

resulting from two distinct molecular species or binding orientations. To determine which case applies, additional thresholds would need to be implemented where traces that contain a high number of data points within a range of conductance values selected from a chosen distribution peak are extracted. This analysis would be difficult, as it would require manually determining the thresholds for this extraction based on the histogram peak of interest. Likewise, the previous case of applying thresholds to remove erroneous traces is also manual as these thresholds are also decided by the analyst. This manual process of user-defined thresholds for selecting groups of interest is not desirable. Instead, an automated process that can reduce the user interpretation and prior assumptions that comes with manual processes would be more suitable.

From the desire to develop this analysis process, works have been turning to machine learning (ML) for a more powerful toolset. For instance, in 2016 a study by Lemmer et al. demonstrated a novel dimensionality reduction (DR) technique designed to study the presence of subpopulations within datasets of BJ traces.[37] Here, entire BJ traces were considered as vectors such that the differences between them could be quantified and plotted as a lower-dimensional representation. This method, known as multiparameter vector-based classification (MPVC), was able to reveal clusters within simulated and experimental BJ data that were previously hidden in the statistical histogram visualisation method. All of which was achieved whilst making no prior assumptions as to the appearances of traces features (Figure 1.4). Following this work, additional studies explored other DR techniques for the purpose of BJ analysis. This included investigations into the use of principal component analysis (PCA) for the purpose of DR,[38] and has more recently been explored using deep learning (DL) autoencoders (AEs) (Figure 1.5).[39] In particular, a recent work by Vladyka et al. investigated

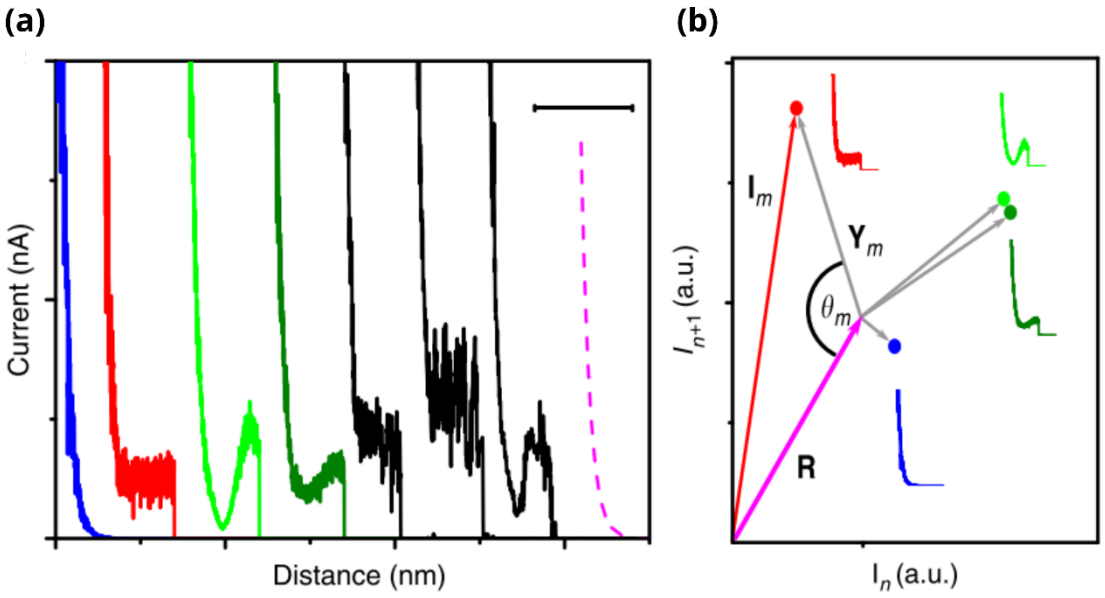



*Figure 1.4*

*Demonstration of the DR of BJs using MPVC adapted from the works of Lemmer et al.*[37] *(a) shows simulated BJs as well as the reference vector (pink). (b) shows the DR embeddings relative to the reference vector using MPVC. The BJs that are most similar are closer within this space.*

the potential of transfer learning (TL) and AEs on the DR of experimental BJ traces.[40] Here BJ traces were converted into RGB images such that the entire shape of traces was able to be preserved in the subsequent ML analysis. It was found that BJ traces sampled from separate molecules manifested in different overlapping clusters within 2D representations. Complementary to this finding it was also seen that two distinct subpopulations of BJ behaviour for one molecule were seen within the 2D embedding. These two subpopulations corresponded to one population of BJs with a single region of conductance density, and another population of BJs with two regions of conductance density. By utilising ML techniques, this analysis procedure was able to find subpopulations within BJ data whilst considering all variables of the I(s) behaviour of traces. From this analysis the nature of different clusters was compared in an attempt to reveal a physical meaning for the split. It was found that the

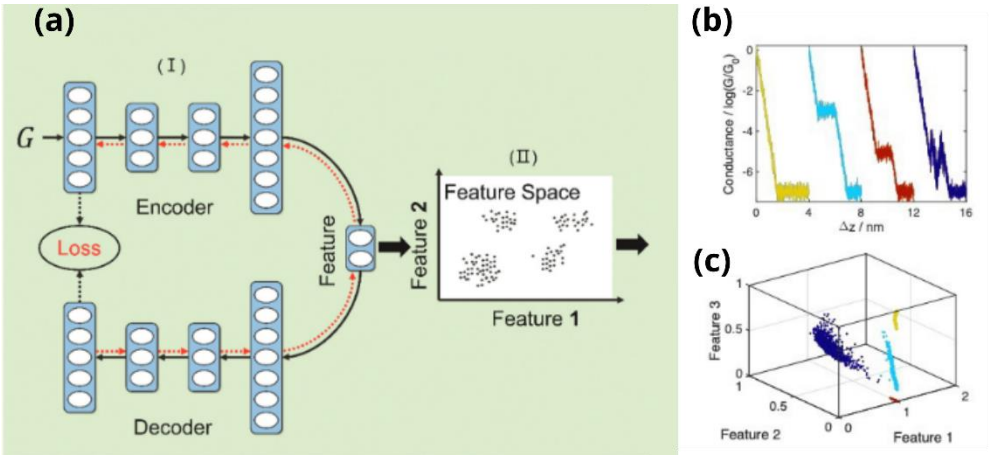



*Figure 1.5*

*Demonstration of the DR of BJs using an AE adapted from the works of Huang et al.*[39] *(a) shows a schematical diagram of an AE showing which neurone outputs are taken as the reduced dimensional embedding. (b) shows examples from the four different simulated BJ classes. (c) shows the three dimensional embeddings of the simulated BJs generated using this AE method.*

different clusters manifested as traces with different decay rates. This difference was speculated to result from reorganisation of the BJ surfaces or the molecule such that a shallower descent occurred. These behaviours would not have been previously visualised without the manual interpretation and assumptions that come with histogram analysis.

### *Current-Time Spectroscopy*

Following on from I(s) techniques, the next SM capture technique is the current-time (I(t)) technique. The technique was first developed in 2004 by Haiss et al. and involved maintaining the tip-substrate separation at a constant distance.[41] Therefore, the STM servos are only used to maintain vertical position during this technique. In more stable systems where there is no drift in the z-direction, the servos are turned off entirely such that the microscopes feedback system does not interfere with formed junctions. To carry out this technique, the STM tip is kept at a user-defined setpoint current from the substrate surface. For the variation of the I(t)

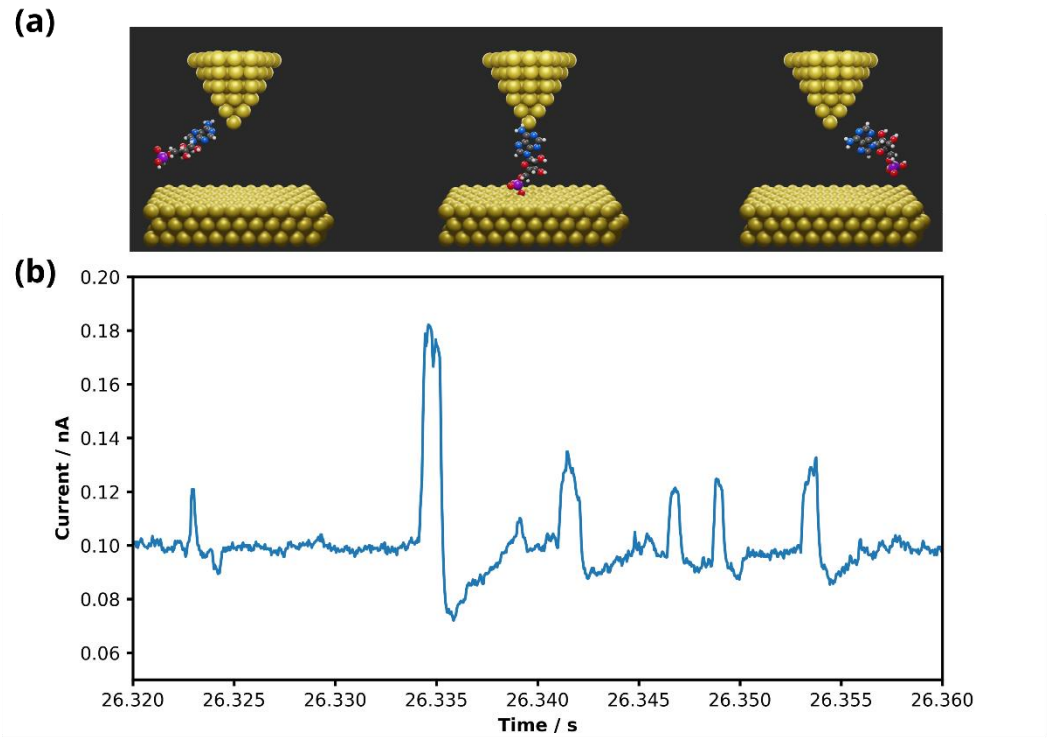



*Figure 1.6*

*Example I(t) events of ribose adenosine monophosphate (rAMP) in an electrochemical STM (ECSTM). (a) shows a diagrammatic representation of the junction formation process during a current modulation event. Namely before (left), during (middle), and after (right) a pulse. (b) shows a snippet of an I(t) trace that contains multiple square wave events that are all similar in appearance to the largest peak at 26.335s.*

technique used in the context of this thesis, a solution containing the molecule of study is present in the environment. This forms the basis of this technique where the analyte molecules can spontaneously diffuse into the tip-substrate gap and cause momentary boosts to the tunnelling current. Throughout this process the tunnelling current is measured as a function of time, which leads to the features demonstrated in Figure 1.6. As a molecule enters the gap, the current increases in accordance with the conductance of the molecular junction. Eventually, the molecule will leave the gap causing the current to drop back down to the baseline value.

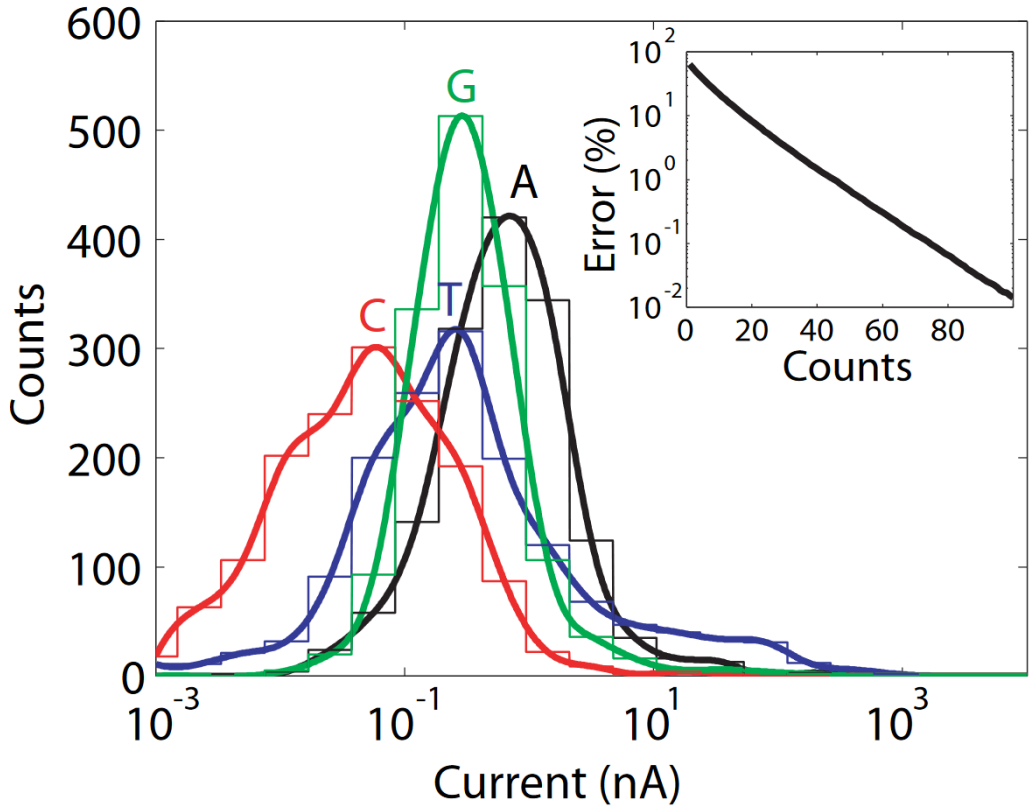



*Figure 1.7*

*Conductance values for the four nucleotide bases taken from the works of Lagerqvist et al.*[49] *The four distributions show considerable overlap which demonstrates the difficulties of classifying these molecules based on their electronic properties.*

Extensive research has been carried out using the I(t) technique to explore its viability as a potential next-generation DNA sequencing technique.[42–47] Initial research attempted to capture and measure nucleotides using the previously developed I(s) technique[48] but later also implemented the I(t) method. Whilst successful in measuring nucleotide events, the research encountered difficulties when trying to distinguish different nucleotides based on the I(t) events. The typical analysis for I(t) data was carried out in a similar manner to the I(s) technique with histograms and fitting distributions predominating. However, it was found that the conductance values of the four different bases had distributions that overlapped

considerably (Figure 1.7).[49] In an effort to improve on the concept of using tunnelling current to sequence DNA molecules, research varied to explore different approaches that could improve differentiation of I(t) events. One approach focussed on controlling the flow of DNA molecules through the tunnelling junction by embedding a tunnelling junction within a nanofluidic channel,[43,50–52] whereas another alteration aimed to improve the separability of conductance distributions by incorporating "molecular readers".

The second of these mentioned experimental variations is known as recognition tunnelling. This technique, in its first form, was introduced in 2005 by Ohshiro and Umezawa.[53] Here an STM tip was functionalised with nucleotide analogues containing thiol moieties. When imaging with a functionalised tip, the positions where the tip is situated above the complementary nucleotide showed increased tunnelling currents compared to other locations. Subsequent advancements of this concept then incorporated capture molecules with the STM I(t) technique to give the current form of recognition tunnelling.[42,44,54–56] There has even been work carried out to attempt to design a "universal" capture molecule that could cause I(t) event magnitudes to appear at varying levels for each of the four nucleotides.[57] The latest research with recognition tunnelling and nucleotide classification showed that despite incorporating capture probes the conductance and duration distributions of I(t) events still overlap considerably.[45] In addition to this, the incorporation of functionalising molecules to the STM electrodes can result in current behaviour characteristic of telegraph noise.[58] It has been speculated that this telegraph noise, results from spontaneous bond forming and breaking within the functionalised junction. Whilst, the presence of telegraph noise was theorised to be a good indicator of molecular binding, it also increases the complexity of the system making analysis more challenging.[59]

Due to the aforementioned difficulties within the experimentation of the I(t) technique for nucleotide classification, research has begun turning toward ML as a potential solution. The latest work from Im et al. implemented a support vector machine (SVM) alongside recognition tunnelling, which was able to classify nucleotide events with an average accuracy of around 85%.[46] The ability of the SVM to achieve this accuracy despite the considerable overlap of the events height and duration is a promising sign and a good case study proving the value that ML has to offer. More details on SVMs are outlined in Section 2.2.1.2.

### 1.1.2. Resistive Pulse Sensing

Resistive Pulse Sensing (RPS) is another technique that is being employed for the classification of DNA nucleotides.[60,61] A subset of RPS is the general nanopore-based sensing techniques that are being employed within SM science. The technique involves two reservoirs of an ionically conducting solution that are separated by a membrane with a nanoscale opening between them.[62] This membrane can be either biological with a biological pore such as a channel protein,[63–66] or a solid-state membrane made out of inorganic materials with a nanoscopic hole.[67–71] Within each reservoir, electrodes are immersed to drive electrophoretic movement based on user-defined potential differences. The ionic solution will contain molecules of study such that, once a potential difference is applied, the molecules will translocate towards the electrode of opposite charge. The translocation of each analyte molecule will occupy space within the separating nanopore. Due to this there can be a resultant drop in the flow of ionic charges. This leads to a drop in the electrical current running through the electrochemical cell which is measured as a resistive current event.[62] The appearance of a typical resistive event is shown in Figure 1.8. Immediately, the similarities between the data acquired with this technique and the aforementioned STM I(t) technique

become apparent. Events produced by each technique manifest as pseudo square wave pulses with duration and magnitude. The differences between techniques being the direction of the pulses and the finer data features that arise from the different physical systems.

This technique has seen successful commercial application in real-time DNA sequencing technologies where RPS events have been measured and analysed to classify DNA nucleotides in a sequence. The company, Oxford Nanopore Technologies, utilises many "flow cells" in parallel to measure blockage currents as entire DNA strands are translocated.[72] To analyse events, Oxford Nanopore Technologies employs recurrent neural networks (RNNs) to translate a sequence of current events into a sequence of DNA bases.[73] The use of this ML

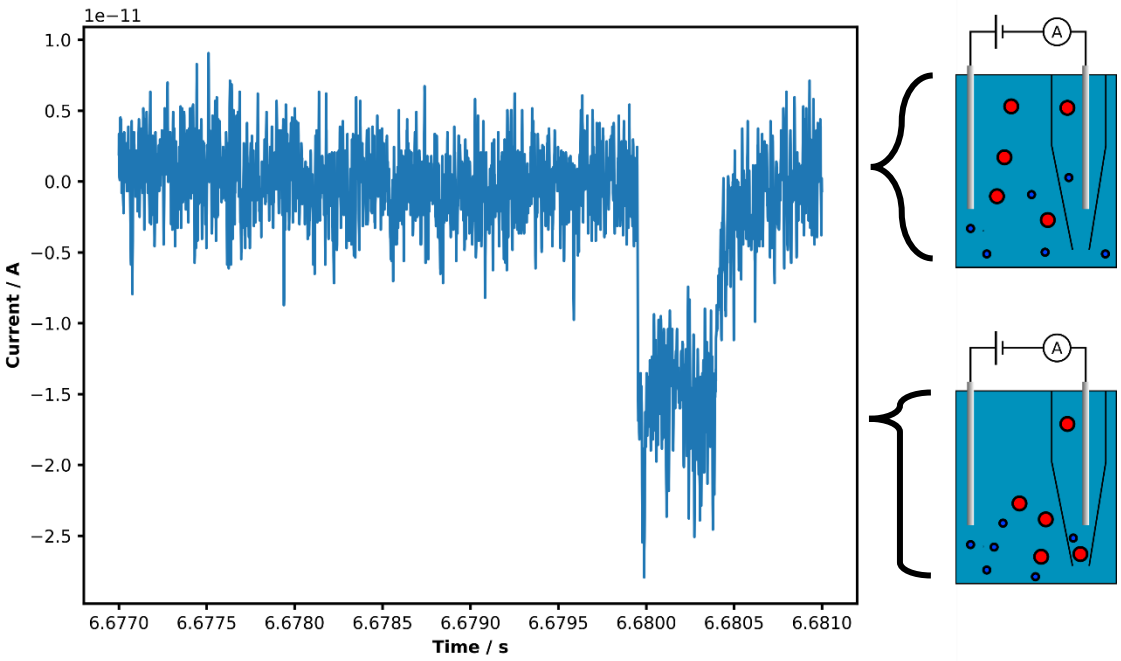



*Figure 1.8*

*Example event sampled during a nanopore translocation experiment. When a potential difference is applied, molecules can translocate through the nanopore and momentarily block the passage of ionic conductors (right). This resistive event manifests as a drop in the measured eletrical current (left).*

technique has allowed for nucleotide calling accuracies of greater than 99%.[74] The incorporation of modern ML strategies has provided a high predictive accuracy that has improved data analysis in this field.

## 1.2. Electrochemistry

Previously it was shown that the solutions to data challenges present in SM science could be improved by using techniques provided by the field of ML. Whilst the application of ML in SM science clearly shows potential, it is not clear how the application to other chemical disciplines would perform. For instance, there are many examples within the electrochemical field where ML would be applicable, and in fact, ML is already being implemented in various cyclic voltametric studies. This is of particular interest in the context of electronic characterisation of single molecules as the various phenomena that arise due to the application of voltages between conductive surfaces greatly impacts the signal measured during STM experiments. This relationship is demonstrated in Section 3.2.1.

### 1.2.1. Cyclic Voltammetry

Cyclic Voltammetry (CV) is an electrochemical technique used to study the redox behaviour of different materials and molecules. In general, a potential difference is swept between electrodes such that the potential of a material of interest is also swept.[75,76] The rate at which this sweep changes the potential difference is set by the operator and is an important parameter that determines the appearance of resultant traces.[77] During the potential sweep the current passing through the circuit is measured. An example of the general shape of CVs is shown in Figure 1.9, which is famously compared to the profile of a duck.

The main features of a CV consist of an anodic (positive) sweep followed by a cathodic (negative) sweep of potential, or *vice versa* depending on the sweep direction. On the anodic sweep a peak will form at a potential where the working electrode would be oxidised. Then on the cathodic sweep a second peak will appear at the potential where the oxidised working electrode reduces back to the non-oxidised form. CVs can contain additional features based on any adsorbed surface molecules or if gases begin to evolve which helps chemists to study surface reactions.[78]

Analysis of CVs can be a challenging process. To extract physical properties of electrochemical systems theoretical models are applied using characteristics extracted from voltammogram curves. For example, to measure the size of the electrochemically active surface area of the working electrode, one has to find the area underneath either of the redox peaks.[79] Other analysis strategies involve, measuring redox peak separation to quantify reversibility and

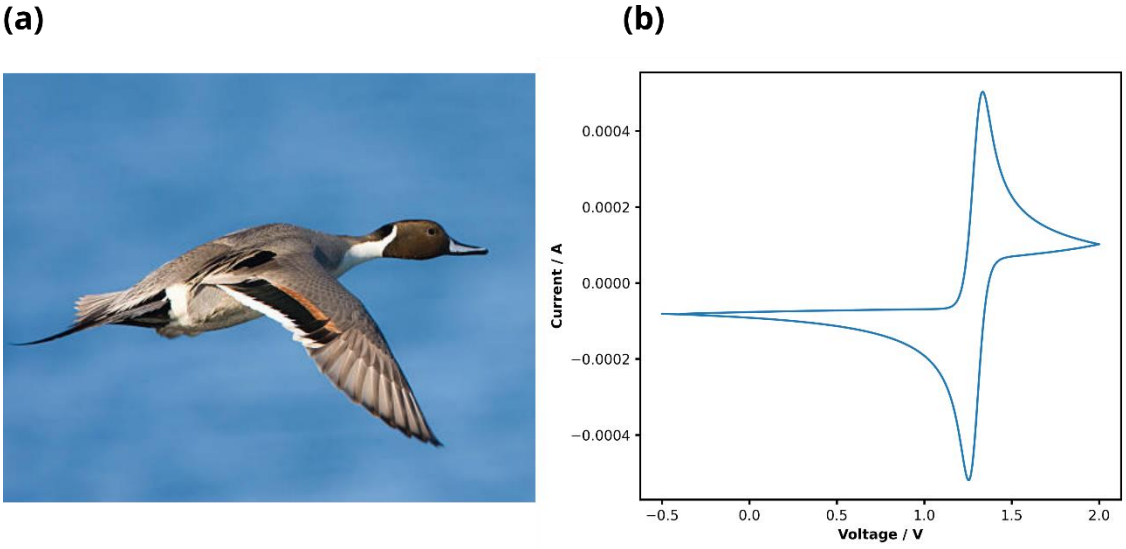



*Figure 1.9*

*An example CV of an E coupled reaction, (b). The example only contains peaks for the standard redox behaviour. The shape of CVs is commonly compared with the appearance of ducks which is shown by comparing with a Northern Pintail Duck in flight, (a).*

transfer kinetics,[77,80] and measuring curve area at non-redox active potentials for double layer characterisation.[81] These types of analyses are easy to perform on measurements sampled from well-defined systems that do not vary significantly from expected behaviour. However, when systems are more complex due to the system containing a wide variety of electrochemically active species, or the geometry of involved surfaces changes over time or between experiments, voltammogram shapes vary considerably which hinders the extraction of physical properties.[78,82,83] Additionally, in situations where CVs are sampled in large volumes continuously, automated processes would also be needed to quickly characterise systems with high throughput. In case of both high system complexity and the need for high analysis throughput, the traditional application of numerical and theoretical modelling is not immediately appropriate.

As an example of system complexity, consider the experimental determination of the electrochemical reaction mechanism. When the products of an electron transfer process (E) are chemically active within the system under study, these products can react with other species within the system. This chemical reaction (C) alters the dynamics of the diffusion layer and results in a change in CV shape. These processes are known as EC coupled reactions and are difficult to characterise.[84,85] The shape changes that arise can be very subtle making it tricky to classify a CVs shape as having either E or EC type behaviour. The complexity rises even further when the chemically produced species is also electrochemically active. This results in even more redox peaks manifesting in CV curves (Figure 1.10).[86,87] Stemming from the complexity of elucidating electrochemical reaction mechanisms, studies have been exploring the predictive capabilities of ML techniques to ease this analysis. For instance, multiple studies have explored supervised learning applications of neural networks for the

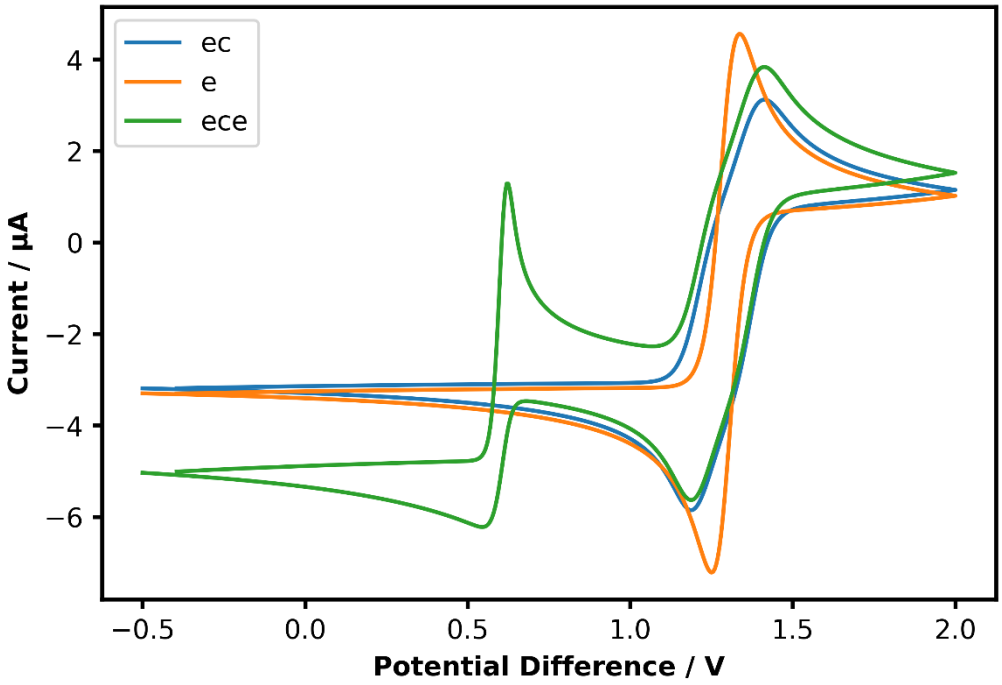



*Figure 1.10*

*Example CVs that simulate the three different electrochemical reaction mechanisms. The E mechanism involves a simple electron transfer step (blue) whereas the EC mechanism involves the chemical reaction of the products of an electron transfer step (orange). This leads to a subtle change in CV appearance due to alterations in chemical kinetics. The ECE mechanism (green) presents a more noticeable feature where the product of the EC reaction is also able to undergo an electron transfer step.*

classification of CVs based on their reaction mechanism.[88–91] These studies utilised datasets of CVs with predetermined reaction mechanisms to train ML models for the accurate prediction of previously unseen CVs.

In another example application, CV is being implemented for the purpose of electrochemical sensors. This stems from the high sensitivity this technique has in producing curves with different shapes resulting from the presence of additional electrochemically active species within the measured environment.[92–97] Within sensing applications, the measured systems can contain a wide range of molecular species which can result in voltammograms with

complex shapes. In addition, sensing techniques are commonly applied in settings where high volumes of data require continuous analysis with as high turnover rates as possible. Because of this, researchers have been attempting to incorporate novel analysis techniques to try and ease the process of analysing electrochemical sensor data. A good case study for this would be the analysis of electrochemical tongue (E-tongue) data.

The purpose of the E-tongue is to detect chemicals that would likewise be detected by the human tongue.[98] One application of such a device finds itself within the field of food sciences. Here, E-tongues are used to discriminate batches of drink items based on their quality. For instance, one study showed that E-tongues can be used to monitor the quality of beer as it ages.[99] In fact, there are multiple studies similar to this that implement ML techniques to aid in sample classification tasks using an E-tongue. The techniques used in this field vary from DR to implementation of neural networks (Figure 1.11).[100–116] It is worth noting that ML techniques are also aiding electrochemical sensors in other chemical applications such as the detection of hazardous substances.[117]

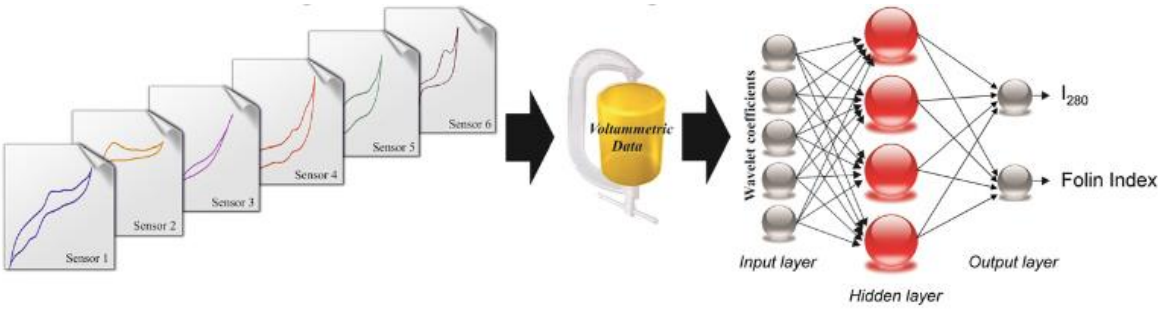



*Figure 1.11*

*Example of an E-tongue pipeline from the works of Ceto et al.[108] where CVs are sampled from sensors before being processed by a ML agent. Here a neural network has been taught to predict the Folin-Ciocalteu index and Polyphenol Index of wines based on their CV appearance. Both indices are used to quantify the quality and flavour of wines.*

## 1.3. Motivation for Machine Learning

In Sections 1.1 and 1.2 example case studies were presented where ML algorithms were applied to complex data problems within the fields of SM science and electrochemistry. In both cases, it was shown that ML agents were able to achieve high classification accuracies, and effective data separation without the need for statistical analysis. Whilst these results are certainly fascinating, it is not immediately clear how ML techniques operate, and how they differ with regards to traditional statistics. A formal definition of AI and ML, as well as a detailed outline of the various topics and algorithms used within the context of this thesis in relation to ML are discussed in Sections 2.2 and 2.3. Overall, ML techniques provide a means for classifying data, forecasting variables using automated regression, and clustering data into groups based on similarity. All of which is achievable without any prior statistical assumptions being made.

However, from one point of view the application of ML to data problems could be seen as an unnecessary overcomplication. In many cases, data analysis tasks could be simply solved by applying less complicated statistical methods. For instance, when studying the absorptivity of molecular species. Here the concentration of a compound is linearly related to its absorbance as given by the Beer-Lambert Law.[118] This means that usually only a simple linear regression model is required to accurately obtain the molecules absorptivity.

The differences and usefulness of ML techniques can be understood when one considers the fundamental purpose of data analyses. In general, data analysis aims can be split into two interrelated desires. The first goal of analyses is to understand how the different variables of a system relate in nature. This is known as inference, which is predominantly achieved by

drawing conclusions from statistical models of measured data.[119] The second goal of analyses is to produce a model that can accurately forecast the value of various variables given a previously sampled dataset. This is called prediction and has seen greater success when utilising the models provided by ML.[120] These two goals are very similar as inferring an accurate understanding of any dataset's nature can allow for accurate predictions and *vice versa*. However, in some cases where data is numerous and possesses a large number of variables, the statistical inference can only be achieved with a less successful predictive ability. When instead focussing on predictive power using ML models, the opposite can occur whereby high prediction accuracy is achieved but at the cost of being unable to infer the nature of the data. This lack of inference results from the reduced interpretability that comes with more advanced ML agents. In general, one would choose to apply ML models to data analysis when the focus is prediction, and the dataset contains a vast number of interrelated variables and datapoints.

Stemming from the similarities and overlaps that statistics and ML possess, there is notable confusion regarding the appropriate use cases of each. In 2018, a study by Bzdok et al. called "Statistics versus machine learning" was published.[121] Here the comparison between statistics and ML was made using their performances on a regression task. This task required the applied algorithms to correctly predict which genes within a set were the cause of a change in a simulated biological phenotype. This simulation involved 40 different genes where the last 10 had varied expression between the two labelled phenotypes. The statistical approach utilised a p-value corrected binomial hypothesis test to label the genes, whereas the ML approach utilised a random forest algorithm to directly generate predictions. Overall, similar results were found for either algorithm. However, it was concluded that the ML approach was

superior as the statistical method made use of previous knowledge regarding RNA sequencing data to produce hypotheses to test, whereas the ML approach required no prior assumptions. Additionally, the ML approach was said to be more scalable as larger systems involving larger sequences of genes would have a lesser impact on the difficulty of the analysis process of the random forest regressor than it would with the hypothesis test.

Overall, the real benefit that ML can provide to analysis tasks is the ability to produce models with improved predictive abilities when compared to traditional analysis methods. Furthermore, these ML techniques allow for easier data analysis when considering large datasets with many interrelated variables, and allow for traditional statistical assumptions, such as normality and confidence intervals, to be avoided.

## 1.4. Aims and Projects

The complete aim of this thesis is now outlined. Overall, this work contains three main projects that will be outlined below.

The first project reexplores the concept of nucleotide classification using their electrical properties. The aim of this project is to look at potential techniques one could use with an STM to more reliably detect I(t) technique events involving nucleotides. Taking inspiration from the achievements of nanopore-based DNA sequencing, this project also explores the use of more advanced ML techniques than those previously used to classify I(t) events, specifically neural networks and DL (Chapter 3).

In Chapter 4, a comparison of statistical and ML techniques for anomaly detection is performed. Specifically, the aim here is to identify which technique is best suited to find anomalies (events) in noisy, low signal-to-noise ratio (SNR) time series data.

Lastly, Chapter 5 investigates an in-depth study of the uses of unsupervised data classification within the field of STM-BJ. Here DR and clustering is used to demonstrate the ML approach to data selection and sorting based on unbiased means. More specifically, this project aims to identify the improvements that can be made by ML for data cleaning and sub-population analysis. Additionally, to demonstrate the versatility of these techniques an application in CV is also presented.

## 1.5. References


1 D. E. Makarov, *Single Molecule Science: Physical Principles and Models*, 2015.

2 S. Singh, V. Kumar, D. S. Dhanjal, S. Datta, R. Prasad and J. Singh, in Microbial Biotechnology: Basic Research and Applications, 2020, pp. 317–335.

3 C. A. Mirkin and M. A. Ratner, *Annu Rev Phys Chem*, 1992, **43**, 719–754.

4 S. Mueller, in *Upgrading and Repairing PCs*, 17th edn., 2006.

5 Apple unveils M2 Pro and M2 Max: next-generation chips for next-level workflows, https://www.apple.com/newsroom/2023/01/apple-unveils-m2-pro-and-m2-max-next-generation-chips-for-next-level-workflows/, (accessed 15 May 2023).

6 F. Chen, J. Hihath, Z. Huang, X. Li and N. J. Tao, *Annu Rev Phys Chem*, 2007, **58**, 535–564.

7 W. A. Hofer, *Prog Surf Sci*, 2003, **71**, 147–183.

8 V. Yu. Yurov and A. N. Klimov, *Review of Scientific Instruments*, 1994, **65**, 1551–1557.

9 J. J. Gooding and K. Gaus, *Angewandte Chemie International Edition*, 2016, **55**, 11354–11366.

10 G. Binnig and H. Rohrer, *Surf Sci*, 1983, **126**, 236–244.

11 C. J. Chen, *Introduction to Scanning Tunneling Microscopy*, Oxford University Press, Oxford, 2nd edn., 2007.

12 M. T. González, S. Wu, R. Huber, S. J. van der Molen, C. Schönenberger and M. Calame, *Nano Lett*, 2006, **6**, 2238–2242.

13 L. Venkataraman, J. E. Klare, C. Nuckolls, M. S. Hybertsen and M. L. Steigerwald, *Nature*, 2006, **442**, 904–907.

14 J. He, O. Sankey, M. Lee, N. Tao, X. Li and S. Lindsay, *Faraday Discuss.*, 2006, **131**, 145–154.

15 J. M. Hamill, C. Weaver and T. Albrecht, *The Journal of Physical Chemistry C*, 2021, **125**, 26256–26262.

16 N. J. Tao, *Nat Nanotechnol*, 2006, **1**, 173–181.

17 F. Chen, J. Hihath, Z. Huang, X. Li and N. J. Tao, *Annu Rev Phys Chem*, 2007, **58**, 535–564.

18 V. Kaliginedi, A. V. Rudnev, P. Moreno-García, M. Baghernejad, C. Huang, W. Hong and T. Wandlowski, *Physical Chemistry Chemical Physics*, 2014, **16**, 23529–23539.

19 Y. Komoto, S. Fujii, M. Iwane and M. Kiguchi, *J Mater Chem C Mater*, 2016, **4**, 8842–8858.

20 A. Nitzan and M. A. Ratner, *Science*, 2003, **300**, 1384–1389.

21 L. Venkataraman, J. E. Klare, C. Nuckolls, M. S. Hybertsen and M. L. Steigerwald, *Nature*, 2006, **442**, 904–907.

22 C. J. Muller, J. M. van Ruitenbeek and L. J. de Jongh, *Physica C Supercond*, 1992, **191**, 485–504.

23 R. Requist, P. P. Baruselli, A. Smogunov, M. Fabrizio, S. Modesti and E. Tosatti, *Nat Nanotechnol*, 2016, **11**, 499–508.

24 A. Vezzoli, I. M. Grace, C. Brooke, R. J. Nichols, C. J. Lambert and S. J. Higgins, *J Chem Phys*, 2017, **146**, 092307.

25 W. Haiss, H. van Zalinge, S. J. Higgins, D. Bethell, H. Höbenreich, D. J. Schiffrin and R. J. Nichols, *J Am Chem Soc*, 2003, **125**, 15294–15295.

26 M. L. Perrin, E. Galán, R. Eelkema, J. M. Thijssen, F. Grozema and H. S. J. Van Der Zant, *Nanoscale*, 2016, **8**, 8919–8923.

27 C. Yang, Y. Guo, S. Zhou, Z. Liu, Z. Liu, D. Zhang and X. Guo, *Advanced Materials*, 2023, **35**, 2209750.

28 M. L. Perrin, E. Burzurí and H. S. J. Van Der Zant, *Chem Soc Rev*, 2015, **44**, 902–919.

29 B. Xu and N. Tao, *Science*, 2003, **301**, 1221–1223.

30 D. Segal, A. Nitzan, W. B. Davis, M. R. Wasielewski and M. A. Ratner, *J Phys Chem B*, 2000, **104**, 3817–3829.

31 J. K. Tomfohr and O. F. Sankey, *Phys Rev B*, 2002, **65**, 245105.

32 Z. Yu, Y. Xu, J. Su, P. M. Radjenovic, Y. Wang, J. Zheng, B. Teng, Y. Shao, X. Zhou and J. Li, *Angewandte Chemie International Edition*, 2021, **60**, 15452–15458.

33 T. Fu, K. Frommer, C. Nuckolls and L. Venkataraman, *J Phys Chem Lett*, 2021, **12**, 10802–10807.

34 D. Stefani, C. Guo, L. Ornago, D. Cabosart, M. El Abbassi, M. Sheves, D. Cahen and H. S. J. van der Zant, *Nanoscale*, 2021, **13**, 3002–3009.

35 C. Wu, A. Alqahtani, S. Sangtarash, A. Vezzoli, H. Sadeghi, C. M. Robertson, C. Cai, C. J. Lambert, S. J. Higgins and R. J. Nichols, *Nanoscale*, 2020, **12**, 7914–7920.

36 Y.-Y. Sun, Z.-L. Peng, R. Hou, J.-H. Liang, J.-F. Zheng, X.-Y. Zhou, X.-S. Zhou, S. Jin, Z.-J. Niu and B.-W. Mao, *Physical Chemistry Chemical Physics*, 2014, **16**, 2260.

37 M. Lemmer, M. S. Inkpen, K. Kornysheva, N. J. Long and T. Albrecht, *Nat Commun*, 2016, **7**, 1–10.

38 J. M. Hamill, X. T. Zhao, G. Mészáros, M. R. Bryce and M. Arenz, *Phys Rev Lett*, 2018, **120**, 016601.

39 F. Huang, R. Li, G. Wang, J. Zheng, Y. Tang, J. Liu, Y. Yang, Y. Yao, J. Shi and W. Hong, *Physical Chemistry Chemical Physics*, 2020, **22**, 1674–1681.

40 A. Vladyka and T. Albrecht, *Mach Learn Sci Technol*, 2020, **1**, 035013.

41 W. Haiss, R. J. Nichols, H. van Zalinge, S. J. Higgins, D. Bethell and D. J. Schiffrin, *Physical Chemistry Chemical Physics*, 2004, **6**, 4330.

42 P. Pang, B. A. Ashcroft, W. Song, P. Zhang, S. Biswas, Q. Qing, J. Yang, R. J. Nemanich, J. Bai, J. T. Smith, K. Reuter, V. S. K. Balagurusamy, Y. Astier, G. Stolovitzky and S. Lindsay, *ACS Nano*, 2014, **8**, 11994–12003.

43 T. Ohshiro, K. Matsubara, M. Tsutsui, M. Furuhashi, M. Taniguchi and T. Kawai, *Sci Rep*, 2012, **2**, 501.

44 S. Chang, S. Sen, P. Zhang, B. Gyarfas, B. Ashcroft, S. Lefkowitz, H. Peng and S. Lindsay, *Nanotechnology*, 2012, **23**, 425202.

45 S. Biswas, S. Sen, J. Im, S. Biswas, P. Krstic, B. Ashcroft, C. Borges, Y. Zhao, S. Lindsay and P. Zhang, *ACS Nano*, 2016, **10**, 11304–11316.

46 J. Im, S. Sen, S. Lindsay and P. Zhang, *ACS Nano*, 2018, **12**, 7067–7075.

47 H. Tanaka and M. Taniguchi, *Jpn J Appl Phys*, 2017, **56**, 08LB02.

48 J. He, L. Lin, P. Zhang and S. Lindsay, *Nano Lett*, 2007, **7**, 3854–3858.

49 J. Lagerqvist, M. Zwolak and M. Di Ventra, *Nano Lett*, 2006, **6**, 779–782.

50 M. Tsutsui, M. Taniguchi, K. Yokota and T. Kawai, *Nat Nanotechnol*, 2010, **5**, 286–290.

51 T. Ohshiro, M. Tsutsui, M. Taniguchi and T. Kawai, in *2012 12th IEEE International Conference on Nanotechnology (IEEE-NANO)*, IEEE, 2012, pp. 1–2.

52 Y. Sasaki, T. Ohshiro, S. Kawano, M. Taniguchi and T. Kawai, in *2012 12th IEEE International Conference on Nanotechnology (IEEE-NANO)*, IEEE, 2012, pp. 1–5.

53 T. Ohshiro and Y. Umezawa, *Proceedings of the National Academy of Sciences*, 2006, **103**, 10–14.

54 S. Huang, J. He, S. Chang, P. Zhang, F. Liang, S. Li, M. Tuchband, A. Fuhrmann, R. Ros and S. Lindsay, *Nat Nanotechnol*, 2010, **5**, 868–873.

55 S. Chang, S. Huang, J. He, F. Liang, P. Zhang, S. Li, X. Chen, O. Sankey and S. Lindsay, *Nano Lett*, 2010, **10**, 1070–1075.

56 S. Chang, S. Huang, H. Liu, P. Zhang, F. Liang, R. Akahori, S. Li, B. Gyarfas, J. Shumway, B. Ashcroft, J. He and S. Lindsay, *Nanotechnology*, 2012, **23**, 235101.

57 F. Liang, S. Li, S. Lindsay and P. Zhang, *Chemistry - A European Journal*, 2012, **18**, 5998–6007.

58 S. Lindsay, J. He, O. Sankey, P. Hapala, P. Jelinek, P. Zhang, S. Chang and S. Huang, *Nanotechnology*, 2010, **21**, 262001.

59 S. Chang, J. He, L. Lin, P. Zhang, F. Liang, M. Young, S. Huang and S. Lindsay, *Nanotechnology*, 2009, **20**, 185102.

60 U. F. Keyser, *Nat Nanotechnol*, 2016, **11**, 106–108.

61 D. Kozak, W. Anderson, R. Vogel and M. Trau, *Nano Today*, 2011, **6**, 531–545.

62 R. W. DeBlois and C. P. Bean, *Review of Scientific Instruments*, 1970, **41**, 909–916.

63 Y. Astier, O. Braha and H. Bayley, *J Am Chem Soc*, 2006, **128**, 1705–1710.

64 T. Z. Butler, J. H. Gundlach and M. Troll, *Biophys J*, 2007, **93**, 3229–3240.

65 G. Maglia, M. R. Restrepo, E. Mikhailova and H. Bayley, *Proceedings of the National Academy of Sciences*, 2008, **105**, 19720–19725.

66 J. Nivala, D. B. Marks and M. Akeson, *Nat Biotechnol*, 2013, **31**, 247–250.

67 J. Li, M. Gershow, D. Stein, E. Brandin and J. A. Golovchenko, *Nat Mater*, 2003, **2**, 611–615.

68 A. J. Storm, C. Storm, J. Chen, H. Zandbergen, J.-F. Joanny and C. Dekker, *Nano Lett*, 2005, **5**, 1193–1197.

69 S. M. Iqbal, D. Akin and R. Bashir, *Nat Nanotechnol*, 2007, **2**, 243–248.

70 K. Healy, B. Schiedt and A. P. Morrison, *Nanomedicine*, 2007, **2**, 875–897.

71 D. Fologea, J. Uplinger, B. Thomas, D. S. McNabb and J. Li, *Nano Lett*, 2005, **5**, 1734–1737.

72 Flow cells and nanopores, https://nanoporetech.com/how-it-works/flow-cells-and-nanopores, (accessed 5 January 2023).

73 How basecalling works, https://nanoporetech.com/how-it-works/basecalling, (accessed 5 January 2023).

74 Nanopore Sequencing Accuracy, https://nanoporetech.com/accuracy, (accessed 4 September 2023).

75 D. H. Evans, K. M. O'Connell, R. A. Petersen and M. J. Kelly, *J Chem Educ*, 1983, **60**, 290.

76 P. T. Kissinger and W. R. Heineman, *J Chem Educ*, 1983, **60**, 702.

77 G. A. Mabbott, *J Chem Educ*, 1983, **60**, 697.

78 N. Elgrishi, K. J. Rountree, B. D. McCarthy, E. S. Rountree, T. T. Eisenhart and J. L. Dempsey, *J Chem Educ*, 2018, **95**, 197–206.

79 S. Trasatti and O. A. Petrii, *Journal of Electroanalytical Chemistry*, 1992, **327**, 353–376.

80 R. J. Klingler and J. K. Kochi, *J Phys Chem*, 1981, **85**, 1731–1741.

81 D. M. Morales and M. Risch, *Journal of Physics: Energy*, 2021, **3**, 034013.

82 G. A. Mabbott, *J Chem Educ*, 1983, **60**, 697.

83 K. R. Ward, N. S. Lawrence, R. S. Hartshorne and R. G. Compton, *Physical Chemistry Chemical Physics*, 2012, **14**, 7264.

84 L. Čižmek, Š. Komorsky-Lovrić and M. Lovrić, *ChemElectroChem*, 2015, **2**, 2027–2031.

85 P. Song, A. C. Fisher, J. D. Wadhawan, J. J. Cooper, H. J. Ward and N. S. Lawrence, *RSC Adv*, 2016, **6**, 70237–70242.

86 J. J. O'Dea, K. Wikiel and Janet. Osteryoung, *J Phys Chem*, 1990, **94**, 3628–3636.

87 M. A. Mann, J. C. Helfrick and L. A. Bottomley, *J Electrochem Soc*, 2016, **163**, H3101–H3109.

88 G. F. Kennedy, J. Zhang and A. M. Bond, *Anal Chem*, 2019, **91**, 12220–12227.

89 H. Chen, E. Kätelhön, H. Le and R. G. Compton, *Anal Chem*, 2021, **93**, 13360–13372.

90 M. Raissi, P. Perdikaris and G. E. Karniadakis, *J Comput Phys*, 2019, **378**, 686–707.

91 E. P. Sapozhnikova, M. Bogdan, B. Speiser and W. Rosenstiel, *Journal of Electroanalytical Chemistry*, 2006, **588**, 15–26.

92 M. K. Zachek, A. Hermans, R. M. Wightman and G. S. McCarty, *Journal of Electroanalytical Chemistry*, 2008, **614**, 113–120.

93 F. Alatraktchi, S. Breum Andersen, H. Krogh Johansen, S. Molin and W. Svendsen, *Sensors*, 2016, **16**, 408.

94 A. Terbouche, S. Lameche, C. Ait-Ramdane-Terbouche, D. Guerniche, D. Lerari, K. Bachari and D. Hauchard, *Measurement*, 2016, **92**, 524–533.

95 Q. Zhao, Z. Gan and Q. Zhuang, *Electroanalysis*, 2002, **14**, 1609–1613.

96 N. S. Prinith and J. G. Manjunatha, *Mater Sci Energy Technol*, 2019, **2**, 408–416.

97 J. F. Rusling and S. L. Suib, *Advanced Materials*, 1994, **6**, 922–930.

98 L. G. Dias, S. G. Meirinho, A. C. A. Veloso, L. R. Rodrigues and A. M. Peres, in *Bioinspired Materials for Medical Applications*, Elsevier, 2017, pp. 371–402.

99 M. Ghasemi-Varnamkhasti, M. L. Rodríguez-Méndez, S. S. Mohtasebi, C. Apetrei, J. Lozano, H. Ahmadi, S. H. Razavi and J. Antonio de Saja, *Food Control*, 2012, **25**, 216–224.

100 K. Tiwari, B. Tudu, R. Bandyopadhyay and A. Chatterjee, *J Food Eng*, 2013, **117**, 205–210.

101 V. Parra, Á. A. Arrieta, J. A. Fernández-Escudero, M. Íñiguez, J. A. de Saja and M. L. Rodríguez-Méndez, *Anal Chim Acta*, 2006, **563**, 229–237.

102 K. Lopetcharat, F. Kulapichitr, I. Suppavorasatit, T. Chodjarusawad, A. Phatthara-aneksin, S. Pratontep and C. Borompichaichartkul, *J Food Eng*, 2016, **180**, 60–68.

103 T. N. Chatterjee, R. B. Roy, B. Tudu, S. Biswas, R. Bandyopadhyay, P. Pramanik and N. Bhattacharyya, in *2016 2nd International Conference on Control, Instrumentation, Energy & Communication (CIEC)*, IEEE, 2016, pp. 60–63.

104 A. Roselló, N. Serrano, J. M. Díaz-Cruz and C. Ariño, *Electroanalysis*, 2021, **33**, 864–872.

105 M. Palit, N. Bhattacharyya, S. Sarkar, A. Dutta, P. K. Dutta, B. Tudu and R. Bandyopadhyay, in *2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems*, IEEE, 2008, pp. 1–6.

106 S. Acharya, T. N. Chatterjee, S. Mukherjee, D. Das, R. B. Roy, B. Tudu and R. Bandyopadhyay, in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, ed. H. Bhattacharyya, S and Snasel, V and De, D and Pan, I and Dhar, S and Bhattacharjee, D and Bhaumik, IEEE, 2018, pp. 108–111.

107 P. Oliveri, M. A. Baldo, S. Daniele and M. Forina, *Anal Bioanal Chem*, 2009, **395**, 1135–1143.

108 X. Cetó, J. M. Gutiérrez, M. Gutiérrez, F. Céspedes, J. Capdevila, S. Mínguez, C. Jiménez-Jorquera and M. del Valle, *Anal Chim Acta*, 2012, **732**, 172–179.

109 P. Rattanawarinchai, P. Krongkrachang, T. Chodjarusawad and D. Phromyothin, *Mater Today Proc*, 2017, **4**, 6410–6414.

110 X. Cetó, J. M. Gutiérrez, A. Mimendia, F. Céspedes and M. del Valle, *Electroanalysis*, 2013, **25**, 1635–1644.

111 X. Cetó, J. M. Gutiérrez, L. Moreno-Barón, S. Alegret and M. del Valle, *Electroanalysis*, 2011, **23**, 72–78.

112 J. M. Gutiérrez, Z. Haddi, A. Amari, B. Bouchikhi, A. Mimendia, X. Cetó and M. del Valle, *Sens Actuators B Chem*, 2013, **177**, 989–996.

113 Q. Niu, S. Chen, L. Wang, Y. Hui and Q. Wang, in *2015 IEEE International Conference on Information and Automation*, IEEE, 2015, pp. 588–593.

114 Y. Yu, H. Zhao, G. Dong, R. Yang, L. Li, Y. Liu, H. Wu and W. Zhang, *Int J Electrochem Sci*, 2015, **10**, 10119–10131.

115 R. Bhattacharyya, B. Tudu, S. C. Das, N. Bhattacharyya, R. Bandyopadhyay and P. Pramanik, *J Food Eng*, 2012, **109**, 120–126.

116 N. Liu, Y. Liang, J. Bin, Z. Zhang, J. Huang, R. Shu and K. Yang, *Food Anal Methods*, 2014, **7**, 472–480.

117 S. N. Dean, L. C. Shriver-Lake, D. A. Stenger, J. S. Erickson, J. P. Golden and S. A. Trammell, *Sensors*, 2019, **19**, 2392.

118 T. G. Mayerhöfer and J. Popp, *ChemPhysChem*, 2019, **20**, 511–515.

119 G. Casella and R. L. Berger, *Statistical Inference*, 2nd edn., 2002.

120 J. N. Goetz, A. Brenning, H. Petschko and P. Leopold, *Comput Geosci*, 2015, **81**, 1–11.

121 D. Bzdok, N. Altman and M. Krzywinski, *Nat Methods*, 2018, **15**, 233–234.

# Chapter 2

## Theoretical Aspects of Single-Molecule Science and Machine Learning

Contents

## 2.1. Charge Transport

Charge transport through single molecules is governed by multiple factors that include dependencies on temperature, molecular structure, coupling strength to electrodes, the electrode materials, and the applied potential difference. Given that the size of single molecules is on the nanoscale, the phenomena of quantum mechanics have an important effect on conductance.

### 2.1.1. Quantum Mechanics

The first step in understanding the factors that affect SM charge transport is to first understand the behaviour of electrons. In quantum mechanics, a particle is represented by an abstract vector, $|\psi\rangle$, which contains all the possible information of every observable quantity that particle can possess. This vector is provided meaning by expressing it in terms of an observable quantity's eigenbasis.[1] For example, the energy eigenbasis of a quantum particle is simply a linear combination of that observable's possible values:

$$|\psi\rangle = \sum_i c_i |E_i\rangle \qquad 2.1$$

where $c_i$ is proportional to the probability that the quantum particles energy equals the energy eigenvalue of the energy eigenvector, $|\mathrm{E_i}\rangle$. This is known as a superposition of quantum states. However, when observables have a continuous range of possible values, the quantum state is instead written as an integral. For example, consider the position observable:

$$|\psi\rangle = \int \psi(x)|x\rangle dx \qquad 2.2$$

where $\psi(x)$ represents a continuous function of all possible coefficients of corresponding definite position, $x$, states.[1,2]

As it was previously stated, the coefficients $c_i$ and $\psi(x)$ are related to the probability that the quantum state is in the corresponding definite states. Indeed, the Born rule states that these probabilities are the squared magnitudes of the coefficients.[3] As a result, the squared magnitude of the position coefficient function, $\psi(x)$, yields a probability density function of the quantum particles position.

$$\rho(x) = |\psi(x)|^2 \tag{2.3}$$

Overall, an observable is represented by a linear map which maps the quantum state, $|\psi\rangle$, to the corresponding observable's eigenbasis. Consequently, if the quantum state is in a definite energy state, mapping $|\psi\rangle$ would return the same vector scaled by the energy value:

$$\hat{E}|\psi\rangle = E|\psi\rangle \tag{2.4}$$

, where $\widehat{\mathrm{E}}$ represents the energy map known as an operator. More likely, the quantum state is instead in a superposition of possible energy states. Instead, the expected value is given by:

$$\langle E\rangle = \langle\psi|\hat{E}|\psi\rangle \tag{2.5}$$

, which is the inner product of the quantum state, $|\psi\rangle$, and the mapped quantum state, $\widehat{\mathrm{E}}|\psi\rangle$. This is the same for any observable value of the quantum particle.

### 2.1.2. Particle-in-a-Box

Consider the traditional example of a quantum particle in a 1D box (Figure 2.1). Inside the box the particle would have a potential energy less than its total energy. Outside the box, the particle would have a constant potential energy greater than the total energy. Classically, total energy, $E$, of the particle is determined by the following expression:

$$E = U + T \tag{2.6}$$

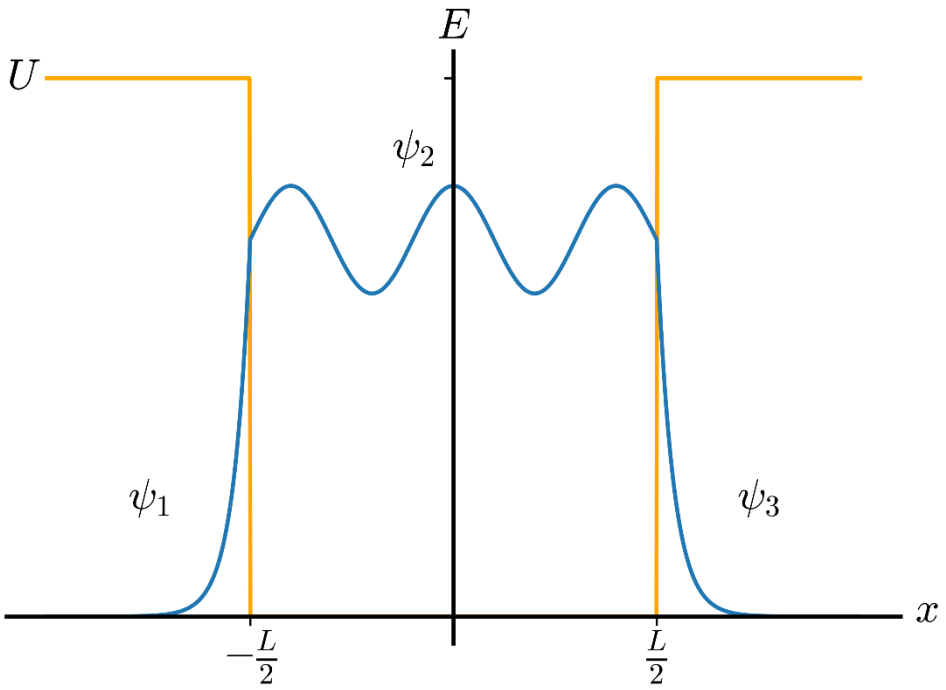



*Figure 2.1*

*Schematical representation of a particle in a 1D box with non-infinite potential regions. With a finite potential there is a non-zero value for the wavefunction in the classically forbidden regions.*

where $U$ is the potential energy, and $T$ is the particle's kinetic energy. Considering this equation and that it is not possible for a particle to possess a negative kinetic energy, it can be concluded that, classically, the particle cannot exist in any region where $E < U$. However, in quantum mechanics, this is not true.

To model the possible position of a quantum particle, the position coefficient function, $\psi(x)$, is required. To find the nature of this function, which is also known as the wavefunction, the time-independent Schrödinger equation is solved for the current system.[4] This is done by using the boundary conditions of the system and then solving the following:

$$E\psi(x) = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + U(x)\psi(x) \qquad 2.7$$

where $\hbar$ is the reduced Planck's constant, $m$ is the particles mass, and $U(x)$ is the potential energy function.

For the particle in a box model, there are three different regions of interest (ROIs). Each of these three different regions will possess separate wavefunctions. These are $\psi_1$, $\psi_2$, and $\psi_3$, for the left, middle and right regions, respectively. When solving Equation *2.7*, the three wavefunctions take the following forms:

$$\psi_1(x) = \psi_1(0)e^{Kx} \quad 2.8$$

$$\psi_2(x) = \psi_2(0)e^{\pm ikx} \quad 2.9$$

$$\psi_3(x) = \psi_3(0)e^{-Kx} \quad 2.10$$

where $k$ and $K$ are constants of the form:

$$k = \frac{\sqrt{2m(E-U)}}{\hbar} \quad 2.11$$

$$K = \frac{\sqrt{2m(U-E)}}{\hbar} \quad 2.12$$

This result is interesting because the wavefunctions for the classically forbidden regions is non-zero. Incidentally, this causes the resultant probability density function to be non-zero also. The particle has a probability to exist within these classically forbidden regions.5

### 2.1.3. Quantum Tunnelling

A junction in conductive SM science can be modelled as two wells with a thin but high potential barrier between them. This is analogous to placing two boxes next to each other in terms of the previous example. Previously, it was shown that a quantum particle could exist in a potential barrier. Therefore, if the length of the potential barrier is thin enough such that the wavefunction has not decayed considerably, it can be concluded that the particle has a

non-zero probability of passing through this potential barrier. This is indeed the case and is a phenomenon known as quantum tunnelling.[6]

To get to the formulism of an electric tunnelling current, the variable of time needs to be introduced. As electric current is a measure of how much charge passes per unit time, to get to a similar property in quantum mechanics it needs to be stated how a quantum particle evolves through time. More specifically, it would be necessary to define how the particles position changes with respect to time. Here this is represented by a probability current which is related to how the probability density changes in time:

$$J(x,t) = \frac{\hbar}{2mi}\left(\psi(x,t)^* \frac{\partial \psi(x,t)}{\partial x} - \psi(x,t)\frac{\partial \psi(x,t)^*}{\partial x}\right) \quad 2.13$$

$$\frac{\partial \rho(x,t)}{\partial t} = -\frac{\partial J(x,t)}{\partial x} \quad 2.14$$

where $J$ represents the probability current, $t$ represents time, and $*$ represents the complex conjugate.[4]

In the previous example of the particle in a box, the wavefunctions were found by utilising the time independent Schrödinger equation. This form of the equation works on the assumption that the particles position probability distribution is not dependent on time. The resultant states are thus referred to as stationary states which possess definite energies.[4] In a stationary state, the change in the probability density function of position with respect to time is zero. As a consequence of this, using Equation *2.14*, the probability current does not change with respect to position. In summary, the probability current is constant and is calculated as follows:

$$J = \frac{\hbar k}{m}\rho \qquad 2.15$$

where $\rho$ is the amount of probability density per unit of position.

The resultant probability current will result in an electric current if the quantum particle has a charge. The resultant current is therefore calculated as:

$$I = qJ \qquad 2.16$$

where $q$ is the particles charge.

Previously, it was shown that a particle in a classically allowed region will possess a wavefunction in the form shown in Equation *2.9*. In this case the constant, $k$, is related to the particle's momentum. Therefore, a positive value for $k$ would indicate a particle with momentum in the positive direction (left to right). Inversely a negative $k$ value would indicate travel in the opposite direction (right to left). Now consider the tunnelling case as shown in Figure 2.2. Here we consider the situation of a charged particle travelling from left to right (in the positive direction). The initial state of this particle can be described by the wavefunction, $\psi_1$. Within the same left region, another wavefunction, $\psi_2$, can exist with the opposite momentum. There are then two more wavefunction states. The state of the particle within the tunnelling barrier, $\psi_3$, and the state of the particle after the tunnelling barrier, $\psi_4$. Each of these states and the tunnelling barrier schematic are given in Figure 2.2. In this perspective, it can be concluded that the state $\psi_4$ is the state that a particle would be in after successfully tunnelling through the barrier. Using this logic and the previous definition of electric current, the overall tunnelling current can be defined in terms of the initial current:

$$I_T = I_{in}T \qquad 2.17$$

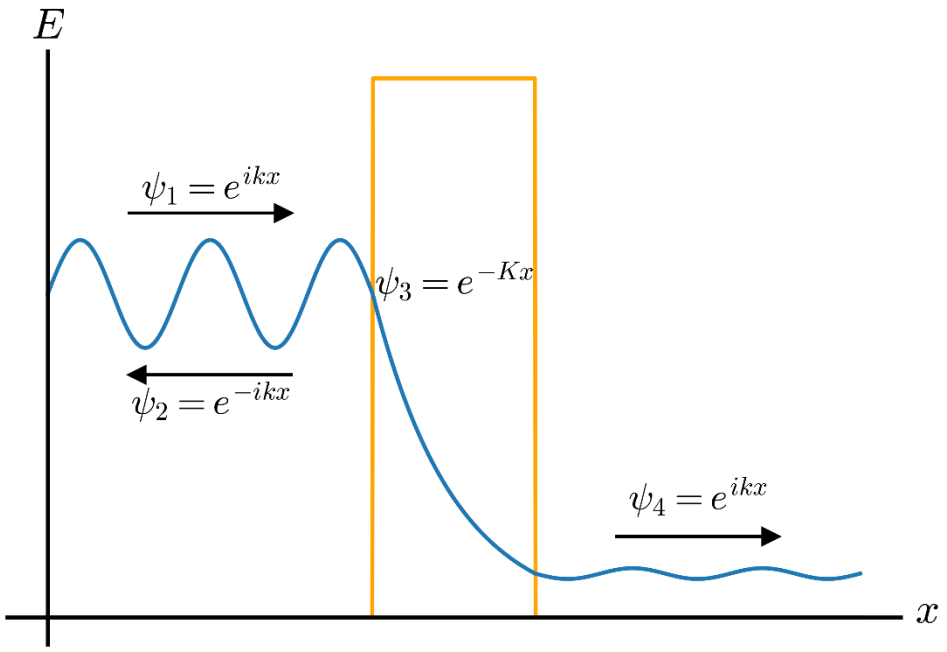



*Figure 2.2*

*Schematical representation of tunnelling though a 1D potential barrier. Allowed solutions for the wavefunction in each region are shown along with the direction of the particle's momentum.*

where $I_T$ is the electric current resulting from the state $\psi_4$, $I_{in}$ is the electric current resulting from state $\psi_1$, and $T$ is a constant value that represents the ratio of how much of the initial currents remains after tunnelling. This constant is referred to as the barrier's transmission coefficient.[5] The inverse of this ratio can be interpreted as the degree of reflection of particles attempting to tunnel. If a particle reaches the tunnelling barrier but fails to tunnel, the only other possibility is for the particle to travel in opposite direction. Therefore, the transmission coefficient, $T$, can also be written with respect to the state $\psi_2$. More specifically, the electric current of $\psi_2$ is:

$$I_R = I_{in}(1 - T) \qquad 2.18$$

where $I_R$ is the reflected electric current corresponding to state $\psi_2$.

Interestingly, the same model can be applied to the reverse case of a particle tunnelling in the opposite direction. If the tunnelling junction is symmetrical, then the tunnelling current for travel in each direction is the same. This means that there would be no overall tunnelling current that results.

Therefore, to generate an overall tunnelling current some form of bias is required. This behaviour is modelled by the Landauer theory of tunnelling.[5] This theory works with the assumption that each electrode functions as a 1D box like in section 2.1.2 with rectangular potential barriers, and that the Pauli exclusion principle applies. Using these rules, the initial current can be rewritten as:

$$I_{in} = \frac{2e^2}{h}V \qquad 2.19$$

The subsequent tunnelling current is then found via:

$$I_T = \frac{2e^2}{h}VT \qquad 2.20$$

where the constant value, $\frac{2e^2}{h}$, is known as the conductance quantum, $G_0$

However, when the potential difference applied between the two electrodes becomes large enough, the shape of the potential barrier changes to a trapezoid shape. This change in barrier shape means that the Landauer theory of tunnelling, and other direct tunnelling mechanisms, may not be applicable. Indeed, in the extreme situations where the energy level of electrons aligns with the triangular edge of the tunnelling barrier the effective tunnelling barrier length is shorter and means the tunnelling current increases. This form of tunnelling is known as Fowler-Nordheim (FN) tunnelling.[7]

### 2.1.4. Charge Transport through Single Molecules

In the previous explanation of quantum tunnelling, the space between the two conductive electrodes was empty. When a single molecule is present and bridges this space, the tunnelling behaviour changes usually resulting in an increased tunnelling current. Looking at Equation *2.20*, when a molecule binds in this manner, the increased tunnelling current must be due to an increase in the transmission probability, $T$. In the case where the applied bias is small, such that the potential barrier shape is not distorted, the transmission coefficient, $T$, can be written in terms of the apparent barrier height:

$$T = e^{-\beta s} \tag{2.21}$$

$$\beta = \frac{2\sqrt{2m\Phi}}{\hbar} \tag{2.22}$$

where $\beta$ is the decay constant, $s$ is the width of the potential barrier, and $\Phi$ is the apparent barrier height.[5] When a molecule bridges the gap between electrodes, the increase in tunnelling current can be attributed to a decrease in the apparent barrier height, $\Phi$, such that the decay constant, $\beta$, is reduced and the transmission probability, $T$, is greater. In addition, to the barrier height, this expression relates distance dependence to the transmission probability. Overall, combining Equations *2.20* and *2.21* provides the following expression for the tunnelling current:

$$I_s = G_0 V e^{-\beta s} \tag{2.23}$$

where $G_0$ is the aforementioned conductance quantum with a value of $77.48\mu S$.

The reduction in the apparent tunnelling barrier and the increase in the tunnelling coefficient can be attributed to the coupling of the molecular energy levels to the energy levels of the electrodes. When these quantum states couple, electrons may be able to transfer between

the different states if energetically favourable to do so. For instance, consider the case of two electrodes, who act as an electron source and an electron drain, with a molecule present within the separating gap (Figure 2.3a). With an applied bias between the electron source and drain, a net current is induced that will flow across the system. If an energy state of the molecule is present within the energy range between the two electrode fermi energies electrons can transfer from the electron source to the molecule. Likewise, electrons within the molecular state can transfer to the electron drain. The rate at which electrons transfer from the source to the molecule, $\Gamma_s$, and from the molecule to the drain, $\Gamma_d$, represent the coupling of the molecule to the corresponding electrode. The total coupling is written as:

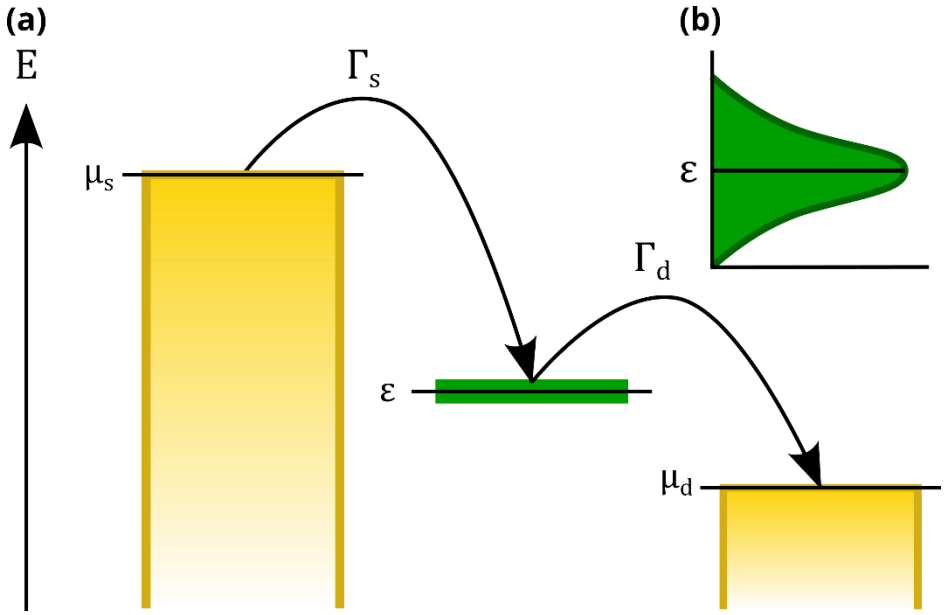



*Figure 2.3*

*Schematic representation of tunnelling involving a molecular bridge. (a) shows the two electron transfer steps from the source electrode with the chemical potential $\mu_s$ to the molecular energy level, ε, and then from the molecular energy level to the drain electrode with chemical potential $\mu_d$. The coupling that coincides with electron transfer leads to the broadening of the molecular energy level, ε, whose energy distribution is shown in (b).*

$$\Gamma = \Gamma_s + \Gamma_d \quad (2.24)$$

This coupling, and thus the rate at which electrons transfer, is a result of the overlap of the quantum states such that their wavefunctions interfere with one another. This wavefunction overlap causes hybridisation of the molecular orbital such that the energy of the molecular state broadens (Figure 2.3b).[8]

The broadening of the molecular level can be explained in terms of the Heisenberg uncertainty principle that links the energy and lifetime of a state. As electrons transfer onto a molecular energy level, there will be a non-zero time that the particle resides on the molecule. Because of the uncertainty relation there will, therefore, be a non-zero degree of uncertainty in the energy of the state.[8,9]

All of these behaviours can be linked to the tunnelling probability through using the Non-Equilibrium Green's Function (NEGF) method which can be written as:

$$T(E) = tr\left(\frac{\Gamma_s(E)\Gamma_d(E)}{\Gamma_s(E) + \Gamma_d(E)}(\boldsymbol{G} - \boldsymbol{G}^{\dagger})\right) \quad (2.25)$$

where $tr$ is the trace function of a matrix, $E$ is the energy of the electron, $G$ is the retarded Green's function, and † is the Hermitian conjugate. The retarded Green's function, $G$, is calculated using molecular information as

$$\boldsymbol{G}(E) = (E\boldsymbol{S} - \boldsymbol{H} - \boldsymbol{\Sigma_s} - \boldsymbol{\Sigma_d})^{-1} \quad (2.26)$$

where $S$ is the symmetric overlap matrix, $H$ is the molecule's Hamiltonian matrix, and $\Sigma$ is the self-energy matrix for the source and drain.[8,10]

### *Coherence of Charge Transport*

The coherence of charge transport relates to how a transport process alters the phase information of a quantum particle. Coherent tunnelling is defined as transport of charge across the tunnelling barrier resulting in no change to the tunnelling particles phase.[11] The previous tunnelling regimes of direct tunnelling and FN tunnelling are examples of coherent tunnelling. However, when the tunnelling regime contains a third reservoir where particles are classically allowed, there is the possibility that electrons may transfer to this third location. In the case where this third location does not act as an electron sink such that the net current only flows across the original junction, the tunnelling electrons can still interact with the third location and randomise their phase information. This defines an incoherent transfer process.[12] In the example of a molecule bridging the tunnelling barrier, the molecule can act as this third electron reservoir. A tunnelling electron can therefore transfer onto the molecule and undergo an inelastic process whereby the particle loses energy, excites the captured molecule, and in the process loses phase information. [13]

Overall, the transmission of electrons across a tunnelling junction containing a molecule manifests as a combination of both coherent and incoherent transmission probabilities:

$$T_{tot} = T_c + T_i \qquad 2.27$$

where $T_{tot}$ is the total transmission probability, $T_c$ is the coherent transmission probability, and $T_i$ is the incoherent transmission probability. An electron incident on the tunnelling junction can either transfer directly through the junction without interaction with the inelastic scatterer or it can transfer sequentially through the scatterer. These two possibilities are not independent meaning the coherent component cannot be modelled by a purely coherent process. [12] As a consequence of this, the charge transfer across a molecular junction becomes

increasingly complex to model. As such, the interested reader is referred to additional sources.[4,14,15]

### *Resonant Tunnelling*

Lastly, there is one notable phenomenon that can arise during charge transfer through a molecular junction. In the above tunnelling regime, there are two tunnelling barriers. One before and one after the inelastic scatterer. When considering the coherent case, there is a non-zero probability of reflection at the second tunnelling barrier. The wavefunction of this reflected particle is then able to interfere with the transmitted particles from the first tunnelling barrier. When the wavefunctions possess the correct wavelengths, this interaction can result in the complete destructive interference of the reflected wavefunction such that the transmission probability becomes unity. The phenomenon is known as resonant tunnelling which explains the higher-than-expected conductance of double tunnelling barriers.[4]

## 2.1.5. Scanning Tunnelling Microscopy

The functionality of STMs, as introduced in Section 1.1.1, require precise control over the electrode separation. This is achieved by employing a system of piezoelectric crystals that control the 3D position of one of the electrodes (Figure 2.4).[5] The electrode under piezoelectric control is the STM tip, which is designed to be sharpened to a fine, ideally monoatomic, point.[16] The fine adjustment of the tip's position is controlled by varying the voltage across any combination of the three piezoelectric crystals. When increasing the potential difference across these crystals their size increases causing the STM tip to be moved in the corresponding cartesian directions.[17] Of the three piezo crystals, two correspond to the

lateral movement of the tip in the $x$ and $y$ directions. The third crystal controls the perpendicular movement in the $z$ direction.

The most widely used mode of the STM is the constant-current mode where the user defines a desired tunnelling current for the STM to maintain. Once the STM tip is close enough to the other electrode's surface and a potential difference is applied such that a tunnelling current can be measured, the STM implements an automated systems for maintaining this tunnelling current. A feedback system takes the measured current and converts this into a voltage that controls the z-piezo such that the tip moves in the appropriate direction to approach the

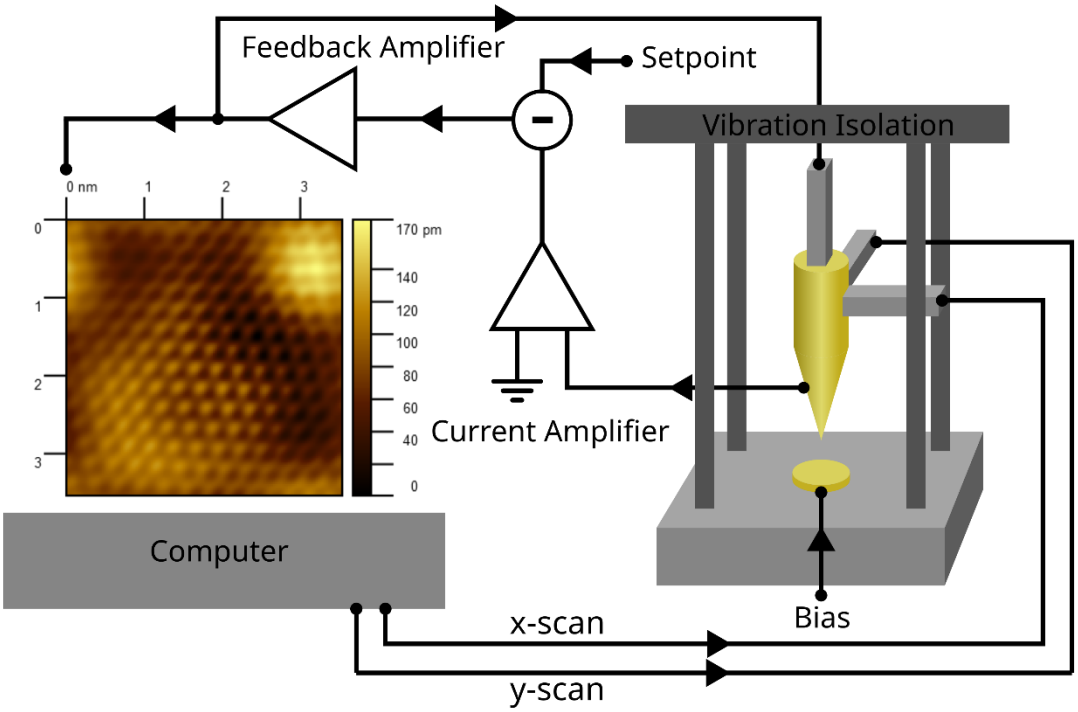



*Figure 2.4*

*Outline of the construction of an STM. The diagram shows a simplified summary of the points of connection between the sample and the controlling computer. In addition, a setup of amplifiers is shown to summarise how the measured signal is communicated and implemented in the feedback system.*

desired setpoint current (Figure 2.4). This feedback system takes the form of a PID controller where three customizable gains can be used to tweak the tip distance behaviour.[18] This control allows the speed at which the tip will reacquire the setpoint current to be customised and ultimately helps with sharpening recorded images or preventing setpoint overshoots.

To produce an image of the electrode surface, the measured current requires amplification and conversion into a digital signal so that a controlling computer can display measurements. This initial amplification of the measured currents is required as the measured tunnelling currents are generally very small on the order of $10^{-9}$A. This amplification also converts the measured currents into a corresponding voltage. Overall, these amplifiers will have gains set that represent a conversion of current to voltage.[5] For example, a gain of 10nA/V would convert a 10nA tunnelling current into a 1V signal. Following this, the analogue voltage signal is converted into a digital signal such that the controlling computer can read and display measured results.

### *Electrochemical Scanning Tunnelling Microscopy*

ECSTM incorporates additional electrodes into the original STM setup to allow for the simultaneous characterisation of the redox behaviour of the tip and substrate electrodes whilst conducting STM experiments. The additional electrodes are referred to as the counter and reference electrodes which totals to a system of four electrodes. When immersed in an ionic solution, this setup allows for fine control over both the tip and substrate electrodes potentials. This relies on the use of a bipotentiostat controller to control the potentials of two working electrodes that share the same counter and reference electrodes.

Overall, within an ECSTM there are three potential differences that are controlled. As with traditional STM the applied bias between the tip and substrate electrode is present. However, with ECSTMs there is the incorporated potential differences between the tip and the reference electrode as well as the substrate and reference electrode. All three of these potential differences are related by the following equation:

$$V = E_{sub} - E_{tip}.^{19} \qquad 2.28$$

where $E_{sub}$ is the substrate potential and $E_{tip}$ is the tip potential. Because of this relationship between the three potentials, there is a continuous range of values for the tip and substrate potentials for a chosen applied bias. This means that the applied bias between the tip and substrate can be kept constant whilst sweeping the tip and substrate potentials in unison. Alternatively, the applied bias can be swept by either fixing the tip or substrate potential whilst varying the potential of the other electrode. This fine control allows for deeper investigations into the electrochemical behaviour of STM electrodes *in situ*.

## 2.2. Machine Learning

The term ML was first used by the American computer scientist Arthur Samuel in 1959. Samuel became a pioneer within the field of AI and ML for their development of a checkers playing agent, which was able to outperform a human competitor.[20] The definition of a ML agent is commonly referred to as a computer program that is able to "*learn to solve problems without being explicitly programmed*" which is a paraphrase that was coined during reference to Samuel's work.[21] Despite the roots of ML starting within game AI, the field has since broadened to encompass data analytics as well.

Nowadays, any ML task can be split into one of three different classes. These include supervised learning tasks, unsupervised learning tasks, and reinforcement learning tasks. Both supervised and unsupervised learning are used extensively in data science, whereas the application of reinforcement learning is generally confined to more niche use cases of agents exploring different states of a system. It is also worth noting that techniques from each section are commonly used in conjunction to improve task performance.

### 2.2.1. Supervised Learning

Supervised learning encompasses any ML technique that involves learning based on pre-labelled data.[22] For instance, consider a dataset $\boldsymbol{A}$ that is comprised of $n$ number of vectors such that

$$x_1, x_2, \dots, x_n \in \boldsymbol{A} \tag{2.29}$$

where

$$x_i \in \mathbb{R}^m \tag{2.30}$$

Each vector within dataset $\boldsymbol{A}$ will have another associated vector to it that acts as each variable's label. The nature of these labels varies based on the supervised learning task. Dataset $\boldsymbol{A}$ can instead be written as

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \in \boldsymbol{A} \qquad 2.31$$

The goal of supervised learning is to employ techniques that can learn the relationship between data and their labels. To put this into mathematical terms, ML in this setting attempts to learn a function $g(x)$ where

$$g(x_i) = y_i \qquad 2.32$$

#### *2.2.1.1. Regression*

Regression is commonly used within data analysis to attempt to approximate the relationship between two variables.[23] In the context of supervised learning this means, using the previous example, the vector $y_i$ would be the variable dependent on the value of vector $x_i$. This subset of supervised learning, whilst not explored within the context of this thesis, is a large field that deserves mentioning.[24–26]

#### *2.2.1.2. Classification*

In ML, classification is the act of separating data into groups. This method is supervised by a training set of data where the label $y_i$ for data point $x_i$ represents a class label.[27] An agent would use this training set to learn the mapping function between data and their class labels. There are numerous methods that can be used to model this mapping process such that the class label, $y_i$, for any data point can be predicted. The models mentioned in this thesis are described in the following section.

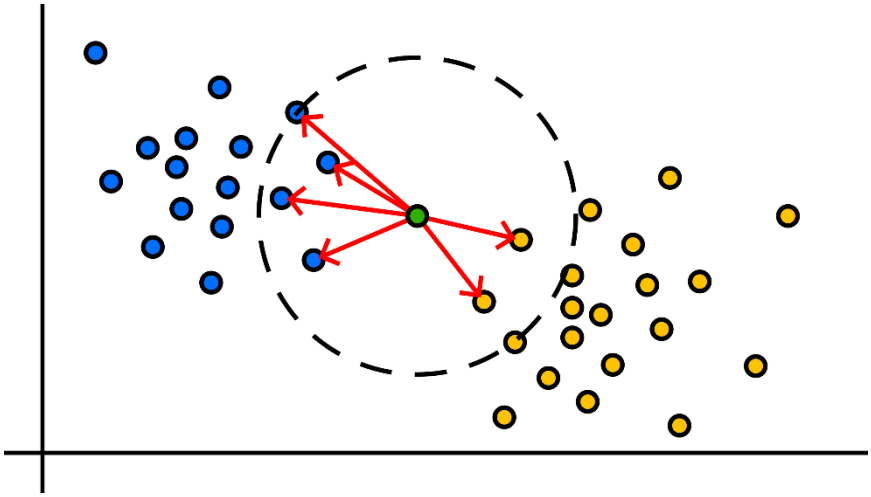


*Figure 2.5*

*Demonstration of classifying an unknown point (green) using the KNN algorithm. Here a $k$-value of six is shown which would result in the point being classified as belonging to the left (blue) class.*

### k-Nearest Neighbour Classifier

k-Nearest Neighbour (KNN) is one of the simplest methods of classifying data. This method works by placing any unknown data points within the same mathematical basis as the training data. The label of the unknown point is then determined by looking at a user defined number of neighbouring points, $k$. The training label that is most prevalent amongst the neighbours decides the label given to the unknown (Figure 2.5).[28]

### Naïve Bayesian Classifier

The Naïve Bayesian Classifier is another simple method that uses Bayesian probabilities to determine the probability that an unknown data point is part of a class. As such, prior knowledge about available classes, and the distributions of both classes and data are required. The equation that links these variables is the famous Bayes theorem and is defined as

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)} \qquad 2.33$$

where $P(C_i|x)$ is the posterior probability of assigning the class label $C_i$ to point $x$. This posterior probability is calculated for each possible value of $C_i$. $P(x|C_i)$ is known as the likelihood probability and is the probability of sampling the value $x$ within the distribution of class $C_i$. $P(x)$ and $P(C_i)$ are known as the prior probabilities, ($P(x)$ is also sometimes known as the evidence). These values are the probabilities of sampling the value $x$ within the whole distribution of $x$, and sampling class $C_i$ out of all classes, respectively. Upon calculation of all posterior probabilities, the predicted class for point $x$ is selected using the most probable probability.[29]

### Decision Tree Classifier

The decision tree classifier takes an information-based approach to classification. The classifier adopts a binary tree structure with checks at each node to partition the input data. The algorithm starts with a single node called the root node. Here the algorithm decides a linear threshold on one of the datasets features and bins the data into two groups (Figure 2.6a). The algorithm uses the training labels from both partitions to determine how effective the split was. There are various metrics that can be used to quantise the success of a split, which will be outlined later. The algorithm iterates through possible thresholds on every feature to find the threshold that maximises the information gained. For each of the resultant subpopulations the algorithm recursively applies the same logic to further subdivide populations. This results in the generation of a binary tree of splits until the resultant populations are considered pure, only containing data with the same label (Figure 2.6b). Once completed, unknown data can be processed using the tree of splits to find the predicted class.[30]

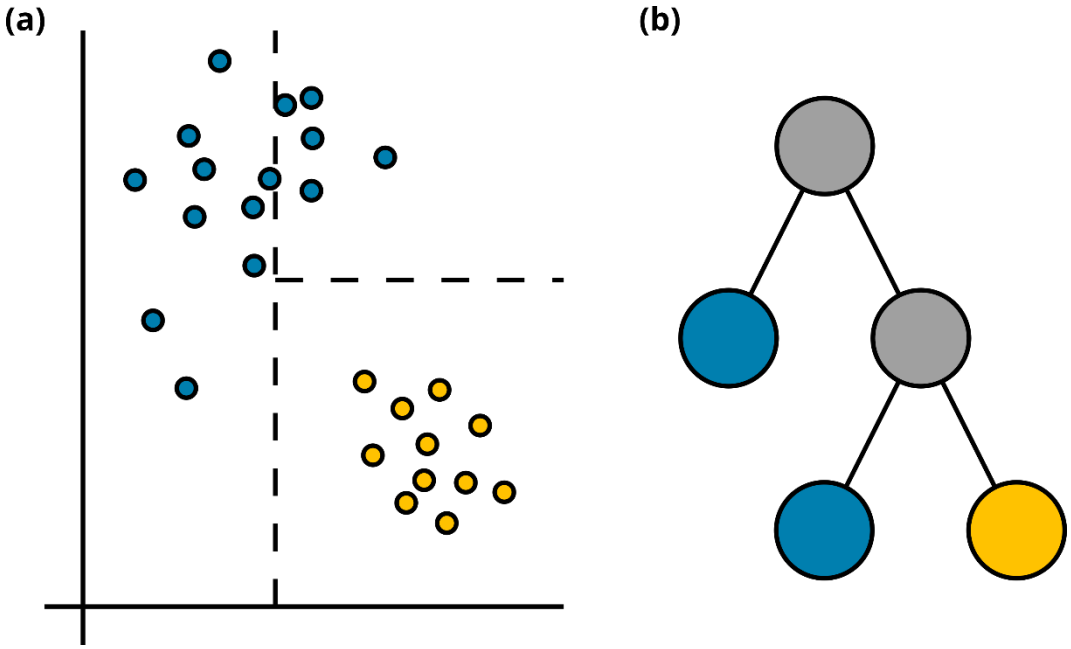



*Figure 2.6*

*Demonstration of decision boundaries within a training dataset, (a), along with the corresponding decision tree structure that would facilitate those boundaries, (b).*

The measure of success for a split is quantified as information gained. Suppose that dataset $\boldsymbol{A}$ is put into the root node of a decision tree. The root node will have an associated split parameter and feature index, $(t, f)$. Using these, the left, $\boldsymbol{L}$, and right, $\boldsymbol{R}$, populations can be assigned.

$$\boldsymbol{L} = \{\boldsymbol{A}_j \mid \boldsymbol{A}_{j,f} \leq t\}, \tag{2.34}$$

$$\boldsymbol{R} = \{\boldsymbol{A}_j \mid \boldsymbol{A}_{j,f} > t\}, \tag{2.35}$$

For each of the parent and the two child populations, a loss function is applied to calculate the entropy/purity of the corresponding population. This function is typically the Gini Impurity but can also take the form of the Shannon entropy. These are

$$H_G(\boldsymbol{A}) = \sum_c p_c \, (1 - p_c) \tag{2.36}$$

and

$$H_S(\boldsymbol{A}) = -\sum_c p_c \log_2 p_c \qquad 2.37$$

respectively. The information gained is then calculated as the average child entropy subtracted from the parent's entropy.[30]

$$IG = H(\boldsymbol{A}) - \frac{n_l}{n} H(\boldsymbol{L}) - \frac{n_r}{n} H(\boldsymbol{R}) \qquad 2.38$$

### Random Forest

The Random Forest model takes the logic of decision trees and improves the model's reliability. Decision trees have a flaw in that they are very sensitive to fluctuations within the training data. So much so that a change in one data point can lead to a completely different tree of splits. This stems from decision trees propensity to be very 'greedy' whereby they attempt to maximise performance on the training data.[31] This leads to a high training accuracy but can result in poor predictive accuracy when being applied to new data. This issue is known as the bias-variance trade-off or overfitting.[32] The random forest model attempts to solve this issue with less need for parameter optimization.

Overall, random forests are ensemble agents which consist of many decision trees.[33] To train these ensembles, the method of bootstrap aggregating (bagging) is employed whereby a series of randomly sampled datasets are generated to train each tree individually. More precisely, for a forest consisting of $B$ trees, $B$ datasets are created by randomly sampling from the training dataset with replacement. Each dataset consists of the same number of datapoints as the original dataset but will contain duplications. After each tree is trained, predictions are made on previously unseen datapoints. In the case of a classifier, the majority

vote from the forest is taken as the overall prediction.[34] Overall, random forest models yield better performance than a single tree as they decrease model variance with marginal increase to the bias.

### Support Vector Machines

SVMs are ML agents that use linear algebra to calculate a hyperplane in an n-dimensional feature space that acts as a decision boundary for classification. As the mathematical inception of this technique is complicated, the idea of a maximal margin classifier will be explained in a binary classification scenario. Additionally, non-linear classification and the 'kernel trick' will briefly be covered.

Consider two parallel lines embedded in the feature space, each anchored to the vector that is closest to the opposing class. It is desirable to find the rotation of these planes such that the distance between them is maximised. The decision boundary for binary classification can then

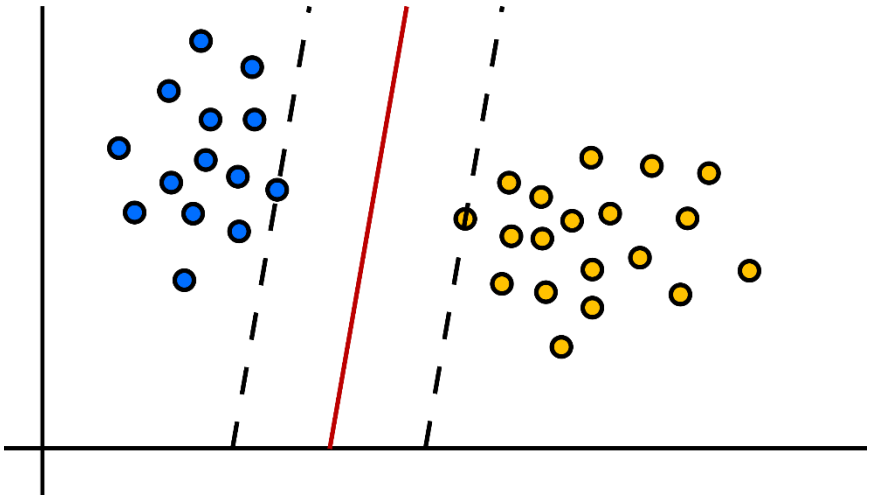


*Figure 2.7*

*Demonstration of the decision boundary, (red), that would be found by applying SVM to the example training set. This boundary would be found using the two parallel support vectors which reside on the dotted margin lines.*

be set as a third hyperplane this is equidistant to each classes' hyperplane. This is demonstrated in Figure 2.7.

The equation of a hyperplane in n-dimensional space is written as:

$$\vec{r} \cdot \vec{w} + b = 0 \tag{2.39}$$

where $\vec{r}$ is any vector that lies on the plane, $\vec{w}$ is the planes normal vector, and $b$ is a constant related to the distance along $\vec{w}$ from the origin that the plane is located. Using this, a classification rule can be created such that if an unknown vector $\vec{u}$ is on one side of the plane, it's dot product with the planes normal vector, $\vec{w}$, plus the offset, $b$, is not equal to zero.

$$\vec{u} \cdot \vec{w} + b = \begin{cases} 1, & \geq 0 \\ -1, & < 0 \end{cases} \tag{2.40}$$

The above cases can be combined into a general form:

$$y(\vec{x} \cdot \vec{w} + b) - 1 = 0 \tag{2.41}$$

where $y$ is the class label of vector $\vec{x}$. If this general form is applied to training data, boundary conditions can be set that aid in determining $\vec{w}$.[35]

The above equation, Equation *2.41*, is true for the vectors that lie on the margin edges. These vectors are known as the support vectors. Given that the goal is to maximise the width of the margin between two hyperplanes, these support vectors can be used to aid in determining $\vec{w}$. The width of the margin would be determined as

$$Width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|} \tag{2.42}$$

where $\vec{x}_+$ and $\vec{x}_-$ are the support vectors for the positive and negative classes.

The best classifier would be the one whose $\vec{w}$ vector maximises the margin width. Using linear algebra and by plugging in the labels for the boundary conditions into Equation *2.41*, we find that the margin width is inversely proportional to the magnitude of the normal vector $\vec{w}$ only.[35]

Using the above logic, optimisations can be run using the method of Lagrange multipliers to find that $\vec{w}$ is the weighted sum of the training vectors,

$$\vec{w} = \sum_i \alpha_i y_i \vec{x}_i \qquad 2.43$$

where $\alpha_i$ is the weight associated with vector $\vec{x}_i$ as found by running the Lagrange optimisation. Therefore, when classifying new points, the predicted label is determined using the dot product of the unknown vector with all training vectors whilst considering their weights and labels.[36]

$$y_u = sgn\left(\sum_i (\alpha_i y_i \vec{x}_i \cdot \vec{u}) + b\right) \qquad 2.44$$

Naturally, this method only works when the two classes are linearly separable. Therefore, to solve a non-linear case, the idea was developed where the vector space is transformed into a non-linear space where the classes are linearly separable. The classification would be carried out in this new vector space before transforming back to the original feature space. This would result in non-linear decision boundaries. The most notable development with this in mind is the innovation of the "kernel trick".

The "kernel trick" is a mathematical technique that allows the non-linear hyperplane to be calculated without mapping the feature space to that non-linear vector space.[37] Previously, in

Equation *2.44*, it was shown that the prediction label, $y_u$, is dependent on the dot product of pairs of vectors. That means in order to classify unknown data points only the dot products are required. The kernel trick calculates these dot products in the following way:

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j) \tag{2.45}$$

where $k$ is the kernel function of a vector space, and $\varphi$ is the mapping function to that vector space. Factoring this into the previous classification equation yields:

$$y_u = sgn\left(\sum_i (\alpha_i y_i k(\vec{x}_i, \vec{u})) + b\right) \tag{2.46}$$

The common kernel functions that are used are the polynomial and radial basis functions and are as follows:

$$k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^n \tag{2.47}$$

$$k(\vec{x}_i, \vec{x}_j) = e^{-\frac{\|\vec{x}_i - \vec{x}_j\|}{\sigma}} \tag{2.48}$$

Overall, this trick allows non-linear decision boundaries to be set whilst avoiding the need to map the vector space into another non-linear space by directly calculating the dot products.

### 2.2.2. Unsupervised Learning

In the previous example, the processing of labelled data in the form $(x_i, y_i)$ was explored. For unsupervised learning, however, data is not labelled. As such, the example dataset as shown in Equation *2.29* will be used to demonstrate unsupervised learning. The goal of unsupervised learning is to find information about the structure of a dataset to find trends or groups that help to understand the nature of the data.[38]

#### 2.2.2.1. Dimensionality Reduction

Previously, in Section 1.3, it was stated that working with datasets that contain a large variety of interrelated variables is generally difficult from a statistical inference point of view. However, the difficulties of working with feature rich data are not only limited to statistical inference. There is a plethora of phenomena that arise when working with these types of datasets. Collectively these phenomena are said to be consequences of the "Curse of Dimensionality" (CoD).[39] As an example, one of the phenomena relates to the sparsity of data within high dimensional space. It can be shown that when datasets are embedded in a high number of dimensions, the measure of Euclidean distance breaks down. This is because as the number of dimensions increased datapoints appear to become more spread out. To demonstrate this, three datasets each containing 10,000 datapoints were created. Each of these datasets were sampled from a normal distribution with mean and standard deviation of 0 and 1, respectively. Each distribution varied by their respective dimensionalities. The pairwise Euclidean distances for each dataset were subsequently calculated before being demonstrated on a histogram (Figure 2.8). Here it can be seen that datapoints within the same normal distribution become more spread when increasing the number of dimensions. For more details on the other aspects of the CoD, the interested reader is referred to additional sources.[39,40]

Due to the various challenges of analysing high dimensional spaces, various algorithms have been devised with the intention of reducing the number of dimensions within a dataset. The collective term for the methods by which this is achieved is dimensionality reduction (DR). DR aims to reduce the dimensionality of a given dataset whilst maintaining the information stored in the original high dimensionality basis.[41] The degree of information that is retained is

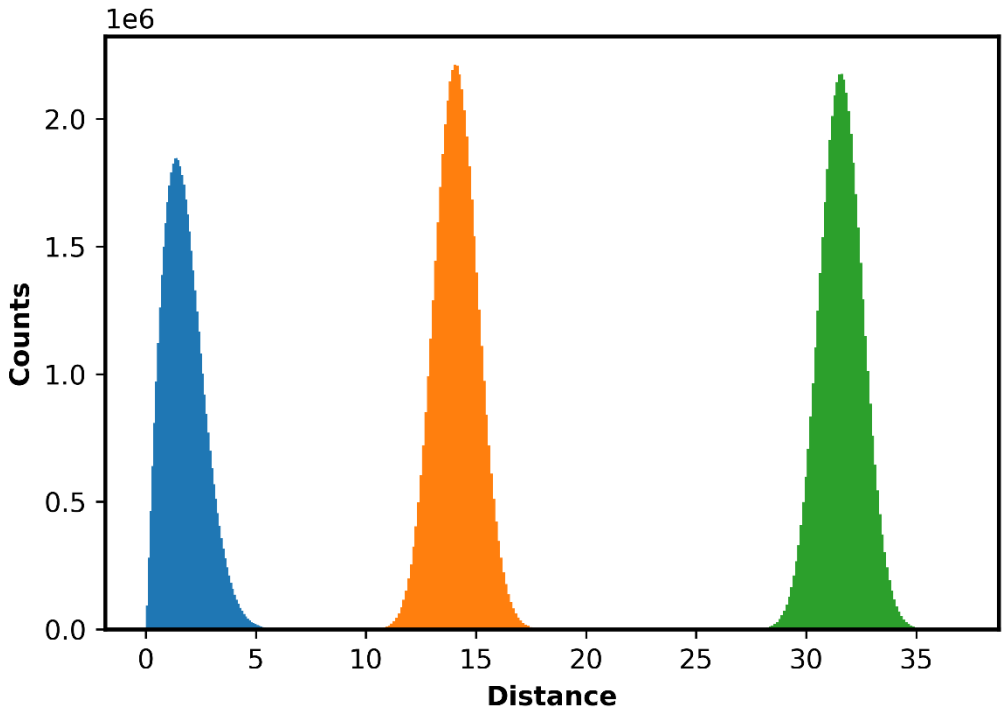



*Figure 2.8*

*Comparison of pairwise Euclidean distances in different dimensional bases. The pairwise distances are shown for dimensionalities 2 (blue), 100 (orange), and 500 (green).*

commonly quantified by comparing dimensional variances and neighbourhood structures of data points within both the original high dimensional space and the returned low-dimensional representation. Once achieved successfully, DR alleviates the aforementioned CoD and allows high dimensional datasets to be more easily visualised and presented. The various methods by which DR is achieved within this thesis are outlined below.

### Principal Component Analysis

PCA is a DR technique that transforms a dataset into a different vector space where the original variance is described in fewer dimensions.[42] The first step in this process is to organise the dataset of study into a matrix such that each entry is a row vector. The matrix representation of dataset $\boldsymbol{A}$ appears as follows

$$\boldsymbol{A} = \begin{bmatrix} \cdots & \vec{x}_1 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \vec{x}_n & \cdots \end{bmatrix} \tag{2.49}$$

Subsequently the dataset is centred such that the average value for each feature is $0$. The average value of each column in $\boldsymbol{A}$ is subtracted from each entry in the corresponding column. This is mathematically written as

$$\boldsymbol{B} = \boldsymbol{A} - \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \overline{x} \tag{2.50}$$

where $\boldsymbol{B}$ is the mean-centred dataset and $\overline{x}$ is the vector containing the average values of each feature:

$$\overline{x}_j = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{A_{ij}} \tag{2.51}$$

where $n$ is the total number of entries within dataset $\boldsymbol{A}$.

The mean-centred data is used to compute the covariance matrix, $\boldsymbol{C}$, where the diagonal entries of said matrix represent the variances of each feature, and the off-diagonal entries represent the covariances of the corresponding features. This is computed by scaling the inner product of the mean-centred data matrix with itself:

$$\boldsymbol{C} = \frac{1}{n-1} \boldsymbol{B}^T \boldsymbol{B} \tag{2.52}$$

This covariance matrix is used to find some transformation into another vector space where the new covariance matrix has a maximised form. This maximised form for a covariance matrix has minimised covariances between features and additionally maximises the variance within fewer dimensions. Additionally, this transformation will maintain the neighbourhood distances between points within the original feature space.

To calculate the new vector space, the eigen decomposition of the covariance matrix is performed. To carry out this decomposition, eigenvectors and eigenvalues are found such that the following equation is respected:

$$\boldsymbol{CV} = \vec{\lambda}\boldsymbol{V} \tag{2.53}$$

where $\boldsymbol{V}$ is a matrix containing the eigenvectors of $\boldsymbol{C}$, and $\vec{\lambda}$ is a vector of $\boldsymbol{V}$'s corresponding eigenvalues. Using these vectors, the original dataset can be projected into this new coordinate system where the axes align with trends in the data. [43]

It is at this point where the DR is applied. Given that the data is aligned as such, some of the features of this principal component (PC) space can be selected for removal without losing vast amounts of variance information within the data. For clarification, the eigenvectors whose eigenvalues do not take up a large proportion of the total value of all eigenvalues can be removed. As the eigen decomposition was carried out on the covariance matrix, the eigenvalues correspond to the variance in the new PC space. With this knowledge a percentage measure of information loss when removing components can be defined:

$$VE = \frac{\sum_{i=1}^{r} \vec{\lambda}_i}{\sum_{j=1}^{N} \vec{\lambda}_j} \times 100 \tag{2.54}$$

where $VE$ is the variance explained as a percentage, $r$ is the number of components that are kept, and $N$ is the total number of PCs.[44]

### t-distributed Stochastic Neighbour Embedding

t-distributed Stochastic Neighbour Embedding (t-SNE) is the second iteration of a DR technique known as Stochastic Neighbour Embedding (SNE). The aim of t-SNE's development

was to fix two issues that were part of the SNE algorithm. Therefore, to explain the working of t-SNE, the SNE algorithm and its limitations will be explained first.

The motivation behind SNE is to use a different metric of distance other than Euclidean distance for the high-dimensional data. This is because, as discussed previously, the Euclidean distance breaks down at higher dimensions.[45] Therefore, SNE employs pairwise probabilities to all points where each probability is the likelihood that a point would pick another point as its neighbour. Then the algorithm would place data points into low-dimensional space and use a gradient descent algorithm to reconstruct the same pairwise probabilities.

Let the high-dimensional pairwise probabilities be defined as so:

$$\boldsymbol{P} = \begin{bmatrix} 0 & p_{0|1} & \cdots & p_{0|n} \\ p_{1|0} & 0 & \cdots & p_{1|n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n|0} & p_{n|1} & \cdots & 0 \end{bmatrix} \tag{2.55}$$

where $\boldsymbol{P}$ is the high-dimensional pairwise probabilities as a matrix, $p_{j|i}$ is the probability that point $x_i$ would pick point $x_j$ as its neighbour, and $n$ is the total number of datapoints. The pairwise probabilities are assigned using Gaussian distributions centred on each point:

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)} \tag{2.56}$$

where $\sigma_i^2$ is the variance of the distribution centred on point $x_i$. This variance is also known as the kernel width. It is worth noting here that the kernel width is not necessarily the same for every point. This is because SNE and t-SNE rely on a user-defined value called perplexity to assign the kernel width. This value takes on integer values which roughly correspond to the

number of neighbours which are encapsulated by the Gaussian. For each point in the dataset, the algorithm iteratively finds the kernel width which corresponds to the user given perplexity. The perplexity of a point is calculated from the pairwise probabilities as follows:

$$Perp(\boldsymbol{P}_i) = 2^{-\sum_j \boldsymbol{P}_{ij} \log_2 \boldsymbol{P}_{ij}} \tag{2.57}$$

where $\boldsymbol{P}_i$ is the matrix slice of $\boldsymbol{P}$ that contains all the probabilities of a point being neighbours with point $x_i$. In the original paper, they employed binary searches to find the kernel widths for each point such that the resultant probability matrix, $\boldsymbol{P}$, also results in a $Perp(\boldsymbol{P}_i)$, for every value of $i$, that equals the user's desired perplexity value.[46]

Once the probability matrix, $\boldsymbol{P}$, is assigned, it is at this point where the first fix of t-SNE is implemented. In SNE, this matrix is not symmetrical meaning $\boldsymbol{P}_{ij} \neq \boldsymbol{P}_{ji}$.[46] This is problematic as it leads to outliers in the high-dimensional space having very little impact on the gradient descent optimization that occurs later in the algorithm. This issue is alleviated by manually making the matrix symmetrical by applying

$$\boldsymbol{P}_{ij} = \frac{\boldsymbol{P}_{ij} + \boldsymbol{P}_{\boldsymbol{ji}}}{2n} \tag{2.58}$$

to every combination of $i$ and $j$.[47]

Once the probability matrix is symmetrised, the dataset is initialised in a lower dimensional vector space. Let the low dimensional representation of point $x_i$ be labelled as $y_i$. At first, the points, $y_i$, are placed at random points in space. From here a second pairwise probability matrix, $\boldsymbol{Q}$, is calculated. In SNE, this matrix is calculated by using Gaussian distributions again but with a fixed kernel width of $1/\sqrt{2}$. However, this leads in to the second issue and fix that is remedied by t-SNE. The use of Gaussians in the lower dimension leads to what is known as

"the crowding problem" where the lower-dimensional space cannot accurately recreate mid-range pairwise probabilities that are present in the higher-dimensional space.[47] Therefore, t-SNE implements a Student t-distribution to exaggerate the tails of the distribution in the lower-dimensional representations. As such $\boldsymbol{Q}$ is populated as:

$$\boldsymbol{Q_{ij}} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_l \sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \tag{2.59}$$

With matrices for both the high-dimensional and low-dimensional spaces, a score is calculated to signify how well $\boldsymbol{Q}$ represents $\boldsymbol{P}$. This score is calculated by applying a cost function. In the case of t-SNE, the Kullback-Leibler (KL) divergence is used which is defined as

$$C = \sum_i \sum_j \boldsymbol{P_{ij}} \log \frac{\boldsymbol{P}_{ij}}{\boldsymbol{Q}_{ij}} \tag{2.60}$$

Ultimately, the low-dimensional positions are updated such that the resulting cost is minimised. To do this the gradients of $C$ with respect to each point, $y_i$, is calculated:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (\boldsymbol{P}_{ij} - \boldsymbol{Q}_{ij})(y_i - y_j)\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1} \tag{2.61}$$

Then gradient descent is applied to improve the values of $y_i$:

$$y_i^{(t)} = y_i^{(t-1)} + \eta \frac{\delta C}{\delta y_i} + \alpha(t)\left(y_i^{(t-1)} - y_i^{(t-2)}\right) \tag{2.62}$$

where $\eta$ is the learning rate of the optimiser, and $\alpha(t)$ is the momentum of the descent in iteration $t$.[47]

### Uniform Manifold Approximation and Projection

Uniform Manifold Approximation and Projection (UMAP) is another DR technique that functions very similarly to t-SNE. However, the theoretical foundations of UMAP reside in the

mathematical field of topology. Due to the abstract nature of topology, this explanation will cover the intuitions without extensive exploration of the lower-level topological mathematics.

Initially UMAP represents the high-dimensional neighbourhood structure within a weighted graph. To construct this graph, the algorithm computes $n$ sets, one for each datapoint, that contain the KNNs for that point. The weights of the connections between each point and its nearest neighbours are given by the following equation:

$$w\left(\left(x_i, x_{i_j}\right)\right) = \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right) \qquad 2.63$$

where $x_i$ is the point of interest, $x_{i_j}$ is the $j$th nearest neighbour of point $x_i$, and $d$ is the distance function of the high-dimensional topological space. The constants $\rho_i$ and $\sigma_i$ are the local-connectivity constraint and the smoothing factor, respectively. Both constants are calculated for each point by

$$\rho_i = \min\left\{d\left(x_i, x_{i_j}\right) \middle| 1 \leq j \leq k, d\left(x_i, x_{i_j}\right) > 0\right\} \qquad 2.64$$

and by finding a $\sigma_i$ such that

$$\sum_{j=1}^{k} \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right) = \log_2(k) \qquad 2.65$$

which serves to normalise the connection weights.[48]

This step is functionally the same as the t-SNE step of constructing pairwise probabilities, except here the probabilities are calculated using a different metric. The subsequent step is also similar to t-SNE in that the adjacency weights required symmetrisation. Let $\boldsymbol{A}$ be an

adjacency matrix whose elements are the weights calculated using Equation *2.63*. UMAP creates a symmetrical matrix, $\boldsymbol{B}$, by applying the following:

$$\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{A}^T - \boldsymbol{A} \otimes \boldsymbol{A}^T \quad 2.66$$

where $\otimes$ is the element-wise multiplication operator known as the Hadamard product.

From here a low-dimensional representation is initialised. This placement can be random,[48] however UMAP, by default, uses another DR technique known as Spectral Embedding to initialise the low-dimensional points. For more information on Spectral Embedding, the interested reader is directed to additional reading resources.[48,49] From here, UMAP generates a low-dimensional adjacency map using the following equation:

$$\Phi(x, y) = \frac{1}{1 + a(\|x - y\|^2)^b} \quad 2.67$$

where $\Phi(x, y)$ represents the probability that connection of point $x$ to point $y$ exists, and $a$ and $b$ are constants calculated by fitting the equation $\Phi(x, y)$ to the results of another equation, $\Psi(x, y)$ where

$$\Psi(x, y) = \begin{cases} 1, & if\ \|x - y\| \leq \min\_dist \\ \exp(-(\|x - y\| - min\_dist)), & otherwise \end{cases} \quad 2.68$$

The constant $min\_dist$ is a user-defined parameter that controls the spread of points within the lower-dimensional representation.[48]

UMAP performs a user-defined number of algorithm iterations called epochs where the positions of points, $y_i$, are updated in two ways. On each iteration every point is moved twice. The first movement pulls a point towards another low-dimensional point that represents one of its nearest neighbours in high-dimensional space. For instance, when moving point $y_a$ UMAP selects another point, $y_b$, randomly from the corresponding neighbours. This random

selection probability is proportional to the connection strength in the original neighbourhood, with closer points being more likely to be selected. Point $y_a$ is then moved using the following gradient descent algorithm:

$$y_a^{(t)} = y_a^{(t-1)} + \alpha \cdot \nabla(\log(\Phi))(y_a, y_b) \quad 2.69$$

where $\alpha$ is the learning rate, and $\nabla(\log(\Phi))(y_a, y_b)$ is the gradient of the log of function $\Phi$ with respect to points $y_a$ and $y_b$. This gradient is synonymous to find the gradient of the cost function. The second movement pushes a point away from a low-dimensional point that is not representing one of its nearest neighbours. Let $y_c$ be a third point that is not a member of $y_a$'s nearest neighbours that is selected at random. In a similar manor the push is calculated as follows:

$$y_a^{(t)} = y_a^{(t-1)} + \alpha \cdot \nabla(\log(1 - \Phi))(y_a, y_c) \quad 2.70$$

After the two movements, attraction and repulsion, the learning rate, $\alpha$, is reduced to aid the descent algorithms with finding their respective minimums.40

#### *2.2.2.2. Clustering*

This set of algorithms is often used in conjunction with DR to group vectors based on their separation. Clustering is referred to as an unsupervised classification method due to its ability to put data points into classes without any prior class knowledge. The need for such algorithms stems from the uncertainty that datasets present in that it is not always clear how to separate data into individual subgroups.[50]

##### k-Means Clustering

The k-means clustering algorithm is arguably the simplest of clustering algorithms available. This technique functions by placing a user-defined number of centroid vectors within the

training dataspace. The algorithm works iteratively with the following logic. At each time step, the algorithm organises the training dataset into subsets based on whichever centroid point is closest using a squared Euclidean distance. After creating these subsets, the new centroid locations are calculated. For each subset of data points, the new centroid location is calculated as the centre of gravity for all points within the cluster. The algorithm then repeats the creation of subsets and updating of centroid locations until the centroid positions converge (Figure 2.9).[51]

One of the main deciding factors that effect this algorithms performance is how the initial centroid locations are chosen. The most ubiquitous method for doing this is the kmeans++

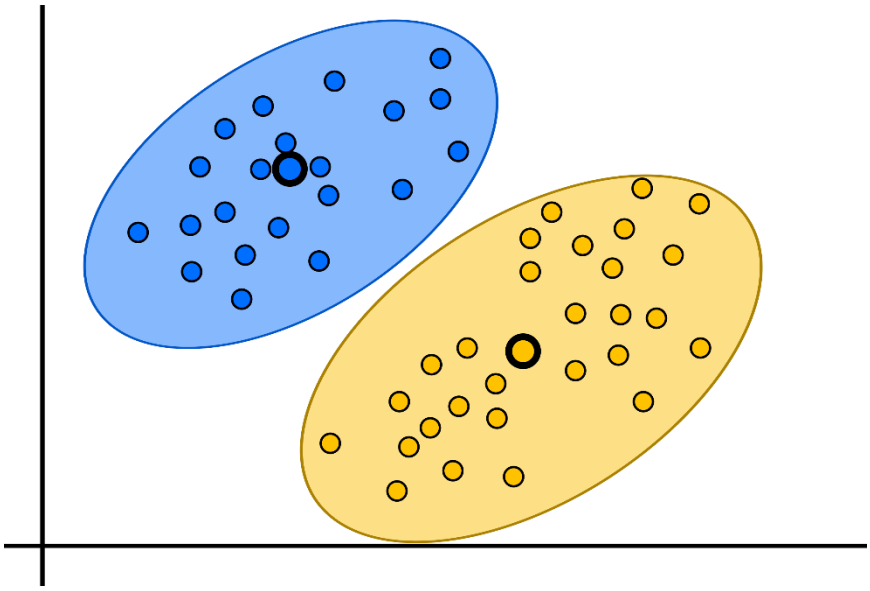


*Figure 2.9*

*Demonstration of the result of applying k-means clustering with two cluster centroids on an example dataset. The centroids ultimately converge to local centres of mass. The above example also presents a case where the incorrect $k$-value was selected as the yellow cluster could be further divides into two separate populations.*

algorithm. Here the centroid locations adopt positions equal to existing points within the training dataset. This is done randomly with the rule that the initial centroid locations should be spread evenly throughout the data space. To do this each centroid is placed one after another with the first centroid being placed using a uniform distribution. The subsequent centroids are selected by applying weights to the remaining datapoints such that points close to existing centroids have a lower probability of being chosen for the next centroid. This repeats until all $k$ centroids have been placed.[52]

The second performance deciding factor is the value of $k$, the number of centroids chosen. The most common technique used to choose the number of clusters is known as the elbow method. Here the clustering algorithm is repeated for different $k$ values so that each can be evaluated. The evaluation takes the sum of the squared Euclidean distances from each point to their closest centroid. These squared distance sums are then plotted against the chosen $k$-value. The resultant curve will possess a decreasing sum as the $k$-value increases. However, the optimum value for $k$ will be the value at which the curve kinks like an elbow.[53]

## 2.3. Deep Learning and Neural Networks

DL is a subset of ML that governs the use of artificial neural networks. So named after the architecture of DL agents, where data is transformed into various layers of deepening abstraction. The more abstraction the deeper the neural network.[54,55] As a subset of ML, DL is able to carry out all the supervised and unsupervised learning tasks that ML can and can even supersede ML in performance. Most notably, DL insights have led to ground-breaking classification performance in the field of image recognition.[56,57] In current times, DL research has resulted in generative models, which aim to create new data from what they have learnt from a training set.[58]

### 2.3.1. Perceptrons

One of the first innovations in the development of neural networks was the discovery of the perceptron. Perceptrons were first invented by Frank Rosenblatt in 1958, as a way of modelling the way the biological brain stores and processes information.[59] Although originally intended to be mechanical, today perceptrons are ML agents specialised for binary classification tasks.

A perceptron is commonly represented as a computational node with many inputs and a single output (Figure 2.10). Each connected input has an associated weight, which is used to compute a weighted sum of all inputs. This weighted sum is put through an activation function to yield a classification result. The output of a perceptron can be summarised by:

$$o = f(\vec{w} \cdot \vec{x} + b) \tag{2.71}$$

where $o$ is the output prediction of the perceptron, $\vec{w}$ are the weights that correspond to each feature of input $\vec{x}$, and $b$ is the perceptron's bias. The function $f$ is the activation function. In

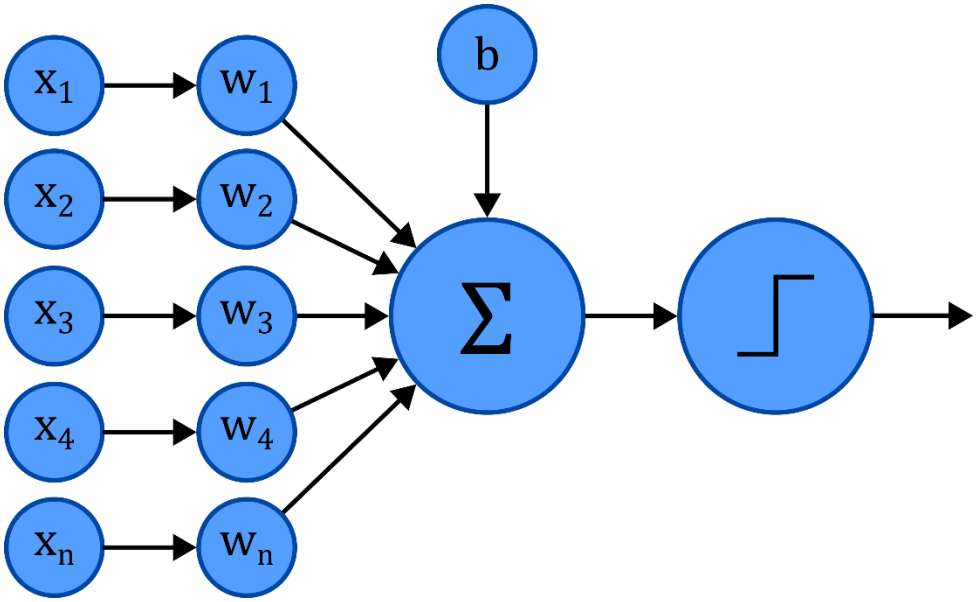



*Figure 2.10*

*Schematical representation of the computational flow within a perceptron. The input vector, $x_i$, is scaled by the weight vector, $w_i$, before being summed along with a bias, $b$. The result is then passed through a step function to yield the output.*

this instance, $f$ will represent the Heaviside step activation function where the output will be set to $1$ if the biased weighted sum is positive, or $0$ otherwise. The original method for learning, was to update the value of the weights and bias based on the mathematical difference between the perceptron's predicted output and the desired training output. At first the weight and bias values are initialised either randomly or by setting them all to equal $0$. Then these values can be updated. The learning algorithm utilised a learning rate to regulate the step size when updating the values. The algorithm can be summarised with the following equations:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} + \alpha(\vec{y}_i - \vec{o}_i) \cdot \vec{x}_i \qquad 2.72$$

and

$$b^{(t+1)} = b^{(t)} + \alpha(\vec{y}_i - \vec{o}_i) \quad 2.73$$

where $\alpha$ is the learning rate, $t$ is the current epoch of the training process, and $\vec{y}$ is the vector of target labels associated with the inputs $\vec{x}$. This training process is repeated a user-determined number of times until the resultant weights and bias have been set such that the difference between predictions and labels is minimised.[59–61]

### 2.3.2. Feed-Forward Networks

The next step in the development of DL after the advent of the perceptron, was to incorporate a network of perceptrons to imitate the nature of the brain. This network of connected perceptrons is commonly illustrated as shown in Figure 2.11. Here perceptrons are organised into layers which each process the input vectors to yield an array of activations. This vector of first layer activations can then be fed into another layer of perceptrons to produce a second layer of activations which can be fed into yet another layer of perceptrons and so on. The final

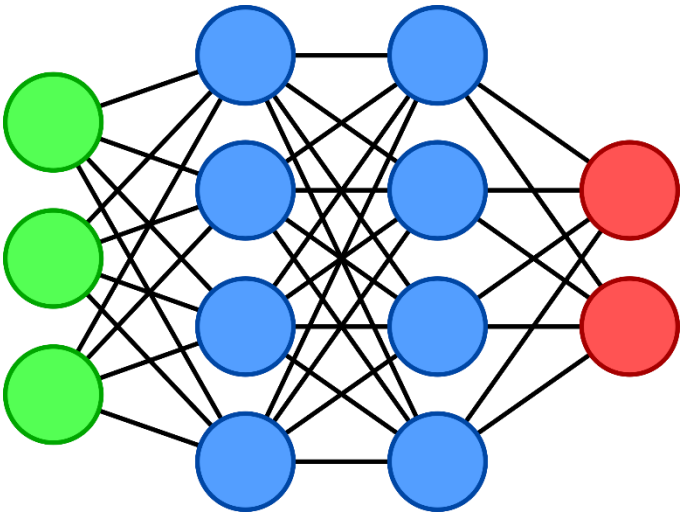


*Figure 2.11*

*Architecture of a typical FFNN. Each element of the input vector, (green), is passed to each neurone in the hidden layers, (blue). The outputs of each layer are passed to each unit of the following layer in a fully connected manner. The final, output vector, (red), is returned by the algorithm.*

*Table 2.1*

*Truth table for the XOR logic gate. The logic of which cannot be replicated by neural networks that utilise only linear activation functions.*

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

layer of this network is known as the output layer which is designed specifically to the task the network is to perform. For instance, in binary classification or 1-dimensional regression the final layer would have a single perceptron to yield a single output value.[62,63]

These networks are known as multi-layer perceptrons (MLPs). However, there are inconsistencies in the field of DL in terms of terminology. Some DL scientists/engineers would treat the MLP as identical to another network type called the feed-forward neural network (FFNN), whilst others specify that MLPs purely contain perceptrons that only utilise step function activation. In this work, the MLP will take on this definition of only using step-functions. Because of this limitation, MLPs cannot update their weights and biases such that non-linear decision boundaries manifest. This means that classification problems with populations that are not linearly separable are not possible to solve using MLPs. The famous example of this is the notorious XOR problem, where the logic in Table 2.1 cannot be learned using linear techniques.[64]

With regards to the aforementioned FFNN, this XOR problem has been solved. Owing to the wider arsenal of activation functions, FFNNs can learn non-linear decision boundaries. FFNNs are illustrated in the same manner as MLPs with multiple layers containing multiple, in this case, neurones. Each layer of neurones can use various activation functions which each come

with their own advantages and limitations.[65] The current most popular activation function is the Rectified Linear Unit (ReLU) function which simply makes any negative activation 0.

Overall, the majority of the applications of FFNN are within the supervised learning tasks of classification and regression. For the various applications of neural networks for this purpose, the interested reader is referred to Section 2.3.6.

### 2.3.3. Gradient Descent and Learning

When it comes to the learning portion of DL, and even ML, it is very common to use what is known as a gradient descent algorithm to minimise loss within a model.[66] Naturally, in order to learn, the first step is to outline a method to measure this error. The metric or function that is used to measure this error is known as the loss or cost function.[67] In DL, and even ML, there are a plethora of different functions of measuring loss some of which are summarised in Table 2.2. On each training epoch, the input data is shuffled before calculating the loss associated with each input vector. The loss on each input is then used to update the values of the weights

*Table 2.2*

*Summary of common loss function used in DL, and even ML, along with their definitions and use cases.*

| Loss Function | Equation | Uses |
|---|---|---|
| **Cross Entropy** | $-\sum y_c \log(p_c)$ | Classification |
| **KL Divergence** | $\sum y_c \log\left(\frac{y_c}{p_c}\right)$ | Classification |
| **Mean Squared Error** | $\frac{1}{n}\sum (y_i - p_i)^2$ | Regression |
| **Mean Absolute Error** | $\frac{1}{n}\sum \lvert y_i - p_i \rvert$ | Regression |

and biases such that this value is reduced. This method is known as stochastic gradient descent (SGD) and is applied with the following equation:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \frac{\partial C}{\partial \theta^{(t)}} \qquad 2.74$$

where $\theta$ is a parameter of the cost function, $C$, to be optimised, and α is the learning rate.[68] The variable $\theta$ can be substituted for any of the weights or biases present in the neural network to be optimised.

However, in DL, there are many different algorithms other than SGD that can be applied to optimisation of weights and biases. The reason for this varied arsenal of what is known as optimisers, is that every problem will have its own unique loss landscape. Some loss landscapes will simply be parabolic in appearance, where there are no local minima and only a single global minimum. On the other hand, some loss landscapes will have a sprawling surface with many local minima.[69] As a result, different optimisers have been developed to tackle the local minimum problem. For example, one of the most popular optimisers is the Adaptive Moment Estimation (Adam) optimiser. Adam uses the concept of mathematical moments to generate an SGD method which incorporates an adaptive learning rate. Each iteration of Adam is denoted as:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t+1)} \cdot \frac{m^{(t+1)}}{\sqrt{v^{(t+1)}} + \varepsilon} \qquad 2.75$$

where $m$ and $v$ are the first and second moments of the loss surface with respect to variable $\theta$, $\alpha$ is the adaptive learning rate, and $\varepsilon$ is a small mathematical constant to prevent a divide by zero error. These moments and the learning rate are defined as follows:

$$m^{(t+1)} = \beta_1 \cdot m^{(t)} + (1 - \beta_1) \cdot \frac{\partial C}{\partial \theta^{(t)}} \quad 2.76$$

$$v^{(t+1)} = \beta_2 \cdot v^{(t)} + (1 - \beta_2) \cdot \frac{\partial C}{\partial \theta^{(t)}}^2 \quad 2.77$$

$$\alpha^{(t+1)} = \alpha^{(t)} \cdot \frac{\sqrt{1 - \beta_2^t}}{(1 - \beta_1^2)} \quad 2.78$$

where the hyperparameters $\beta_1$ and $\beta_2$ are the decay rates that control how fast the moments of the function decrease.[70] Overall, using an optimiser like Adam with an adaptive learning rate allows the weight and bias values to implement large changes at first to avoid local minima but then also implement small changes when the values need to settle into the global loss minimum.

### 2.3.4. Backpropagation

As shown in the previous section regarding gradient descent, for learning in a neural network to occur the gradients of the loss function with respect to each weight and bias needs to be calculated. However, in networks with multiple layers of abstraction this is challenging. To solve this, the backpropagation algorithm was developed. This algorithm allows the loss of a training iteration to be backpropagated to the weights and biases of any layer via application of the differential chain rule.[71]

Let $C$ represent the cost function, $W_{ij}^l$ the weight of the connection between $i$th neurone in layer $l$ and the $j$th neurone in layer $l-1$, and $b_i^l$ the bias of the $i$th neurone in layer $l$. To implement gradient descent the partial derivatives $\frac{\partial C}{\partial W_{ij}^l}$ and $\frac{\partial C}{\partial b_i^l}$ need to be calculated for every combination of $i$, $j$, and $l$. As the name suggests, this is achieved by working backwords

from the output activations. Let the activation and weighted sum of a neurone in the $l$th layer be represented as $\vec{a}_i^l$ and $\vec{z}_i^l$, respectively, such that

$$\vec{a}_i^l = f(\vec{z}_i^l) \tag{2.79}$$

and

$$\vec{z}_i^l = \sum_j W_{ij}^l \, \vec{a}_i^{l-1} + \, b_i^l \tag{2.80}$$

where $f$ is the activation function of the neurone. Initially, Equations *2.79* and *2.80* are used to calculate the gradient of the loss with respect to the weighted sum of the individual neurones in the output layer. As the loss function and activation function will be easily differentiable, the chain rule can be applied such that:

$$\frac{\partial C}{\partial \vec{z}_i^L} = \frac{\partial C}{\partial \vec{a}_i^L} \frac{\partial \vec{a}_i^L}{\partial \vec{z}_i^L} \tag{2.81}$$

where $L$ represents the number of layers in the neural network. From here it is then possible to calculate the gradient of C with respect to the weights and biases in the final layer, $L$, by applying the chain rule further.

However, the above logic only covers the final layer. In the intermediate or hidden layers, the differentials are more complicated as a change in the weights of hidden layers will have an effect on every activation in the layer that follows it. To overcome this, matrix manipulations and the fact that all neurones use the summation operator are used to come up with the following equation:

$$\frac{\partial C}{\partial \vec{z}^l} = \left( (W^{l+1})^T \cdot \frac{\partial C}{\partial \vec{z}^{l+1}} \right) \otimes \frac{\partial \vec{a}^l}{\partial \vec{z}^l} \tag{2.82}$$

where $T$ is the transpose operator.

Once the $\frac{\partial C}{\partial \vec{z}^l}$ vector has been calculated for every layer in the neural network, these vectors can be used to find the weight and bias derivatives as follows:

$$\frac{\partial C}{\partial W_{ij}^l} = \frac{\partial C}{\partial \vec{z}_i^l}\frac{\partial \vec{z}_i^l}{\partial W_{ij}^l} = \frac{\partial C}{\partial \vec{z}_i^l}\,\vec{a}_j^{l-1} \tag{2.83}$$

$$\frac{\partial C}{\partial \vec{b}_i^l} = \frac{\partial C}{\partial \vec{z}_i^l}\frac{\partial \vec{z}_i^l}{\partial \vec{b}_i^l} = \frac{\partial C}{\partial \vec{z}_i^l} \tag{2.84}$$

With all the gradients being backpropagated, they can be used in gradient descent learning.[71] For example, in Equations *2.74* and *2.75* from the previous Section 2.3.3, the $\theta$ is substituted for every $W_{ij}^l$ and $\vec{b}_i^l$ from the network.

### 2.3.5. Autoencoders

AEs are special neural networks that adopt a specific architecture such that the network can function as a DR agent. The most common use for a neural network, as mentioned previously, is within classification or regression tasks. However, with this architecture, the neural network learns to reconstruct input data after transforming the vector through various layers of abstraction. One of these abstract layers, typically the central hidden layer, has its activation vector extracted as the low-dimensional representation of the input vector. As such the architecture adopts an hourglass-like appearance as shown in Figure 2.12.[72]

To measure the accuracy of the low-dimensional representation, AEs rely on the reconstruction error. This error is typically calculated using either the mean squared error (MSE) or mean absolute error (MAE) losses.[73,74] A high reconstruction error would imply that the central activation vector, or latent space, compresses the input vectors so much so that too little information remains. With too little information preserved in the latent space the

decoding side of the AE will be unable to accurately reconstruct the input vector. This would infer that the latent space is therefore not a good representation of the original input space.

As this technique is used for the purpose of DR, they are commonly used to convert high-dimensional data into low-dimensional representations. In 2006, Hinton et al. demonstrated how AEs can be used to convert high-dimensional image data into low-dimensional vectors.[75] This was done by utilising several popular benchmarking datasets. These include the MNIST hand-written digits and Olivetti face datasets. Here the embeddings of an autoencoder were compared to the PCA technique where it was found that the AE could embed images into a space with better cluster separation between classes.

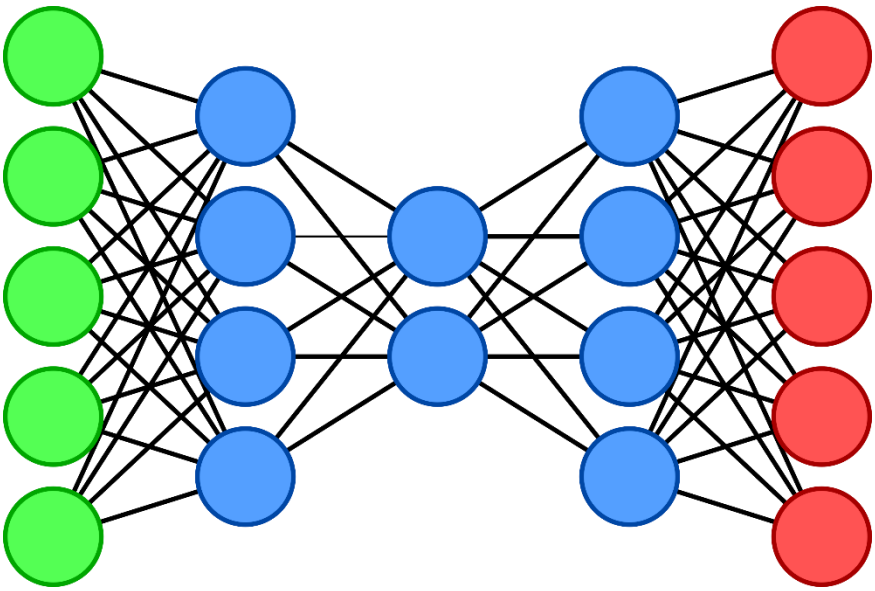


*Figure 2.12*

*Architecture of an example AE. The flow of values is identical to a FFNN with a fully connected system of computational neurones. The central hidden layer functions as a secondary output to allow for non-linear DR.*

### 2.3.6. Convolutional Networks

Convolutional Neural Networks (CNNs) were the next step in the development of the field of DL after the FFNN. This development also takes inspiration from biological processes. Here the aim was to develop a computational model that could replicate the biological concept of a receptive field.[76] After years of development and several different attempts of defining functional models, researchers eventually settled on CNNs to replicate this behaviour. The architecture of a CNN can be thought of as a conventional FFNN with an additional block of layers between the input layer and the first hidden units. This additional block contains two different types of layers called the convolution and pooling layers. These layers work together to extract patterns and exaggerate features within the input data such that classification is enhanced.[77]

Today, the convolutional layers operate by iteratively sliding a kernel over the input matrix. At each kernel position, the following calculation is carried out:

$$o = \sum f(W \otimes X + B) \qquad 2.85$$

where $o$ is an element of the output matrix, $f(X)$ is the activation function, $W$ is the convolutional kernel, $X$ is the input submatrix underneath the kernel's position, and $B$ is the kernel's bias matrix. The size of the output matrix depends on the size of the kernel used, the step size, and the number of kernels used each layer. Overall, the output matrix will be smaller than the input matrix on the original axes but will have a larger additional axis corresponding to the number of kernels used (Figure 2.13a). Each layer can utilise multiple kernels that will each produce a convolved output which are subsequently stacked into the overall output matrix.

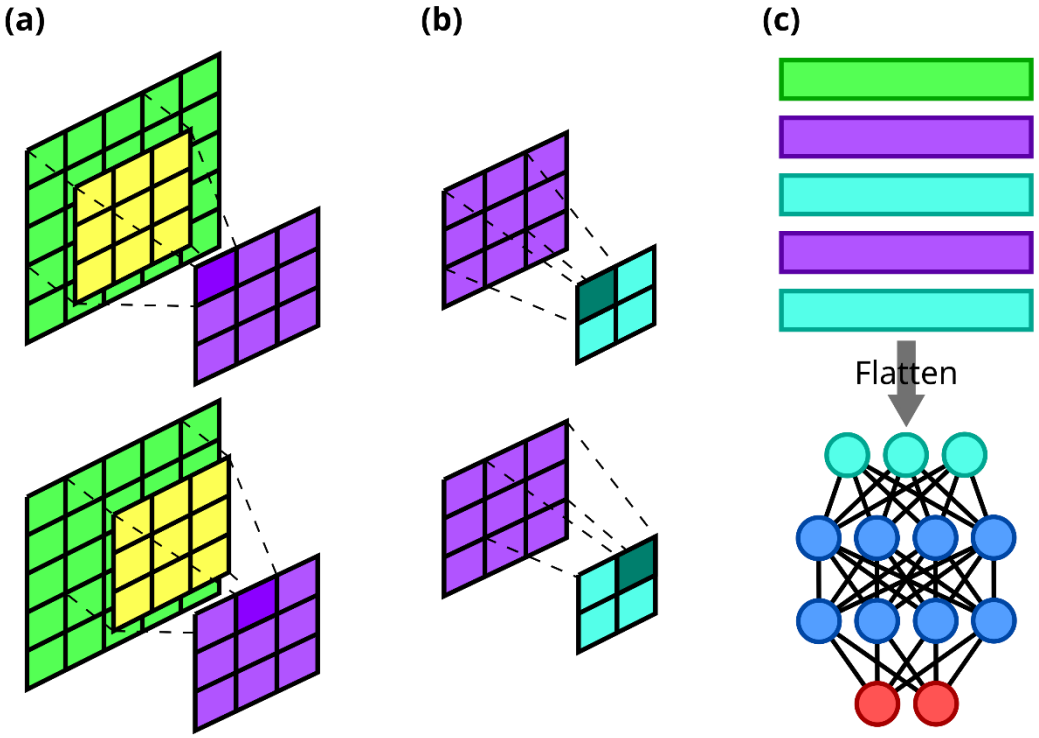



*Figure 2.13*

*Demonstrations of the convolution, (a), and pooling, (b), algorithms. The overall architecture of an example CNN is also shown, (c), where two blocks of convolution, (purple), and pooling, (cyan), feed into a fully connected FFNN.*

Convolutional layers are typically followed by pooling layers which compress the generated feature maps. The output of convolutional layers is typically much larger than the input layer. If these feature maps were to be flattened and inputted into the fully connected dense layers of a neural network, the number weights and biases for this input would be very large. As a result of this, training will be very slow, and a large amount of computational memory will be required. The pooling layers fix this issue by compressing each feature map into a smaller matrix size whilst preserving the overall feature shapes. This is achieved using a similar method to the convolutional layers where a window of user-defined size is iterated across the feature maps generating a single value per position. In this case, however, the pool positions

typically never overlap. Also, at each position the corresponding value is calculated in a different way depending on the type of pooling used. The most common pooling layer used is the max pooling layer where the max value at each pool location is calculated and returned to the next layer (Figure 2.13b).

Overall, the output of the block of convolutional and pooling layers is a matrix consisting of compressed feature maps that exaggerate latent features of the input data.[77] The output of the convolutional block is then flattened before being inputted into a FFNN block (Figure 2.13c).

The applications of CNNs are similar to those of the FFNNs in that they are primarily seen being applied to regression or classification tasks. However, due to the improved performance that coincides with the use of convolutional layers, the majority of neural network classification applications predominantly involves CNNs.[78–80] For CNNs, their main application is to the analysis of image data. For instance, in 2010 Krizhevsky et al. developed a CNN architecture for the classification of 1.2 million images into 1000 classes known as the AlexNet.[81] This network was able to accurately classify images into their respective classes with an error rate of 37.5%. Whilst this error rate seems high, this value was superior to the previous best network which produced an error rate of 45.7%.

## 2.4. References


1 J. Binney and D. Skinner, *The Physics of Quantum Mechanics*, 2008.

2 P. A. M. Dirac, *The Principles of Quantum Mechanics*, 3rd edn., 1947.

3 M. Born, *Zeitschrift für Physik*, 1926, **38**, 803–827.

4 D. Bohm, *Quantum Theory*, New York, 1951.

5 C. J. Chen, *Introduction to Scanning Tunneling Microscopy*, Oxford University Press, Oxford, 2nd edn., 2007.

6 M. Razavy, *Quantum Theory of Tunneling*, 2nd edn., 2014.

7 R. H. Fowler and L. W. Nordheim, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 1928, **119**, 173–181.

8 M. L. Perrin, E. Burzurí and H. S. J. Van Der Zant, *Chem Soc Rev*, 2015, **44**, 902–919.

9 T. Albrecht, *Nat Commun*, 2012, **3**, 829.

10 K. Y. Camsari, S. Chowdhury and S. Datta, In Springer Handbook of Semiconductor Devices, 2023, pp. 1583–1599.

11 T. Meier, P. Thomas and S. W. Koch, In *Coherent Semiconductor Optics*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 25–33.

12 M. Buttiker, *IBM J Res Dev*, 1988, **32**, 63–75.

13 R. C. Jaklevic and J. Lambe, *Phys Rev Lett*, 1966, **17**, 1139–1140.

14 M. Büttiker, *Phys Rev B*, 1986, **33**, 3020–3026.

15 R. Landauer, *Zeitschrift für Physik B Condensed Matter*, 1987, **68**, 217–228.

16 Z. Q. Yu, C. M. Wang, Y. Du, S. Thevuthasan and I. Lyubinetsky, *Ultramicroscopy*, 2008, **108**, 873–877.

17 G. Gautschi, in *Piezoelectric Sensorics*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 5–11.

18 J. Wiedemeier, G. Spencer, M. J. Hagmann and M. S. Mousa, *Microscopy and Microanalysis*, 2019, **25**, 554–560.

19 J. Halbritter, G. Repphun, S. Vinzelberg, G. Staikov and W. J. Lorenz, *Electrochim Acta*, 1995, **40**, 1385–1394.

20 A. L. Samuel, *IBM J Res Dev*, 2000, **44**, 207–219.

21 J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane, *Artificial Intelligence in Design '96*, 1996, 151–170.

22 P. Cunningham, M. Cord and S. J. Delany, in *Machine Learning Techniques for Multimedia*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 21–49.

23 M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro and M. Febrero-Bande, *Neural Networks*, 2019, **111**, 11–34.

24 R. Trinchero and F. Canavero, *IEEE Electromagn Compat Mag*, 2021, **10**, 71–79.

25 R. Choudhary and H. K. Gianey, in *2017 International Conference on Machine Learning and Data Science (MLDS)*, IEEE, 2017, pp. 37–43.

26 J. V. Tu, *J Clin Epidemiol*, 1996, **49**, 1225–1231.

27 A. Singh, N. Thakur and A. Sharma, in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 1310–1315.

28 T. Cover and P. Hart, *IEEE Trans Inf Theory*, 1967, **13**, 21–27.

29 I. Wickramasinghe and H. Kalutarage, *Soft comput*, 2021, **25**, 2277–2293.

30 L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, 1984.

31 S. Murthy and S. Salzberg, in *KDD'95: Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, pp. 222–227.

32 B. Neal, 2019, arXiv:1912.08286.

33 Tin Kam Ho, in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, IEEE Comput. Soc. Press, 1995, vol. 1, pp. 278–282.

34 L. Breiman, *Mach Learn*, 1996, **24**, 123–140.

35 S. Suthaharan, Springer, Boston, MA, 2016, pp. 207–235.

36 O. L. Mangasarian and D. R. Musicant, *Journal of Machine Learning Research*, 2001, **1**, 161–177.

37 T. Hofmann, B. Schölkopf and A. J. Smola, *The Annals of Statistics*, 2008, **36**, 1171–1220.

38 T. Hastie, R. Tibshirani and J. Friedman, Springer, New York, NY, 2009, pp. 485–585.

39 M. Koppen, in *Curse of Dimensionality*, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America , 2000.

40 M. Verleysen and D. François, in *Lecture Notes in Computer Science*, Springer Verlag, 2005, vol. 3512, pp. 758–770.

41 C. O. S. Sorzano, J. Vargas and A. P. Montano, 2014, arXiv:1403.2877.

42 K. L. Sainani, *PM&R*, 2014, **6**, 275–278.

43 J. Shlens, 2014, arXiv:1404.1100.

44 F. Kherif and A. Latypova, in *Machine Learning*, Elsevier, 2020, pp. 209–225.

45 C. C. Aggarwal, A. Hinneburg and D. A. Keim, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2001, vol. 1973, pp. 420–434.

46 G. Hinton and S. Roweis, in *Advances in Neural Information Processing Systems 15*, 2002.

47 L. van der Maaten and G. Hinton, *Journal of Machine Learning Research*, 2008, **9**, 2579–2605.

48 L. McInnes, J. Healy and J. Melville, 2018, arXiv:1802.03426.

49 M. Belkin and P. Niyogi, in *Advances in Neural Information Processing Systems 14*, The MIT Press, 2002.

50 V. Estivill-Castro, *ACM SIGKDD Explorations Newsletter*, 2002, **4**, 65–75.

51 K. P. Sinaga and M.-S. Yang, *IEEE Access*, 2020, **8**, 80716–80727.

52 D. Arthur and S. Vassilvitskii, in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

53 H. Humaira and R. Rasyidah, in *Proceedings of the Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA) 2018, 24-25 January 2018, Padang, Indonesia*, EAI, 2020.

54 Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436–444.

55 H. Schulz and S. Behnke, *KI - Kunstliche Intelligenz*, 2012, **26**, 357–363.

56 G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, vol. 2017-January, pp. 2261–2269.

57 J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, *IEEE Trans Pattern Anal Mach Intell*, 2020, **42**, 2011–2023.

58 R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2022, vol. 2022-June, pp. 10674–10685.

59 F. Rosenblatt, *Psychol Rev*, 1958, **65**, 386–408.

60 S. Chakraverty, D. M. Sahoo and N. R. Mahato, in *Concepts of Soft Computing*, Springer Singapore, Singapore, 2019, pp. 183–188.

61 X. Wang and M. Benning, 2020, arXiv:2012.03642.

62 M. W. Gardner and S. R. Dorling, *Atmos Environ*, 1998, **32**, 2627–2636.

63 J.-G. Attali and G. Pagès, *Neural Networks*, 1997, **10**, 1069–1081.

64 Zhao Yanling, Deng Bimin and Wang Zhanrong, in *The 2nd International Workshop on Autonomous Decentralized System, 2002.*, IEEE Comput. Soc, 2002, pp. 168–173.

65 C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, 2018, arXiv:1811.03378.

66 S. Ruder, 2016, arXiv:1609.04747.

67 Q. Wang, Y. Ma, K. Zhao and Y. Tian, *Annals of Data Science*, 2022, **9**, 187–212.

68 N. Ketkar, in *Deep Learning with Python*, Apress, Berkeley, CA, 2017, pp. 113–132.

69 H. Li, Z. Xu, G. Taylor, C. Studer and T. Goldstein, *Adv Neural Inf Process Syst, 2018*.

70 D. P. Kingma and J. Ba, *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2014*.

71 M. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

72 M. A. Kramer, *AIChE Journal*, 1991, **37**, 233–243.

73 W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias and A. Mechelli, in *Machine Learning*, Elsevier, 2020, pp. 193–208.

74 A. Borghesi, A. Bartolini, M. Lombardi, M. Milano and L. Benini, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, **33**, 9428–9433.

75 G. E. Hinton and R. R. Salakhutdinov, *Science*, 2006, **313**, 504–507.

76 D. H. Hubel and T. N. Wiesel, *J Physiol*, 1959, **148**, 574–591.

77 A. Saxena, *Int J Res Appl Sci Eng Technol*, 2022, **10**, 943–947.

78 Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, *Proceedings of the IEEE*, 1998, **86**, 2278–2324.

79 S. Ren, K. He, R. Girshick and J. Sun, *IEEE Trans Pattern Anal Mach Intell*, 2017, **39**, 1137–1149.

80 K. He, X. Zhang, S. Ren and J. Sun, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 770–778.

81 A. Krizhevsky, I. Sutskever and G. E. Hinton, *Commun ACM*, 2017, **60**, 84–90.

# Chapter 3

## Classification of Tunnelling Current-Time Data in Single-Molecule Science

Contents

## 3.1. Introduction

As mentioned in Chapter 1, the development of a conductance-based nucleotide classifier has stalled due to the inherent overlap of electrical signatures.[1] To reexplore and assess the potential of this method, this chapter investigates the ease at which nucleotide I(t) events can be detected using a commercially available ECSTM and demonstrates how ML techniques can produce accurate classification results despite overlapping distributions.

## 3.2. Results and Discussion

### 3.2.1. Parameter Optimization

Before the potential of a commercial ECSTM for the purpose of I(t) event detection is investigated, the factors effecting measured noise are characterised. This characterisation serves to ascertain the validity of measuring nucleotide events based on a predicted SNR. For this, literature values were collected and compared to reveal an expected conductance for a molecular signal. The conductance of a nucleotide varies from $10^{-2}$ to $10^{0}$ $nS$ which hints that the magnitude of noise oscillations within STM I(t) traces should have conductance magnitudes less than the assumed molecular conductance to allow for nucleotide detection.[1–3]

In general, noise in an STM can be characterised by three separate contributing classes. These are the noise contributions due to electronic noise within the STM preamplifier, noise generated within tunnelling junctions, and noise resulting from fluctuations in the tunnelling gap size.[4] For a large proportion of noise contributors there is little that can be done by the end user of a commercial STM to reduce noise effects. For instance, the electronic noise generated by commercially available preamplifiers and analogue-to-digital converters are

inherent to the electrical engineering of the system, and not designed to be modified by users. On the other hand, there is also a large number of noise contributors that can be managed. These include noise due to vibrations and electromagnetic radiation, which can be remedied by vibrationally isolating the tunnelling junction from the ground and utilising appropriate shielding around electronics, respectively.[5,6] Most notably, the operating parameters of an STM directly contribute to the tunnelling junction gap size.[7] As such, these parameters have a direct influence on the magnitude of measured noise. Therefore, the setpoint and applied bias operating parameters were subsequently optimised.

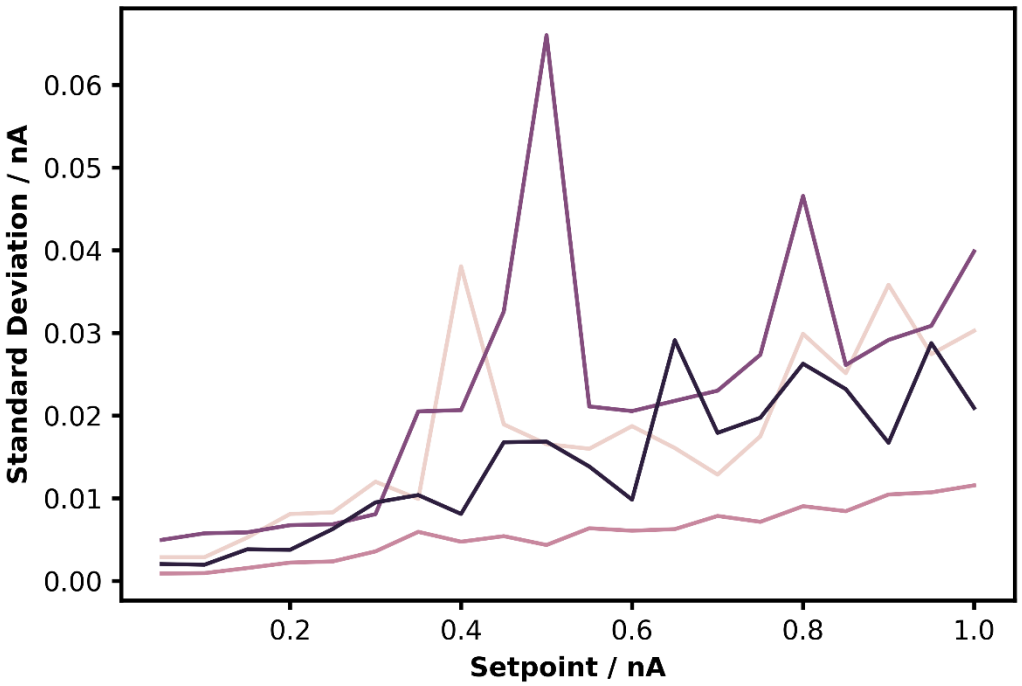



*Figure 3.1*

*Plot showing the relation between the STMs setpoint and the measured noise amplitude. Here the four separate measurements are shown. Overall, the trend shows an increase in noise as the setpoint is increased.*

With the STM that is utilised in this work, the setpoint is defined as the target tunnelling current that the STM should achieve by manipulating the tip-substrate distance. To characterise this parameter, I(t) traces were sampled at various setpoints in an aqueous medium. For the I(t) technique to be used on nucleotides, the ECSTM was characterised in the same aqueous environment as is used during future measurements. Overall, I(t) traces were sampled on 4 separate days with a new etched tip, analyte solution, and prepared sample with each trace lasting 5 seconds. On each occasion, I(t) traces were sampled at the setpoint values of 0.05nA up to 1.00nA in steps of 0.05nA. These traces were sampled with an acquisition frequency of 2083Hz and a constant applied bias of 100mV. The results of this study are shown in Figure 3.1. Overall, the relationship between noise magnitude and setpoint was shown to be positively correlated. Indeed, the higher setpoints yielded noise with a root mean square (RMS) much larger than the lower setpoints. On average, the difference varies from a baseline standard deviation of 2.7 ± 1.7pA for a 0.05nA setpoint up to 25.6 ± 12.2pA RMS for a 1.00nA setpoint. Overall, this shows that yielding larger SNRs would require a lower setpoint to be selected by reducing the noise contributions to the system.

The applied bias parameter is defined as the potential difference applied between the ECSTM tip and the substrate as mentioned in Section 2.1. As this potential difference directly impacts the tunnelling current, it also has a secondary effect on the tunnelling gap size (Equation *2.23*).[8] Therefore, a similar study was carried out where I(t) traces were sampled at various bias voltages within the range of 0.05V to 1.00V in steps of 0.05V. This was also sampled at a frequency of 2083Hz. In this instance, the setpoint was kept constant throughout at a value of 0.05nA. The results of this are shown in Figure 3.2. In general, the different values of applied bias do not appear to significantly impact the magnitude of noise. However, when considering

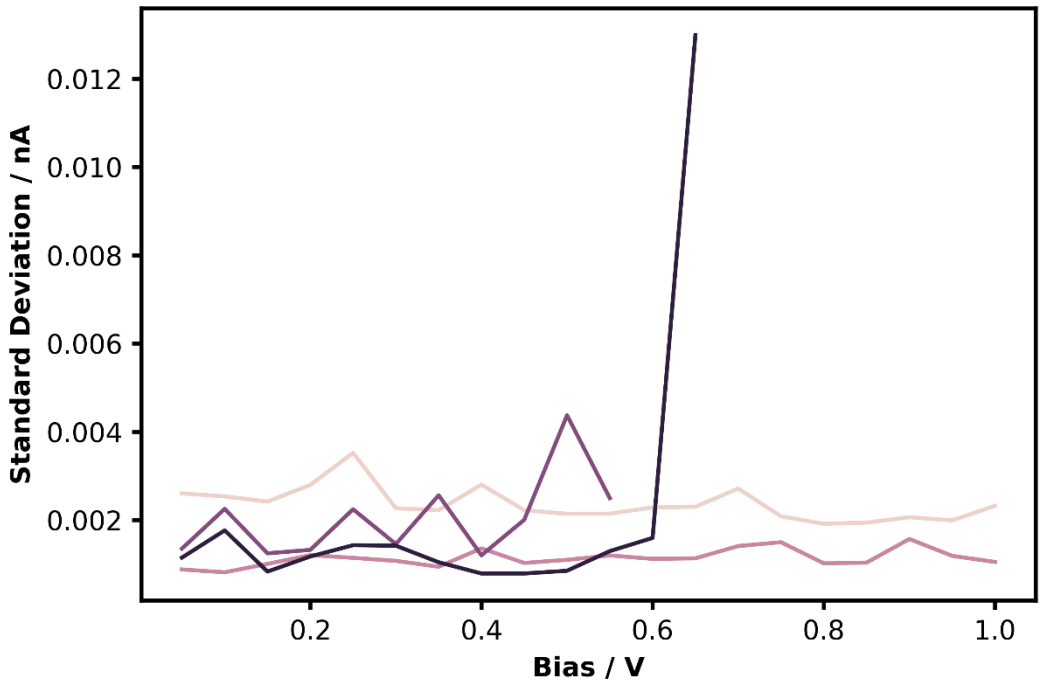



*Figure 3.2*

*Plot showing the relation between the applied bias and the measured noise amplitude. Here four separate measurements are shown. Initially, there appears to be no correlation between applied bias and noise. However, two of the repeats had severe occurrences of high noise that meant the experiment could not continue. This warranted further exploration into the effect of the applied bias on noise using electrochemical techniques.*

two of the four runs, there are no data points available for the higher biases. This is due to a loss in stability during the experiment when applying these higher biases. In these instances, the experiment could not continue as noise caused the STM preamplifier to saturate whereby the tip may have crashed into the substrate or the insulating coating of the tip may have broken down. This behaviour can be interpreted to result from the noise levels being extremely large at those corresponding biases. Given that this large noise in higher biases only occurred in two of the four experiments, it would be appropriate to speculate that additional parameters are interfering with the noise levels that are related to the applied bias. Indeed, a

third study shows that this instability was a result of the tip and substrate potentials themselves.

A subsequent investigation was carried out to examine the noise behaviour related to the applied bias. As stated in Section 2.1.5, there is a continuous range of tip and substrate potentials that correspond to a chosen applied bias. Therefore, electrochemical control was introduced such that the tip and substrate potentials can be investigated. Using CV, both the tip and substrate potentials were characterised simultaneously whilst maintaining the bias potential. These two CVs are shown in Figure 3.3. Analysis of these CVs show that at certain potentials the tip and substrate can undergo redox reactions. More specifically, as mentioned in Section 1.2.1, the gold material of the electrodes will become oxidised at a certain potential

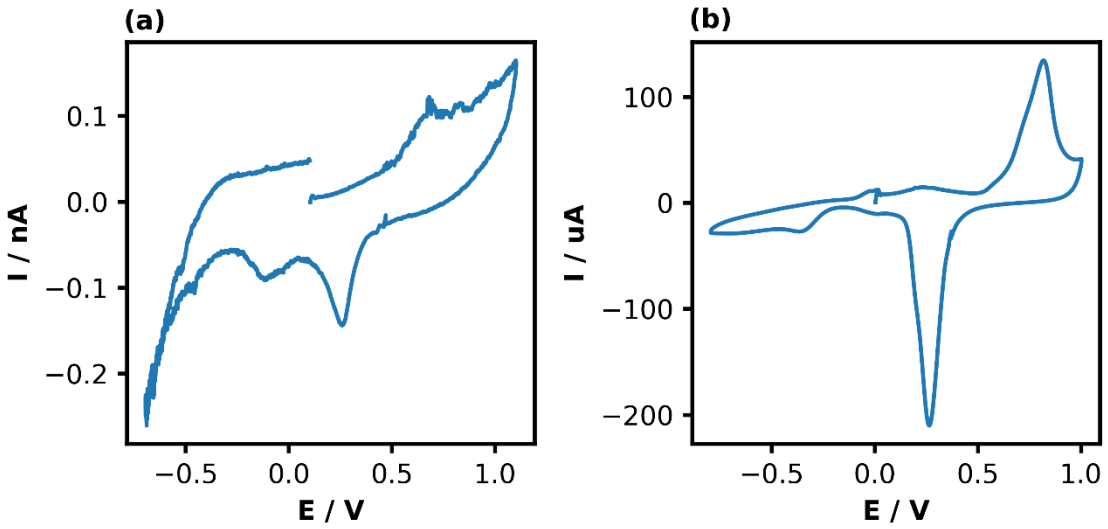



*Figure 3.3*

*CVs of both the tip (a) and substrate (b) electrodes measured in a phosphate buffer medium vs a Pt/Ir quasi-reference electrode. Here it can be seen that at certain potentials faradaic processes occur which generate electrical current and interfere with I(t) measurements. For instance, the oxidation of the tip and substrate occurs at potentials above 500mV.*

which induces an electrochemical current. Likewise, once oxidised, another electrochemical current can be induced by lowering the electrode potentials such that the surface material is reduced back to gold.[9] For the above measurements, the oxidation peaks occur around 0.7 V for both the tip and substrate with a slight difference between the two. These differences likely stem from the different surface structure where the substrate is single crystal, and the tip is likely polycrystalline and much rougher. Additionally, the tip electrode's surface area is far smaller which could lead to slight differences in peak positions.[10] In addition to the redox peaks, if the surface potentials become too high or too low, gases will begin to evolve causing even greater electrochemical currents.[11,12] This can be seen in Figure 3.3a where the voltage fringes show the beginning of large faradaic currents. Lastly, an additional reduction peak can be seen within Figure 3.3a at around -0.1V where dissolved oxygen molecules within the electrolyte become reduced.[13]

All of these features are especially problematic for the tip electrode as the tip current is measured for both CV and spectroscopy techniques. A large electrochemical current at the tip electrode will cause the piezo actuators to move the tip such that the setpoint is re-established. This has a knock-on effect of greatly increasing noise in the tip current. Therefore, to reduce noise contributions due to electrochemical reactions, the tip and substrate potentials should be adjusted such that faradaic processes do not occur whilst simultaneously achieving the desired bias voltage.

### 3.2.2. Distance Calibration

Forming molecular junctions between the STM tip and substrate requires an appropriate tip-substrate separation. This is demonstrated with the I(s) spectroscopy techniques. Previously,

in Section 1.1.1,the schematic of a typical STM-BJ trace was presented. Here it was shown that the molecular signature, known as a plateau, only manifests at a specific withdrawal distance. More specifically, at an electrode separation that is too small, tunnelling current dominates and there is no effect of molecular binding. Additionally, at an electrode separation that is too large, molecules cannot physically bridge the junction thus resulting in no molecular binding events. Therefore, for the purpose of aiding junction formation, it would be appropriate to select values for the setpoint and bias parameters such that the electrode separation matches the dimensions of the molecule of study.[14]

However, ascertaining the distance between the tip and substrate in an STM in constant-current mode is not trivial. The instrument utilises the tunnelling current to infer a physical distance. Once approached, the STM measures voltage applied to the z-piezo to measure the topography of the surface being studied. However, this topography measure is relative to the piezo's range and not a measure of distance to the surface. Consequently, the relationship between tunnelling current and tunnelling distance within the medium of study was explored. This investigation utilised the "soft-contact" I(s) spectroscopy technique along with tunnelling theory to map the tunnelling current to the electrode separation distance.

During an STM experiment the tunnelling setpoint and applied bias are known meaning Equation *2.23* can be used to produce an effective mapping of tunnelling current to inter-electrode separation by experimental determination of the decay constant. To determine this tunnelling decay constant, $\beta$, three separate experiments were carried out where 1000 I(s) traces were sampled. On each occasion, the setpoint was set to 8nA before withdrawal of the STM tip by 4nm at a rate of 4nm/s. A delay of 15s was used between each trace to measure

the decay throughout a long time period. This allowed any variations between the time of measurement to be visualised. The applied bias was set to -100mV whilst constraining the tip and sample potentials within "quiet" regions of their CV profiles. After sampling, the measured exponential decays were plotted on a logarithmic scale. These logarithmic graphs had linear fits applied using least squares regression such that the gradients could be extracted (Figure 3.4). Based on Equation *2.23*, it can be shown that the logarithmic graph should indeed be a first order polynomial with the gradient representing the decay constant, $\beta$:

$$\ln(I_s) = \ln(G_0 V) - \beta s \qquad 3.1$$

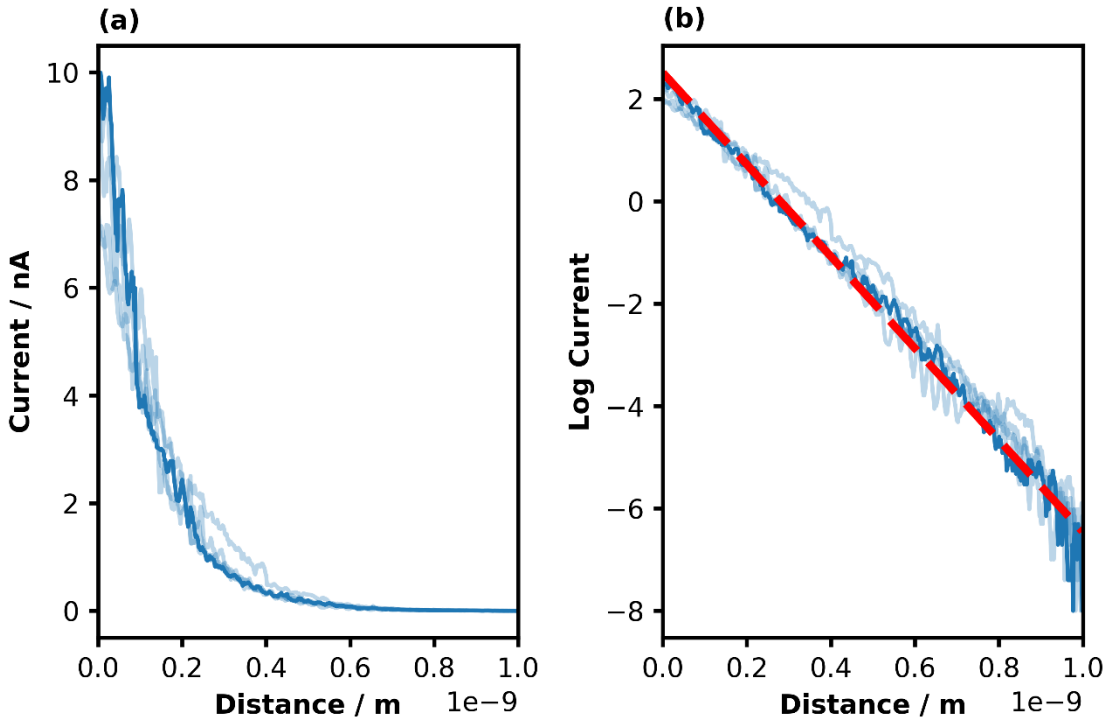



*Figure 3.4*

*Plot of five example decay curves sampled using I(s) spectroscopy. Distance axes were limited to a maximum of 1nm. These are shown on both a linear (a) and logarithmic (b) scale. Additionally, an example linear fit is shown on the logarithmic plot from which decay constants are extracted.*

Applying this procedure to all I(s) traces yielded three sets of 1000 values for $\beta$. Each of these sets were cleaned by removing any traces that failed to reacquire the setpoint or failed to reach the noise floor of tunnelling current. After this filtering a total of 2986 traces remained (Figure A1.1). For more information on I(s) spectroscopy cleaning methods the reader is referred to Chapter 5.

Subsequently, the null hypothesis was tested for the three sampled beta distributions. A one-way ANOVA test was implemented which yielded a p-value of $2.4 \times 10^{-105}$. As this value is lower than the $0.05$ threshold, the null hypothesis is believed to be true such that the three distributions are pooled prior to further analysis. The pooled distribution is shown in Figure 3.5. This distribution was found to be normally distributed with a mean decay constant of 8.16

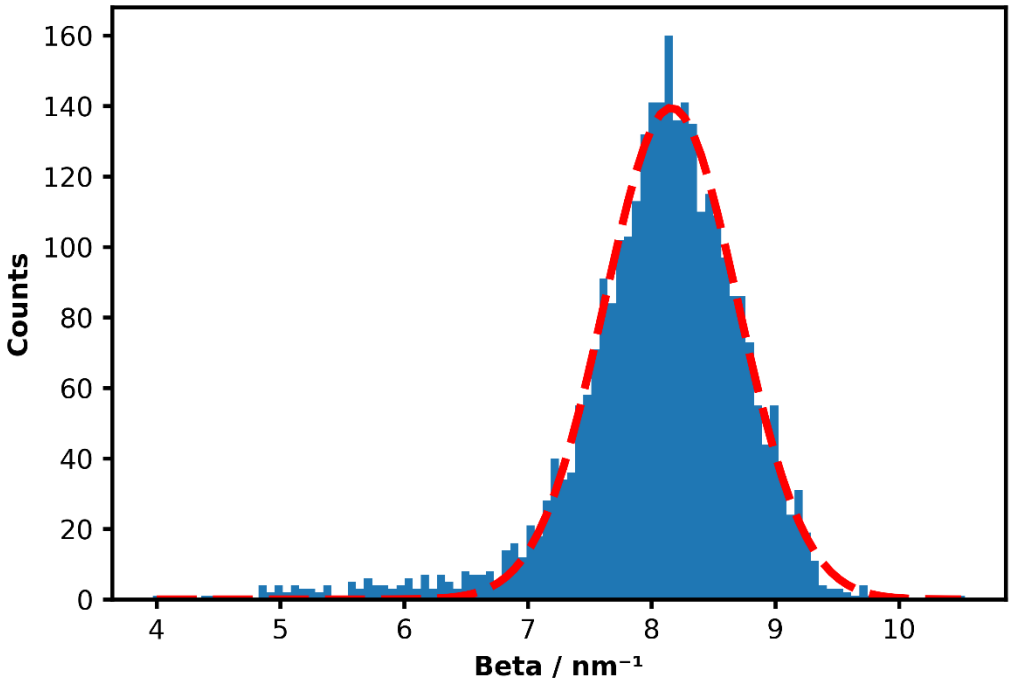



*Figure 3.5*

*Histogram of measured values for the decay constant, $\beta$, pooled from the three experiments. A Gaussian fit is applied as shown (red), to extract an average and standard deviation.*

$nm^{-1}$ and a standard deviation of 0.54 $nm^{-1}$. The goodness of fit was found to be $6.28 \times 10^{-5}$ when applying a Kolmogorov Smirnoff test.

Using this decay constant, any combination of setpoint and applied bias can be mapped to a corresponding electrode distance. For instance, a setpoint of 1nA and an applied bias of 100mV corresponds to an electrode separation of 1.10nm. When considering the distribution of the measured decay constant this gives upper and lower bounds of the electrode separation of 1.18nm and 1.03nm, respectively.

### 3.2.3. Measurement of Ribose-Adenosine Monophosphate

To explore whether these noise and distance optimisation preliminary studies provide any insight into how to improve the STM I(t) technique, an example measurement of rAMP molecules using the I(t) technique was conducted. Previously, it was suggested that the tunnelling junction size should be set to match the size of the molecule under study as this behaviour is seen in I(s) spectroscopy. In Figure 3.6, the molecular structure of rAMP is shown. Here, for the shown snapshot of the molecule, the largest interatomic distance is 12.58Å. From

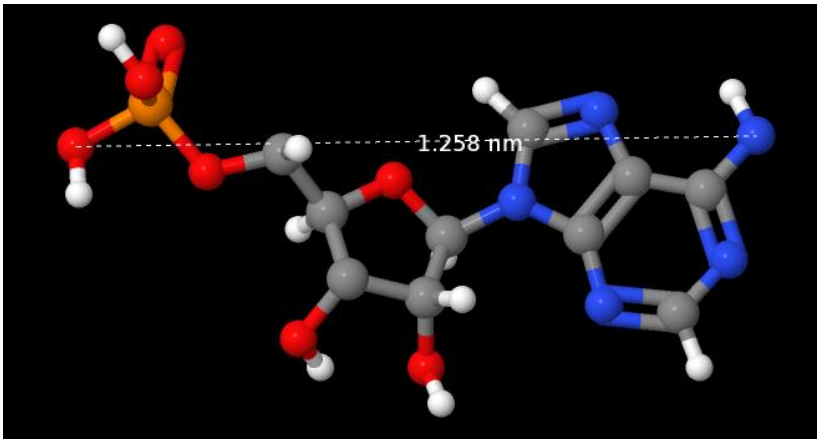



*Figure 3.6*

*A 3-dimensional model of rAMP produced using the software Jmol. The distance between the most separated anchor points is measured and shown.*

*Table 3.1*

*Setpoint values implemented within the I(t) technique alongside their corresponding distances that were calculated via experimental determination of the tunnelling decay constant.*

| Setpoint / nA | Distance / nm |
|---|---|
| 0.01 | $1.66 \pm 0.11$ |
| 0.10 | $1.38 \pm 0.09$ |
| 1.00 | $1.10 \pm 0.07$ |
| 2.00 | $1.01 \pm 0.07$ |
| 3.00 | $0.96 \pm 0.06$ |

this it can be inferred that effective capture of the molecule in this orientation requires the electrode separation to be roughly 12.58Å. Setting the electrode distance to be more than this value should result in no molecular events being captured. However, setting the electrode distance to be shorter may capture the molecule in a different orientation. That being said, there would be a limit on how small the separation can be before the overall tunnelling current dominates and enshrouds the molecular signature.

To test these assumptions, five experiments were performed. For each experiment, I(t) traces were sampled at varying setpoints whilst maintaining the applied bias at -100mV. The chosen setpoints and their corresponding distances are summarised in Table 3.1. For each setpoint, 500 traces were sampled at a rate of 25kHz over one second each. After sampling each trace was then analysed using an amplitude thresholding approach for event detection. Briefly, values that are more than three standard deviations from the experimental average are flagged as anomalous. More information on the details of this event searching algorithm, as well as other algorithms and the challenges of finding events, are outlined in Chapter 4.

Manual inspection of the detected events reveals events with differing characteristics. Figure 3.7a shows a series of sampled events that, directly after each event, feature a brief ‘anti-

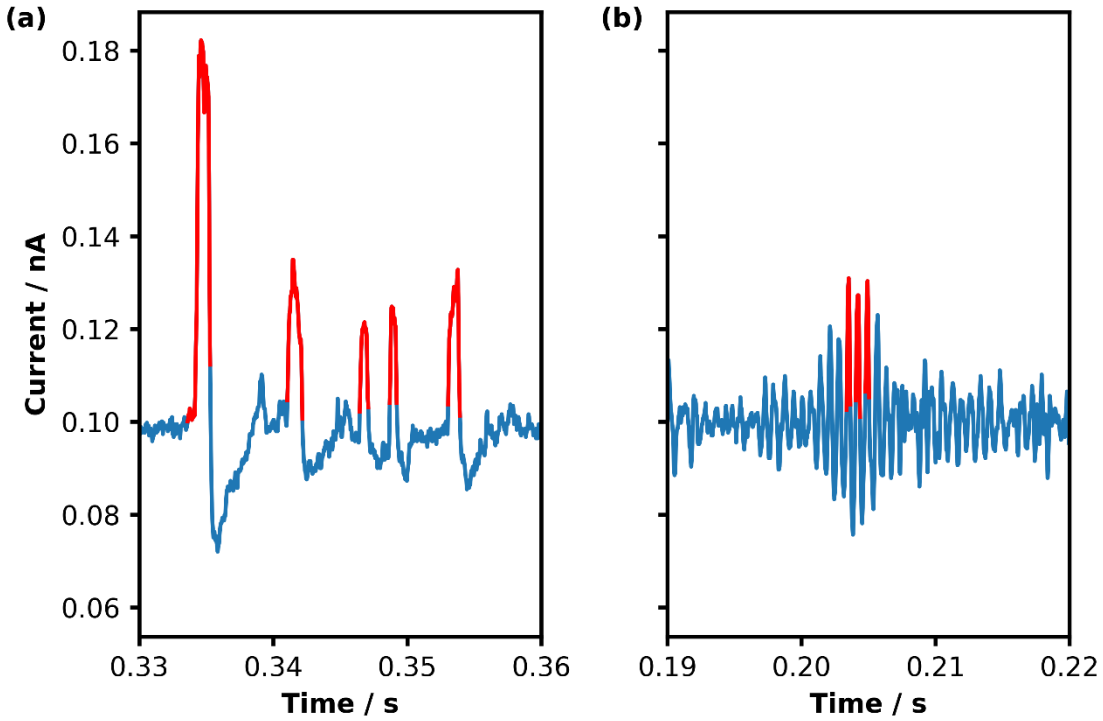



*Figure 3.7*

*Example events (red) detected at a setpoint of 0.1nA and 0.1V applied bias. (a) shows events that are considered to be true positives events, whereas (b) shows events that are considered to be false positives. Shown events are seen to be comparable in height.*

event' where the signal magnitude is briefly reduced. This can be physically explained by the STM being in constant-current running mode. This means that a boost in current resulting from a molecular binding event will result in the STM attempting to re-establish the setpoint. Consequently, the STM tip will retreat from the surface and cause the molecular junction to break. The analyte would then leave the surface. Overall, the measured signal will drop to a level lower than the setpoint, as observed. Subsequently, the STM will reapproach the surface to reacquire the setpoint once again. Alternatively, when looking at events with a similar height, an example of different characteristics can be seen (Figure 3.7b). Here the surrounding noise gradually ramps up before events are detected. As the shape of the event appears

sinusoidal, and the ramping and decaying noise features are present either side, it would suggest these events can be attributed to an increase of noise due to either electrical or vibrational noise.[4,15] It is also possible that these features are artefacts of the operation of the STMs servo as well where a binding event has occurred and caused "ringing" due to PID controller overshoots. These cases highlight the reasons for why it is desirable for the STMs servo to be disabled during the operation of this technique. Unfortunately, this was not possible with the instrument used as vertical drift of the tip along the z-axis was substantial when the servo was disengaged.

From this manual inspection, it was concluded that within the set of detected events there is likely events present that do not manifest from molecular binding. In an attempt to separate the true binding events from the false positives, the traditional approach of viewing distributions of event heights and durations was employed. The results of this analysis are shown in Figure 3.8. Here the two properties of the found events are shown. These are the event duration calculated using the difference in time between the start and the end of an event, and the event height calculated as the setpoint subtracted from the median of datapoints within an event.

A number of observations can be made from the distributions and relationship of these two properties. Firstly, it can be seen that the height and duration of events are dependent on one another. This relationship is best demonstrated in the datasets sampled at 3.0, 2.0, and 1.0nA. Here, there is no existence of events that are both high and long in terms of height and

duration (Figure 3.8c-e). Indeed, any event with a height greater than 0.5nm is very unlikely

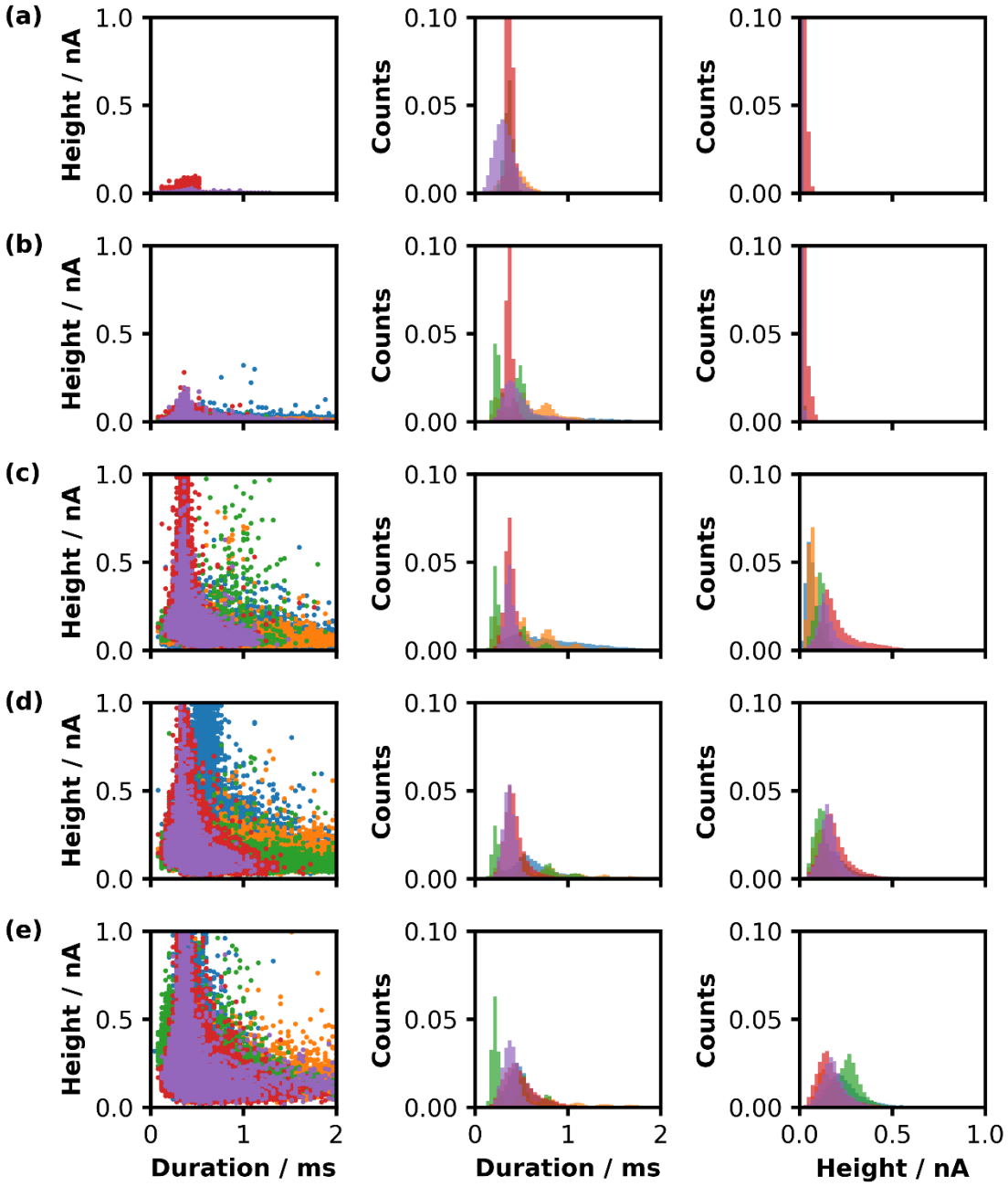



*Figure 3.8*

*I(t) technique results for detecting rAMP at various setpoints. Results are arranged in columns showing event duration vs height scatter plots (left), event duration distributions (middle), and event height distributions (right). Each row is labelled (a-e) which correspond to the setpoints: 0.01, 0.10, 1.00, 2.00, and 3.00nA, respectively. Five statistical repeats are shown for each setpoint, such that the repeats 1-5 correspond to the colours blue, orange, green, red, and purple, respectively.*

to have a duration greater than 1ms. Additionally, any event with a duration greater than 1ms has a small likelihood of having a height greater than 0.5nm. The next observation can be inferred from the event duration distributions. Here it can be seen that for every setpoint, the expected event duration would fall within the range of 0.1ms – 0.5ms. However, it can also be seen that some of the distributions are multi-modal. For instance, the second repeat (orange) sampled at a setpoint of 0.1nA has a second peak manifesting around 0.8ms (Figure 3.8b). This behaviour is also seen in the 1.0nA dataset (Figure 3.8c). The final observation can be stated when considering the event heights at lower setpoints. Within the traces sampled at 0.1 and 0.01nA, it can be seen that no events exist with a height greater than 0.3nA (Figure 3.8a-b). This behaviour agrees with the previous assumption that electrode separation greater than the physical dimensions of the analyte would yield no binding events. However, it is also true that having a lower setpoint will result in less electrical noise being generated by the scanner electronics.[4]

In addition to analysis of event heights and duration, a comparison of the event frequencies with respect to setpoint is made (Figure 3.9). Counter to the above reasoning, there appears to be little effect of setpoint on the number of detected events. When considering the frequencies at 0.01nA, it can be seen that there is a relatively large error. This can be explained by the event detection algorithm. As the algorithm relies on the standard deviation of traces to find events and the traces at lower setpoints have smaller standard deviations, this means that the detection algorithm will be more sensitive to deviations from the trace means. This increased sensitivity will mean that more noise events will be falsely flagged as true binding events and will have measured event frequencies that pick up more subtle noise differences

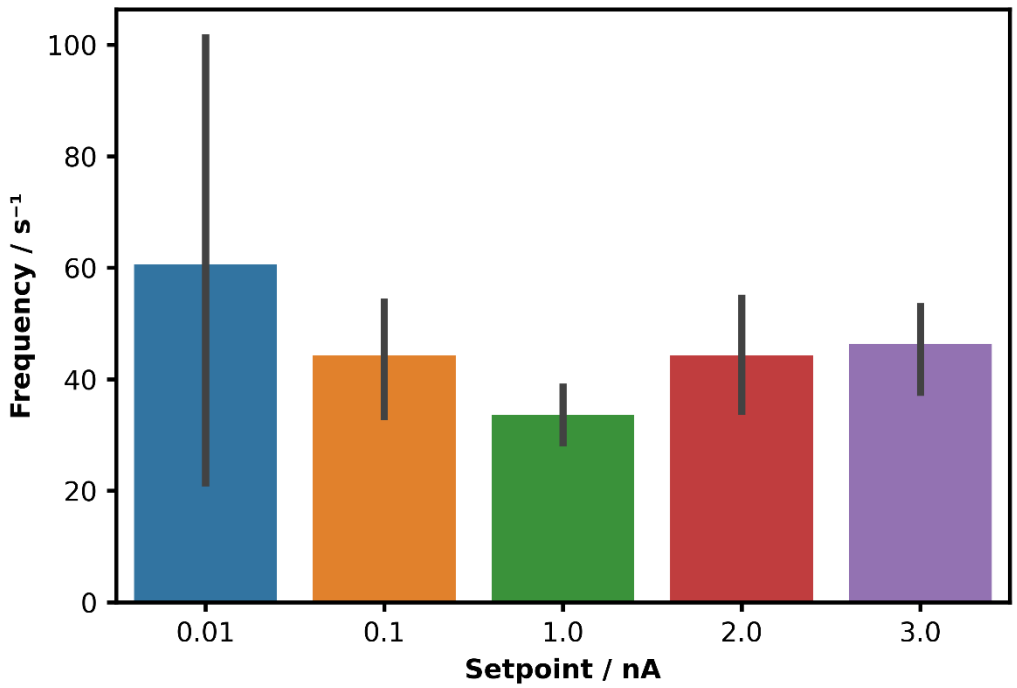



*Figure 3.9*

*Comparison of event frequencies at different setpoints with error bars corresponding to 95% confidence intervals. Frequencies are calculated as the number of events extracted divided by the total time sampled in seconds. It can be seen that setpoint has little effect on the event frequency with no setpoint being statistically different from another.*

between repeats. This shows a limitation with the current event finding algorithm which is further explored in Chapter 4.

In summary, it can be concluded that utilising an ECSTM in this manner, where setpoint and bias are set such that noise contributions are reduced whilst maintaining an appropriate electrode separation distance, is an appropriate method for capturing I(t) events with relative ease. The successful measurement of I(t) events does, however, present the next challenge of both finding I(t) events within traces, and the subsequent separation of true positive events from false positives. In this chapter, the traditional approach of event analysis was

demonstrated whereby the distributions of extracted event properties were compared. This approach, whilst suggesting the presence of multiple populations, did not present any obvious separation between events of different shapes which showed a limitation of the traditional analysis process. As detected events result from a complex process, the I(t) events possess intricate shapes that are described by a large number of variables. The previous statistical approach to analysis of datasets like these is ill-suited as a reduced number of variables is required for analytical ease. For instance, the statistical approach only investigated two variables: the event heights and durations. Because of this, the assumption that different event populations will possess different heights and durations was held, which may mislead analysis. Therefore, new methods for event detection and event separation are required. Instead, ML techniques would be better suited as these techniques are not hindered by large numbers of variables and would avoid any statistical assumptions. As such, alternative approaches to the detection of anomalies are explored in Chapter 4, and alternative approaches to the unsupervised separation of populations within datasets are explored in Chapter 5.

### 3.2.4. Machine Learning Analysis of I(t) Events

Given the previously discussed difficulties surrounding event detection, the extraction of true events from experimental data is not currently possible. However, as the second goal of this chapter is to prove that ML can provide analysis methods to accurately classify datasets with overlapping properties, analysis continued with a simulated dataset. Using a simulated approach allowed events from different classes to have properties that overlap to a specific degree. Additionally, as a simulation can track the true event locations, using a simulation

allows those events to be pulled from I(t) traces with 100% accuracy with no risk of yielding any false events.

The simulated data was constructed by first simulating event-free "empty" traces. These empty traces were developed to mimic behaviours seen in experimental data. Traces were initialised by choosing a setpoint and sampling a normal distribution of points around this value to model noise. For simplicity, the setpoint was simulated to be 0.0nA. Each trace was then initially constructed by sampling 25000 points from a normal distribution with a mean of 0.0nA and a standard deviation of 0.005nA (Figure 3.10a). This simulation is, however, a very

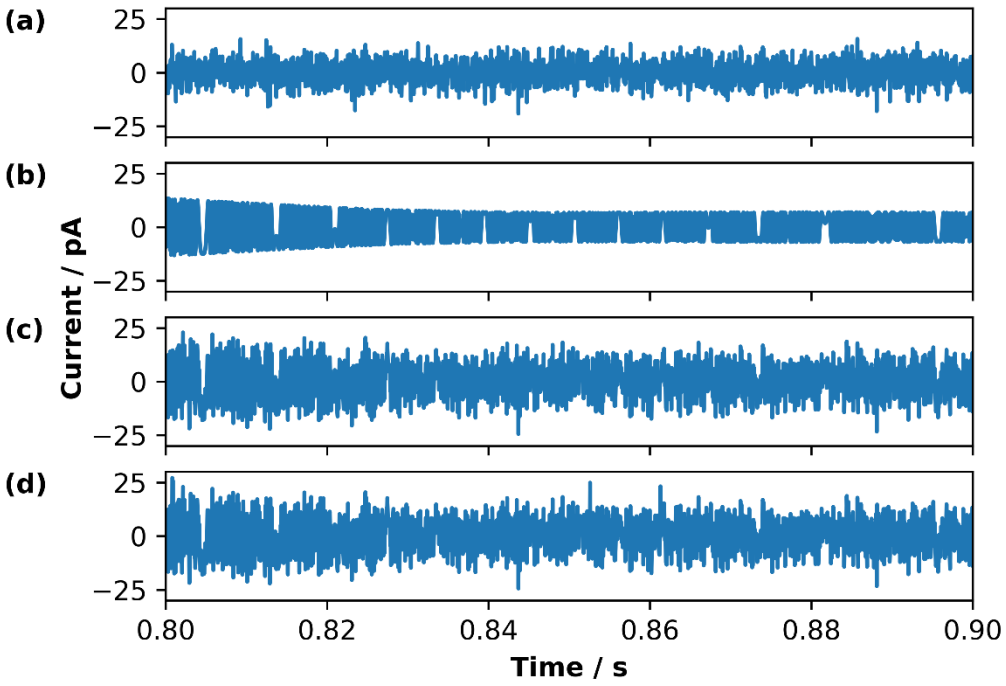



*Figure 3.10*

*Example section of a simulated blank trace. Four steps in the construction process are demonstrated. (a) shows the initial Gaussian noise, whereas (b) shows the generated sinusoid with amplitude and frequency modulation. These are combined to generate what is shown in (c). Lastly, noise events are incorporated to yield the final blank trace shown in (d)*

simple approximation of experimental traces. Therefore, additional features were added to these simulated traces. The first main additional feature was the incorporation of sinusoidal noise. This sinusoidal noise features frequency and amplitude modulation to give traces noise whose amplitude can vary over time within the same trace (Figure 3.10b-c). The second feature of the trace incorporates noise events. These are square wave pulses that are similar to other true events and serve as potential false positive obstacles for event searching. All three of the baseline Gaussian noise, sinusoidal noise, and noise events were added together to provide the end empty traces (Figure 3.10d). For more details on the simulation algorithm, the interested reader is referred to the materials and methods Section 3.4.2. An additional figure displaying the autocorrelation of the baseline noise is also presented within the appendix (Figure A1.2).

After construction of trace baselines, the simulation adds in I(t) events that correspond to a molecular binding event. To do this, the same logic for embedding noise events is used, but with different values for the event's height, duration, and current modulation. To be more specific, the event generation algorithm iterates through a given baseline trace. On each iteration a random number is uniformly generated between zero and one. If this value is less than a specified event chance an I(t) event is generated at this location. The event duration is sampled from a normal distribution with a user-defined average and standard deviation. An event height is also sampled from another normal distribution with user-defined parameters. This event height is subsequently plugged into a third normal distribution as the parameter for its average with the standard deviation being defined by yet another normal distribution with user defined parameters. The complexity of generating events results in no two events having the same height, duration, nor distribution of current values during the event. The

purpose of this randomisation of the intra-event currents was to provide a unique pattern of

*Table 3.2*

*Summary of simulation event generation parameters used by each of the 5 classes.*

| Class | $f$ | $\bar{d}$ | $\sigma_d$ | $\bar{h}$ | $\sigma_h$ | $\bar{c}$ | $\sigma_c$ |
|---|---|---|---|---|---|---|---|
| **Phos** | 20 | 2 | 0.5 | 0.025 | 0.001 | 0.001 | 0.0000 |
| **rAMP** | 20 | 25 | 5 | 0.050 | 0.005 | 0.006 | 0.0012 |
| **rTMP** | 15 | 30 | 6 | 0.070 | 0.007 | 0.004 | 0.0008 |
| **rCMP** | 10 | 20 | 4 | 0.040 | 0.003 | 0.009 | 0.0018 |
| **rGMP** | 20 | 35 | 7 | 0.070 | 0.009 | 0.001 | 0.0002 |

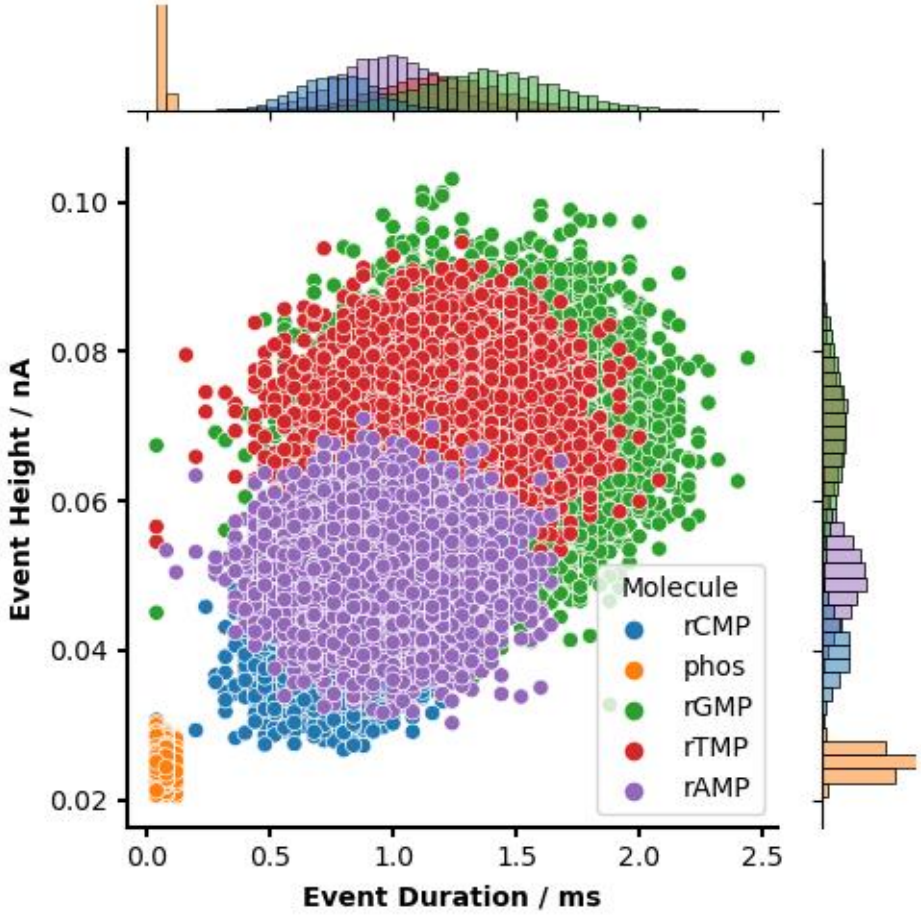



*Figure 3.11*

*Traditional analysis results of simulated events. Both the central scatter plot of the event durations vs event heights and the marginal histograms show that the event properties have considerable overlap. This shows that the simulation offers a difficult classification task.*

values for each event that would not be seen by the traditional analysis techniques. More details on the event creation process are outlined in the materials and methods Section 3.4.2.

A dataset was created using the above simulation. This dataset consisted of five classes to represent the four nucleotides and a control class (Figure A1.3). The values used for the simulation's normal distributions are summarised in Table 3.2. Upon creation of the dataset, the traditional statistical approach was employed for analysis. This involved studies of the event heights and durations to represent the difficulty in separating each class based on traditionally used methods. The measured event properties, as well as the distributions specified in the simulation are shown in Figure 3.11. From this, it can be seen that the event heights and durations overlap considerably, which is synonymous with the findings of the literature.[1,2,16,17]

For the purpose of testing different classification algorithms, the following methods were chosen, a naïve Bayesian classifier (NBC), a KNN classifier, a Random Forest Classifier (RFC), an SVM, a FFNN, and a CNN (Figure 3.14). The NBC was chosen to give an initial insight into how statistical analysis of the event distributions would perform. From this baseline, classifiers were chosen with increasing complexity whilst also investigating those that have been previously mentioned in the literature.[18,19]

As each event has a varying duration, their vector representations will have different dimensionalities. Therefore, all events were padded with zeros such that each event had the same number of features. The dataset was split into a training and testing set such that the test set contained 5% of the dataset. The data points were chosen, for each set, at random whilst maintaining the same class balance. Each classifier was subsequently trained on the

training set and benchmarked using the test set. The metrics chosen were the model's accuracies, their subsequent confusion matrices, and the Matthews Correlation Coefficient (MCC).[20] The results of each method are summarised in Table 3.3 and Figure 3.12. Upon inspection of the MCC scores, it can be seen that the worst performing method is the NBC

*Table 3.3*

*Classification results of the chosen ML agents. The chosen classification metrics shown are accuracy and MCC.*

| Classifier | Accuracy | MCC |
|---|---|---|
| **NBC** | 0.487 | 0.397 |
| **KNN** | 0.781 | 0.722 |
| **RFC** | 0.946 | 0.931 |
| **SVM** | 0.800 | 0.747 |
| **FFNN** | 0.804 | 0.752 |
| **CNN** | 0.950 | 0.937 |

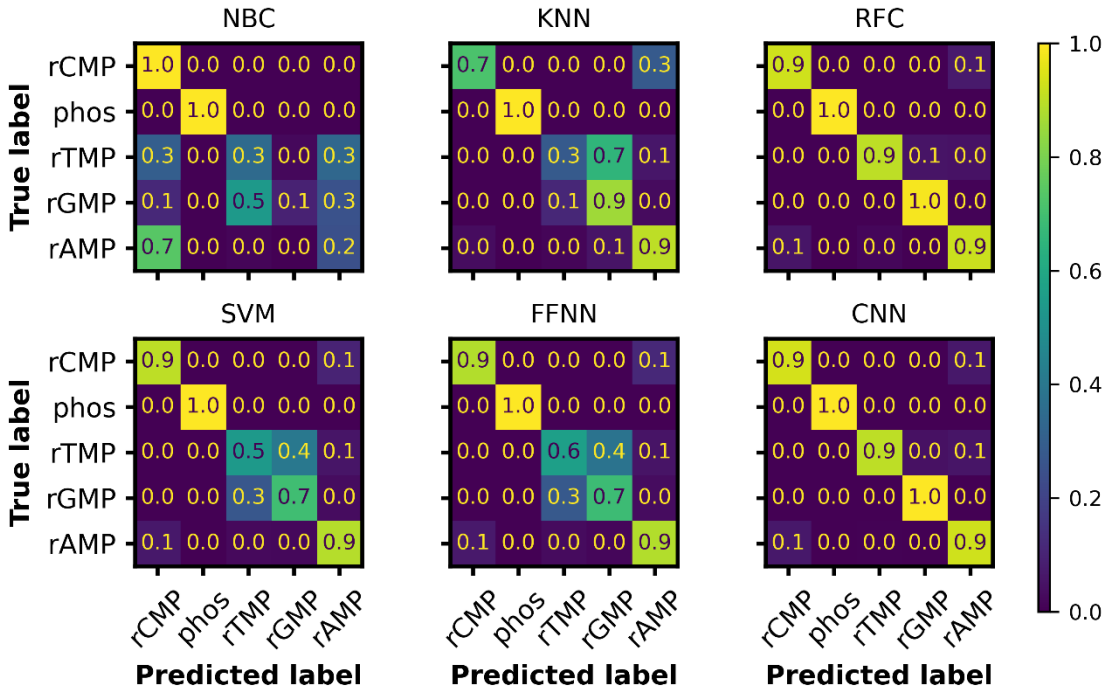



*Figure 3.12*

*Confusion matrices for each of the tested ML classifiers. The most successful agents were found to be the CNN and RFC agents.*

method. This reinforces the notion that classical statistical methods are insufficient for this task as a method that uses Bayesian statistics cannot distinguish between classes due to large overlaps in their distributions. The next most accurate models are the KNN, SVM, and FFNN classifiers with each attaining an MCC score of roughly 0.74. Lastly, the most accurate classifiers were the RFC and the CNN models which were both able to score an MCC value around 0.93. This is an impressive classification score as the overlaps in event properties would be a strong factor for one to believe that classification would be impossible. Another interesting result presents itself when comparing the performance of the RFC and the CNN. Here, a standard ML technique was able to achieve an accuracy that is comparable to a popular DL agent, a CNN. This proves that extraordinary results can be achieved by standard ML techniques without having to resort to more complicated neural network techniques.

Upon inspection of the confusion matrices in Figure 3.12, it can be seen that, indeed, RFC and CNN have high classification accuracies with the majority of the population of their confusion matrices being located on the diagonal. However, when considering the KNN, SVM and FFNN classifiers, it is apparent that most of the classification confusion is caused between the rTMP and rGMP classes. This is understandable given that these two classes have the most overlap when looking at their height and duration distributions (Figure 3.11). However, when considering the NBC, confusion is very unsymmetrical. For instance, the NBC has a 70% chance of incorrectly classifying a rAMP event as a rCMP event but also has a 100% rCMP event classification accuracy. This shows that the NBC has a high propensity for choosing rCMP as the predicted class. This is surprising given that the rCMP class has the fewest training points within the dataset. This type of behaviour can also be seen for the classification of other classes with the NBC algorithm.

## 3.3. Conclusion

In summary, the goal of the studies present in this chapter was to demonstrate that utilising tunnelling current based techniques for SM sensing could be a viable strategy that requires more investigation. To do this the example of nucleotide classification for a potential sequencing technology was used. Demonstration of this potential was split into two sub projects of nucleotide sensing using an ECSTM, and then classification of I(t) events.

The first goal of measuring nucleotides using the STM I(t) technique was achieved. Whilst successful, it is however a very difficult process with a considerable amount of noise characterisation and repeats being required. To be able to measure events with a reasonable reliability, it was required to characterise the ECSTMs noise relationship with the setpoint, sample bias, and electrode potential operating parameters. Additionally, these parameters will have to be tailored to account for the physical dimensions of the molecule of study. All of the above factors make this a non-trivial process. Given this preliminary success in achieving nucleotide detection, future work should shift towards acquiring more statistical repeats to provide a more concrete defence that this procedure is successful. In addition to this, it is also recommended that an ECSTM with greater noise dampening capabilities be used such as one capable of achieving a more constant tip-substrate separation.

The recommendation for future study within this area is motivated further by the successful classification results using simulated data. Here, an I(t) simulation was devised to incorporate findings within the literature with regards to nucleotide event properties, whilst providing a difficult classification challenge. Previously, in the literature, ML applications have only been able to apply agents such as SVM to experimental data, with simulated data challenges having

had DL techniques such as CNNs applied to entire section of traces containing I(t) events. Following this trend, the second goal of this chapter reinvestigated AI techniques for I(t) event classification. Here a broader range of different ML techniques was used and compared. It was found that an RFC and CNN were both able to classify I(t) events with a 95% accuracy when more traditional techniques would fail. Therefore, this chapter concludes that ML would be a viable solution to the classification problem present in the sequencing by tunnelling field.

## 3.4. Materials and Methods

### 3.4.1. Scanning Tunnelling Microscopy

For the purposes of carrying out the I(t) and I(s) experiments, an Agilent 5100 series ECSTM was employed. The setup also featured a scanner operating at a sensitivity of 10 nA/V allowing for a maximum measured current of 10nA. To accompany this, a molecular imaging bipotentiostat was used to allow for electrochemical control.

#### *3.4.1.1. Substrate Preparation*

The substrate used for both I(s) and I(t) experiments takes the form of a Au(111) disc. Before use in the ECSTM a cleaning procedure is carried out.

Before each experiment the substrate is first electrochemically cleaned. The substrate is immersed in a solution of 0.5M sulphuric acid whilst applying a DC voltage of 5V between itself and a counter electrode of the same material. This voltage is held for 30 seconds until the surface of the substrate has been oxidised. Next the substrate is immersed in a solution of 1.0M hydrochloric acid. This dissolves the substrates surface oxide layers resulting in a fresh surface. Any residues are then washed off using ultrapure water (18MΩ). This process is repeated three to five times.

Finally, the substrate is flame annealed. To flame anneal, the substrate is heated by a hydrogen flame until the substrate glows red. The substrate is kept at this temperature for a minute before being cooled in a nitrogen environment.

In addition to cleaning the substrate, cleaning of the cell that holds the substrate in place is also carried out. Here a cell made of Teflon is first heated in concentrated sulphuric acid for 30 minutes. After this, the Teflon along with any acid resistant tweezers and additional glassware is immersed in a solution of acidified potassium permanganate for a minimum of 12 hours. After this the solution is drained and each component is rinsed three times using ultrapure water (18MΩ) before being immersed in the water. Roughly 25ml of a dilute mixture of 1:3 hydrogen peroxide and sulphuric acid is then added to dissolve any organic residues remaining on any components. This is left for 30minutes to ensure all organic matter is decomposed. Lastly, the cleaning solution is drained, and components washed again using ultrapure water before then being boiled in ultrapure water three times.

#### *3.4.1.2. Tip Preparation*

For a successful STM experiment, a piece of metal wire with an, ideally, atom sharp tip is placed into the nosecone of the scanner head. To create this tip a 2cm piece of gold wire with a 0.25 mm thickness is cut. This piece of wire has roughly 1mm of wire immersed in an etching solution of a 1:1 mixture of ethanol and hydrochloric acid. A voltage of 12V is applied between the gold wire and a platinum counter electrode. The tip is removed once the current drops to 0A. The immersed end of the wire is then washed in ethanol and ultrapure water (18Ω) before being placed under a microscope to assess the sharpness of the tip. If the tip is sufficiently

sharp without any warping, and has no contamination, the tip is embedded in a foam holder inside a container such that the tip is not damaged during transit.

An additional step is taken to coat the tip when utilising electrochemical control. To do this a small piece of ethylene-vinyl acetate (EVA) copolymer is melted on a metal surface over a small, millimetre scale hole. A previously sharpened tip is then place into a pin vice mounted on a vertical actuator located below the hole. The tip is slowly raised up through the bottom of the hole through the melted EVA copolymer to a height such that roughly 1cm of the tip's length is coated. The tip is then lowered back through the hole to yield a fully covered tip. However, STM experimentation requires that the very apex of the tip is exposed for tunnelling current to be measured. To re-expose the apex of the tip the opposite end of the wire is gently heated using a propane torch. This gentle heating causes the EVA copolymer to melt and shrink away from the apex thus revealing the tip. During this heating process, the tip is constantly checked under an optical microscope to ensure an appropriate amount of tip is exposed. This completed tip is then washed with ultrapure water (18MΩ) and stored as described above.

#### *3.4.1.3. Sample Preparation*

All experiments utilised a 0.1M phosphate buffer liquid medium. A 200ml stock solution was created by mixing 1.165g of anhydrous potassium phosphate monobasic with 3.097g of potassium phosphate dibasic in a solution of 100ml of ultrapure water (18Ω). The volume was then made up to the desired 200ml using more ultrapure water.

A 1ml stock solution of 5mM rAMP was created by dissolving 1.74mg of adenosine 5'-monophosphate sodium salt in the 0.1M phosphate buffer solution. From this concentrated

stock, a 100µl solution of the desired 100µM concentration was made by mixing 2µl of 5mM stock with 198µl of 0.1M phosphate buffer solution.

#### *3.4.1.4. STM Assembly*

To set up an experiment, an STM tip is first washed in ultrapure water (18Ω) before being mounted in the scanner. This scanner is then plugged into the microscope body and clamped in place. Next, the STM sample dish is assembled. To assemble, the cleaned substrate is placed in the middle of the trays conductive plate. The tray is then mounted into the microscope being careful not to make contact with the mounted tip. The sample tray is then manually brought closer to the tip by adjusting each of the trays mounts until the tip-substrate distance is roughly equal to the width of the tip (250µm). The tray is then removed from the microscope after the above distance has been set, so the electrochemical cell can be attached.

The cleaned cell is screwed to the conductive plate such that the substrate is visible in the centre. The reference and counter electrodes (Pt/Ir) are cleaned and subsequently mounted on the sample tray. These are placed such that the electrode tips are in close proximity to the sample. To clean these electrodes, they are flamed using a propane torch until the wires brightly glow before being mounted. After the electrochemical cell is attached and the electrodes fitted, electrochemical control is switched on using the controlling software. Next, the solution of study is aliquoted into the cell such that the substrate surface is covered as well as both the refence and counter electrodes. The measured potential of the substrate is then compared to the desired potential to ensure that electrochemical control is functioning as desired.

Lastly, the sample tray is remounted into the microscope taking care not to touch either of the reference or counter electrodes to the STM tip, whilst also taking care not to crash the tip into the substrate. If successful, the setup is then ready for approach and subsequent experimentation.

### 3.4.2. I(t) Simulation

As previously stated in Section 3.2.4, the I(t) simulation can be split into two parts: baseline construction; and event construction. Here it was specified that the baseline is comprised of three components, the Gaussian noise, sinusoidal noise, and the noise events. The Gaussian noise was generated by sampling 25000 points from the normal distribution, $N(0.000, 0.005)$. The sinusoidal noise was more complicated to generate as it featured both frequency and amplitude modulation. Let the sinusoidal noise be represented by $S$ such that:

$$S(t) = AM(t) * A * \ sin(FM(t) * F * t \ + \varphi) \tag{3.2}$$

where $t$ is the time within the trace, $AM(t)$ is the amplitude multiplier at that time, $A$ is the amplitude of the noise, $FM(t)$ is the frequency multiplier at that time, $F$ is the frequency of the sinusoid, and $\varphi$ is the phase of the sinusoid. The unmodified amplitude, frequency, and phase of the noise are sampled once per trace from the following distributions:

$$A = N(0.01, 0.005) \tag{3.3}$$

$$F = N(500, 100) \tag{3.4}$$

$$\varphi = U(0, 2\pi) \tag{3.5}$$

where $U$ is a uniform distribution. However, the amplitude and frequency multipliers are not constant per trace and are given by the following equations:

$$AM(t) = 1 + \ N(0.5, 0.05) * \sin\big(AMM(t) * t + U(0, 2\pi)\big) \tag{3.6}$$

$$AMM(t) = N(5,1) * \sin(N(10,2) * t + U(0,2\pi)) \quad 3.7$$

$$FM(t) = 1 + N(100,10) * \sin\big(FMM(t) * t + U(0,2\pi)\big) \quad 3.8$$

$$FMM(t) = N(10,2) * \sin(N(10,0) * t + U(0,2\pi)) \quad 3.9$$

It is important to note that, likewise to $S$, the normal and uniform distributions within the functions for their amplitudes, frequencies, and phases are only sampled once and then used as a constant for every value of $t$. Examples of the resultant components are shown in Figure 3.13. After the incorporation of the sinusoidal noise, noise events are subsequently incorporated. This process is identical to the event generation component as described below which means that for each dataset event generation occurs twice. Once for the noise events, and then again for the molecular binding events.

The event generation component of the simulation proceeds as described in Section 3.2.4. For clarity, each event samples from the previously described normal distributions to obtain the following parameters: event height, $h$, and event duration $d$. With the sampled height and duration, the simulation then overwrites the points after the event start with a sampled array of values given by the function, $C$. This logic can be shown as follows:

$$h = N\big(\bar{h}, \sigma_h\big) \quad 3.10$$

$$d = N(\bar{d}, \sigma_d) \quad 3.11$$

$$C = N(h, N(\bar{c}, \sigma_c)) \quad 3.12$$

Here, the function, $C$, that is used to populate the current values during an event is a normal distribution whose standard deviation is sampled from another distribution per event. This

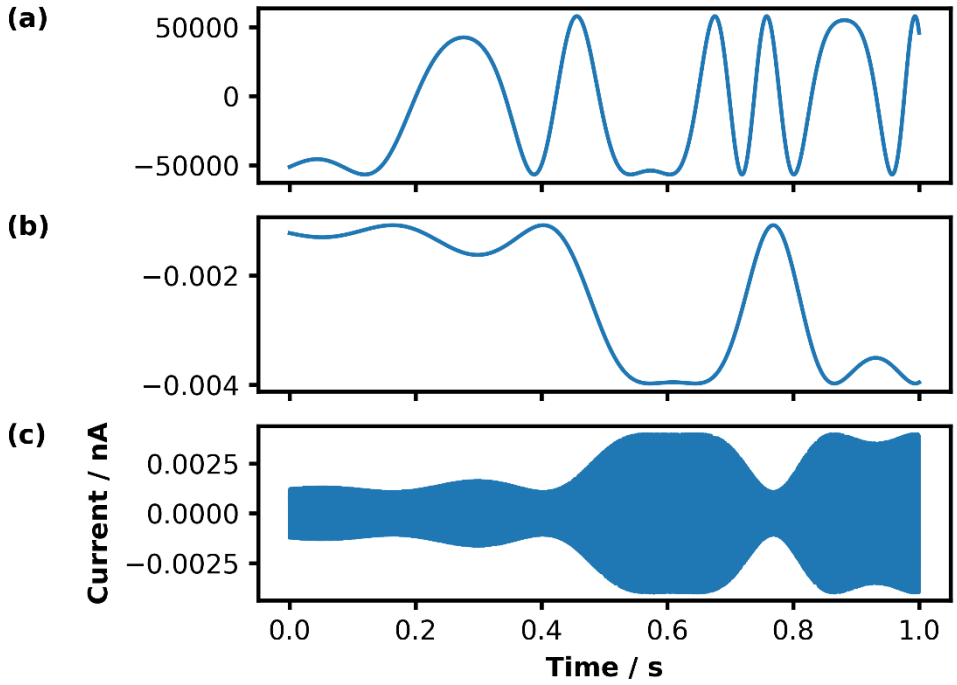



*Figure 3.13*

*Example sinusoid construction for background noise. (a) and (b) show the frequency and amplitude modulation traces that are used to generate the sinusoid shown in (c).*

yields six total parameters for describing the behaviour of each event, with a seventh parameter controlling the event frequency.

### 3.4.3. Machine Learning

Programming of the data analysis processes was carried out using the python programming language with a virtual environment being managed by anaconda. The environment features common data science packages such as numpy (v1.23.3), pandas (v1.4.4), scipy (v1.9.1), matplotlib (v3.5.2), and seaborn (v0.12.0). All analysis scripts were composed in either python files or jupyter notebooks. ML analysis made use of the scikit-learn packages (v1.1.2) for all non-neural net-based models whereas the neural network programming utilised the tensorflow packages (v2.10.0), (Appendix 4).

The NBC was implemented by utilising the GaussianNB class provided by scikit-learn which holds the assumption that each classification group is normally distributed. For fitting, no prior probabilities were set, and all parameters were left as their default values.

The KNN classifier was implemented using the KNeighborsClassifier class provided by scikit-learn. Here, the k parameter was optimised by iteratively searching through the range of 1 to 1000 (Figure A1.4). The optimum parameter was found to be a k value of 196 as this yielded the highest MCC score. All other parameters were left as their default values.

The RFC was implemented using the RandomForestClassifier class provided by scikit-learn. Here the number of trees within the forest was optimised by iteratively searching through the range of 1 to 200 trees (Figure A1.5). The optimum number of trees was found to be 75 as this yielded the highest MCC score. All other parameters were left as their default which included the use of the Gini Impurity for the loss function and no limits to the maximum tree depth.

The SVM classifier was implemented using the SVC class provided by scikit-learn using all default parameters. These default parameters include the use of the radial basis function as the model's kernel function.

Lastly, the FFNN and CNN were both implemented using the functional API of TensorFlow. More specifically, Keras was used to construct the different architectures which are displayed in Figure 3.14. Both models used the sparse categorical crossentropy (SCC) loss metric and were trained using the Adam optimiser with a learning rate and decay rate of $5 \times 10^{-5}$ and $1 \times 10^{-6}$ respectively.

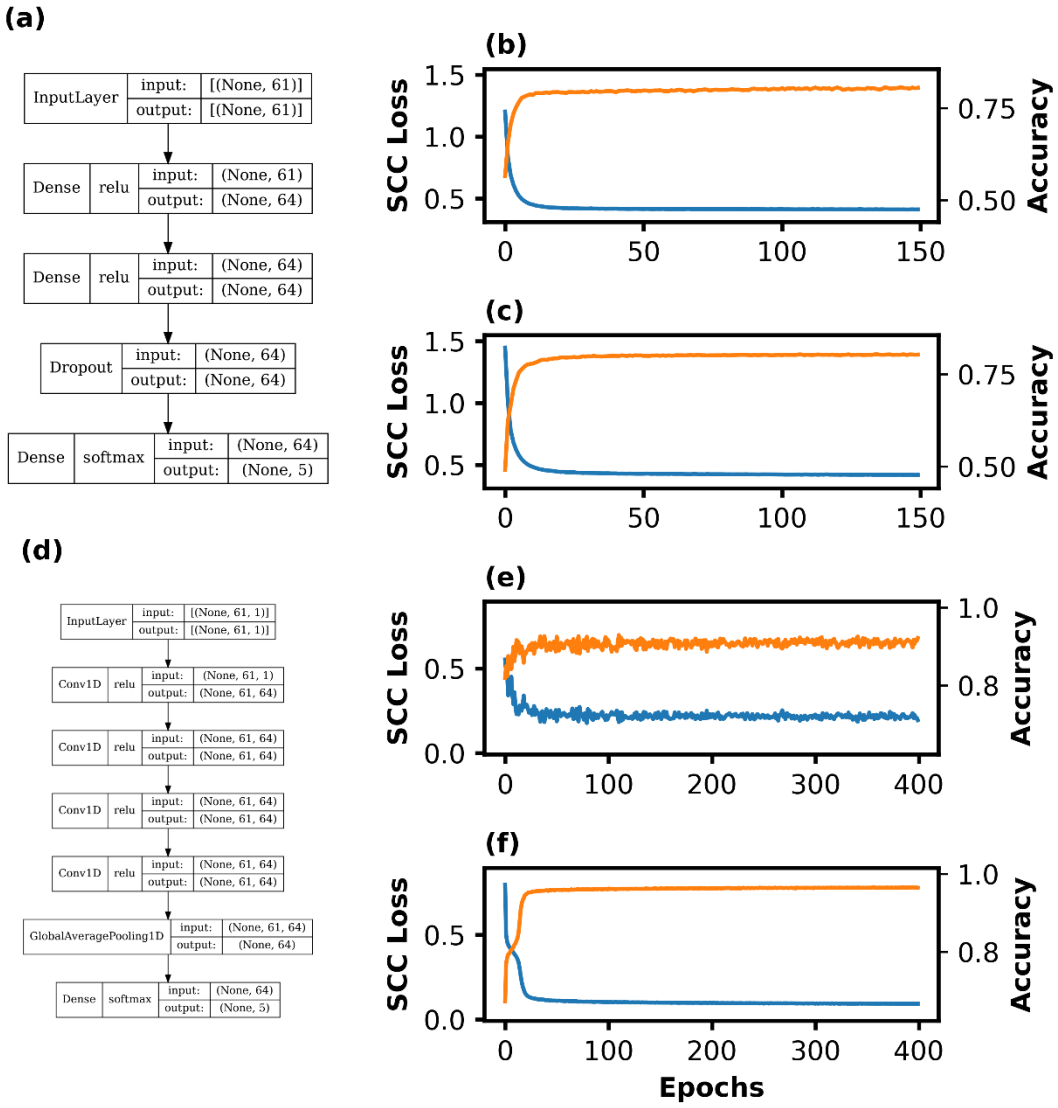



*Figure 3.14*

*Summary of neural network classifiers. (a) and (d) show the network architectures of the FFNN and CNN respectively. These show the used activation functions and the output dimensionality for each layer. (b) and (e) show the SCC validation loss at each epoch during training for the FFNN and CNN respectively. (c) and (f) show the SCC training loss at each epoch during training for the FFNN and CNN respectively.*

## 3.5. References


1 J. Lagerqvist, M. Zwolak and M. Di Ventra, *Nano Lett*, 2006, **6**, 779–782.

2 S. Chang, S. Huang, J. He, F. Liang, P. Zhang, S. Li, X. Chen, O. Sankey and S. Lindsay, *Nano Lett*, 2010, **10**, 1070–1075.

3 S. Chang, S. Sen, P. Zhang, B. Gyarfas, B. Ashcroft, S. Lefkowitz, H. Peng and S. Lindsay, *Nanotechnology*, 2012, **23**, 425202.

4 J. F. Ge, M. Ovadia and J. E. Hoffman, *Review of Scientific Instruments*, 2019, **90**, 101401.

5 K. Iwaya, R. Shimizu, T. Hashizume and T. Hitosugi, *Review of Scientific Instruments*, 2011, **82**, 083702.

6 A. C. Marvin, J. F. Dawson, S. Ward, L. Dawson, J. Clegg and A. Weissenfeld, *IEEE Trans Electromagn Compat*, 2004, **46**, 459–468.

7 C. J. Chen, *Introduction to Scanning Tunneling Microscopy*, Oxford University Press, Oxford, 2nd edn., 2007.

8 J. Halbritter, G. Repphun, S. Vinzelberg, G. Staikov and W. J. Lorenz, *Electrochim Acta*, 1995, **40**, 1385–1394.

9 N. Elgrishi, K. J. Rountree, B. D. McCarthy, E. S. Rountree, T. T. Eisenhart and J. L. Dempsey, *J Chem Educ*, 2018, **95**, 197–206.

10 E. M. Espinoza, J. A. Clark, J. Soliman, J. B. Derr, M. Morales and V. I. Vullev, *J Electrochem Soc*, 2019, **166**, H3175–H3187.

11 A. Lasia, in *Handbook of Fuel Cells*, John Wiley & Sons, Ltd, Chichester, UK, 2010.

12 N.-T. Suen, S.-F. Hung, Q. Quan, N. Zhang, Y.-J. Xu and H. M. Chen, *Chem Soc Rev*, 2017, **46**, 337–365.

13 C. Jeyabharathi, U. Hasse, P. Ahrens and F. Scholz, *Journal of Solid State Electrochemistry*, 2014, **18**, 3299–3306.

14 B. Xu and N. Tao, *Science*, 2003, **301**, 1221–1223.

15 R. Gao, M. A. Edwards, J. M. Harris and H. S. White, *Curr Opin Electrochem*, 2020, **22**, 170–177.

16 M. Tsutsui, M. Taniguchi, K. Yokota and T. Kawai, *Nat Nanotechnol*, 2010, **5**, 286–290.

17 T. Ohshiro, K. Matsubara, M. Tsutsui, M. Furuhashi, M. Taniguchi and T. Kawai, *Sci Rep*, 2012, **2**, 501.

18 T. Albrecht, G. Slabaugh, E. Alonso and S. M. R. Al-Arif, *Nanotechnology*, 2017, **28**, 423001.

19 J. Im, S. Sen, S. Lindsay and P. Zhang, *ACS Nano*, 2018, **12**, 7067–7075.

20 J. Yao and M. Shepperd, in *Proceedings of the Evaluation and Assessment in Software Engineering*, ACM, New York, NY, USA, 2020, pp. 120–129.

# Chapter 4

## Time Series Anomaly Detection with a Low Signal-to-Noise Ratio

Contents

## 4.1. Introduction

One of the main challenges within the field of data science is the challenge of anomaly detection. The ability to detect outliers within a time series is crucial for both data cleaning, and sensing applications. Because of this, considerable research has investigated different methods for accurately locating anomalies.[1–5] Within the field of SM science, there are several sensing techniques that produce data in the form of a time series. Most notably, the STM I(t) technique[6] and RPS[7] both produce traces of signal vs time data that contain brief periods of interest. As such, locating these regions of interest presents a suitable challenge where anomaly detection techniques could be of benefit.

The challenges of finding events within an STM I(t) trace were previously discussed in chapter 3 when attempting to extract events resulting from rAMP binding. Because of this difficulty, this chapter aims to explore various methods of anomaly detection with the goal of improving detection accuracy at lower SNRs. To carry out this investigation, another I(t) simulation was defined to initially provide an appropriate tool for testing detection at user defined SNRs. With this simulation, anomaly detection was investigated with four different techniques. These were named as the: amplitude thresholding, zero-crossing, convolutional AE, and moving average methods. Details as to how each method works, their limitations, and any additional challenges that resulted are stated and compared to conclude which method is superior for anomaly detection in this use case.

## 4.2. Results and Discussion

### 4.2.1. Simulation

Another signal vs time simulation was constructed to mimic the appearance of experimental I(t) technique traces. However, in this case the nature of noise was simplified such that the SNR between the background and anomalous events could be tuned with greater precision. This simulation generates a blank (event-free) background trace with a user-defined length of normally distributed points centred around zero. From here events are placed into the traces. Event locations are generated in the same way as described in Section 3.2.4. Subsequently, the event height, $h$, is calculated using a desired SNR and the following equation:

$$h = SNR * \sigma \tag{4.1}$$

where $\sigma$ is the standard deviation of the background current. This event height value is added to points stated to lie within an event.

Using this simulation, 10 traces of 1,000,000 data points were generated at various SNRs. In total, 7 different SNRs were chosen along with an eventless reference set to yield 80 total traces. The values for the SNRs were chosen as 10, 5, 2, 1, 0.5, 0.25, and 0.1. For each trace, parameters were selected such that the only unique property between different classes was the SNR. To summarise, a standard deviation of 1pA was chosen for the noise level of the background, and the event duration for every class was set by sampling from a normal distribution with mean and standard deviation of 1000 and 20, respectively. Figure 4.1 demonstrates example events generated at different SNRs as well as a comparison of the distribution of points both during and outside of events. Upon inspection of these distributions, it can be seen that at larger SNRs there is little to no overlap between the

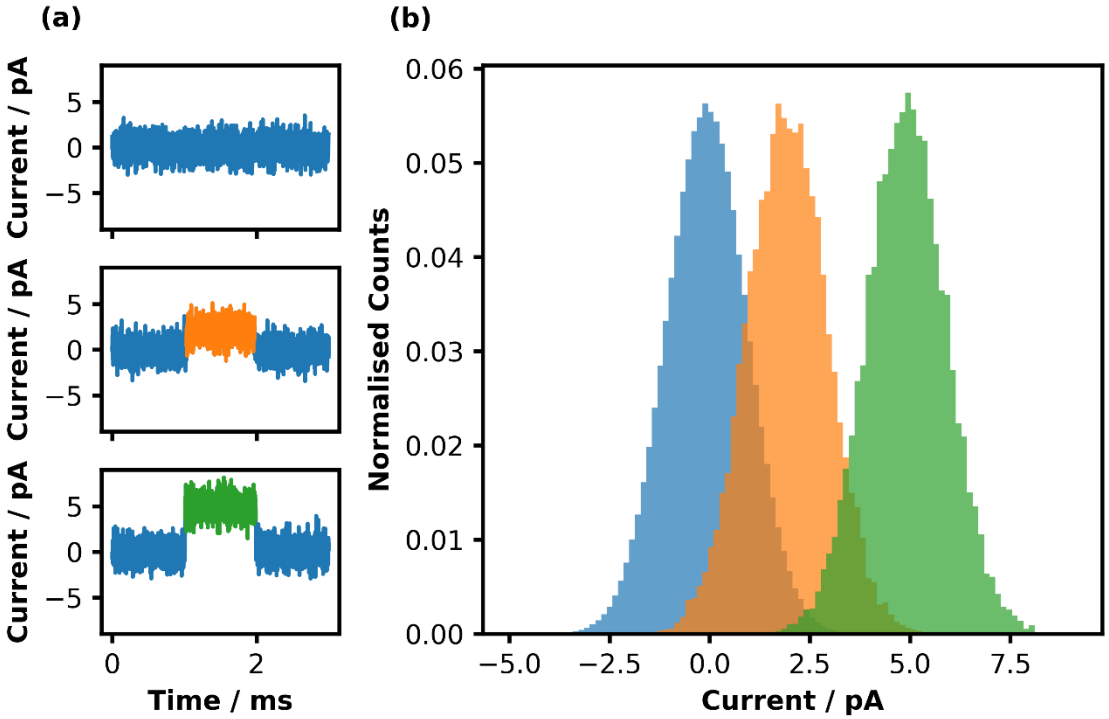



*Figure 4.1*

*Figure showing a comparison of example events produced by the simulation, (a), as well as a comparison of the distributions of current values within events and the background, (b). Here the background is shown (blue) as well as two events with different SNRs of 2 (orange), and 5 (green).*

distribution of event and background data points (Figure 4.1b). However, when considering an SNR below 5, the overlap between these distributions becomes significant which would make perfect separation challenging.

### 4.2.2. Amplitude Thresholding

The amplitude thresholding technique was initially implemented to showcase the challenges of anomaly detection and benchmark the technique used previously in Chapter 3. As described, this technique works by flagging points as anomalous if they are a number of standard deviations from the mean value. The values for this mean and standard deviation are calculated using all of the background, eventless, traces. For event detection, the algorithm functions by iterating through a trace and checking each value sequentially. If a value is above

the chosen threshold, a subroutine is called. This subroutine scopes both forward and backward from the initially flagged point in the same iterating fashion. The subroutine will flag a point as being the start or end of the event if that value is the first value encountered in each direction that falls under a chosen return threshold. In all cases, this return threshold was selected as the mean sampled from the background data. Once the start and end of this event have been located, all points within this range are labelled as anomalies before the algorithm skips passed the event end, and the search routine continues until the entire trace has been scanned.

To trial this technique the thresholds of 5, 4, 3, 2, and 1 standard deviations from the background mean were selected. These thresholds were subsequently implemented on each of the 7 different SNR datasets generated previously. Example events that were flagged are demonstrated in Figure 4.2. It can be seen that, when considering Figure 4.2a and Figure 4.2b, the threshold is too low. Considerable quantities of background data points get falsely flagged as anomalous. This is solved when raising the threshold up to 4 times the standard deviation, as shown in Figure 4.2c. However, raising the threshold can also result in anomalous points being missed. This is demonstrated in Figure 4.2d where the SNR is low enough that sections of true event are missed.

The ground truths were taken from the simulation and compared with the flagged anomalous points to provide a quantified measure of how well the event detection performed. These ground truths were represented as one-hot encodings corresponding to each datapoint. A value of one labelled the respective data point as a true anomalous point. Using both the ground truth and flagged anomalies, the true positive rate (TPR) and false positive rate (FPR)

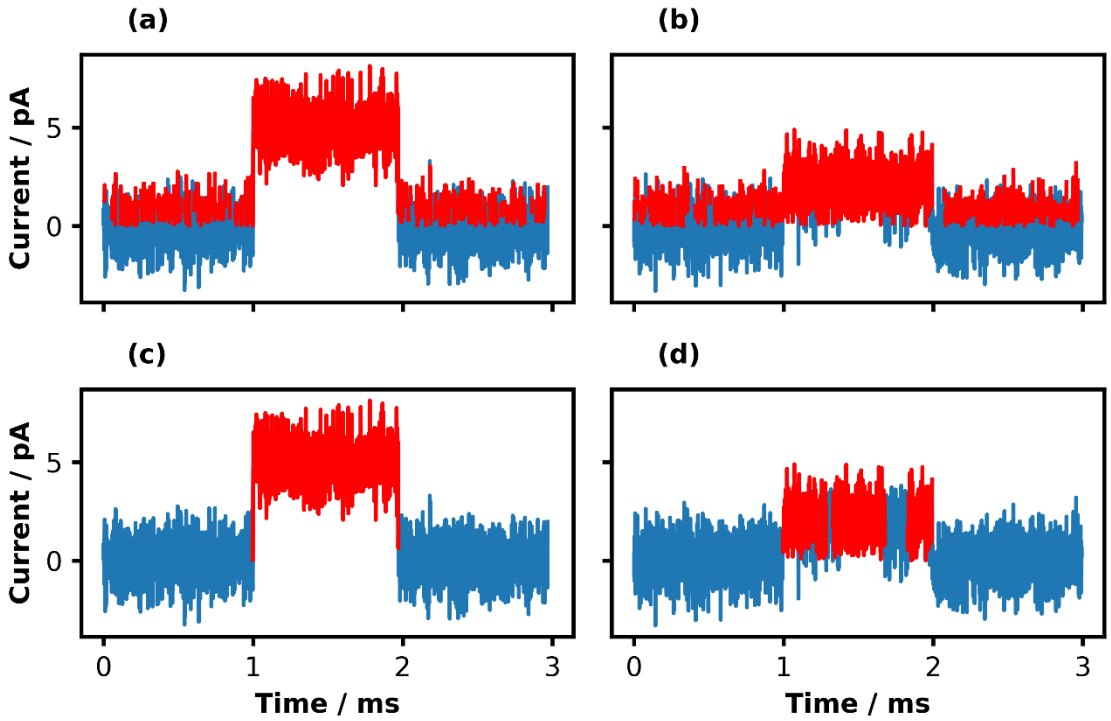



*Figure 4.2*

*Example amplitude thresholding predictions (red) overlayed over a section of simulated trace (blue). Each graph shows an example snippet of trace with a true event situated at the centre. (a) and (c) show the same event with an SNR of 5, whereas (b) and (d) show the same event with an SNR of 2. (a) and (b) show predicted anomalies when using a 1 std threshold, whereas (c) and (d) show predicted anomalies when using a 4 std threshold.*

were calculated. These values for each threshold and SNR are summarised in Figure 4.3. The following equations were used to calculate these quantities of TPR and FPR. Additional equations are provided so the corresponding true negative rate (TNR) and false negative rate (FNR) can be calculated:

$$TPR = \frac{TP}{TP + FN} = 1 - FNR \tag{4.2}$$

$$FPR = \frac{FP}{FP + TN} = 1 - TNR \tag{4.3}$$

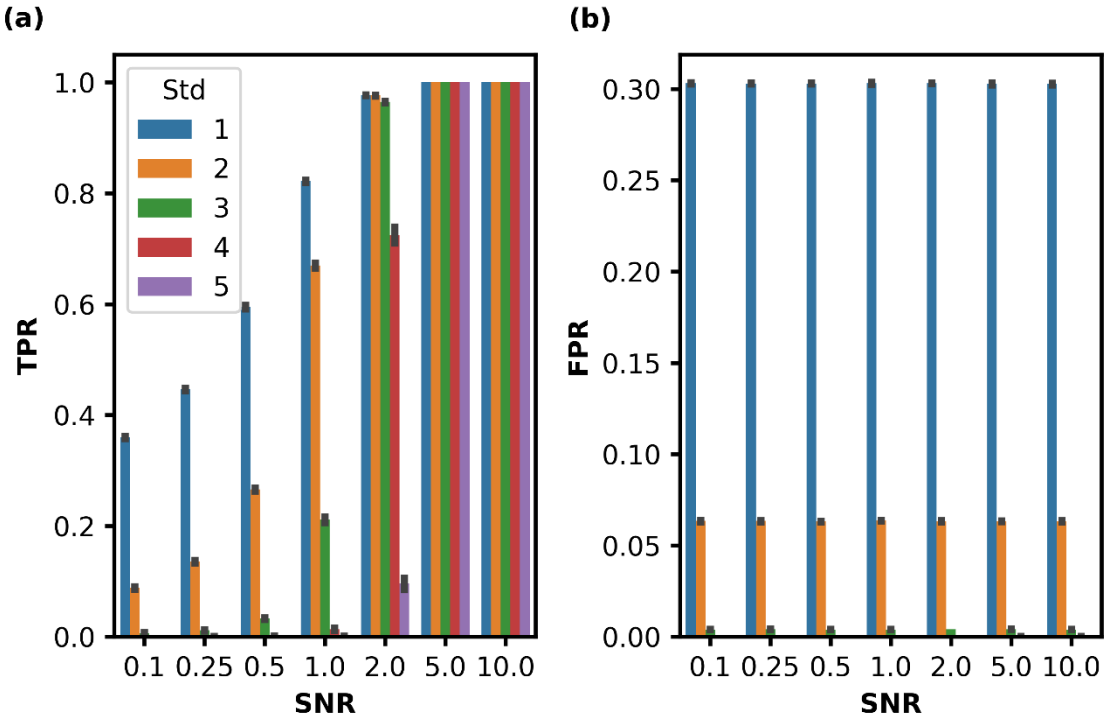



*Figure 4.3*

*Summary of the TPR, (a), and FPR, (b), when labelling anomalies at each combination of detection threshold and event SNR.*

$$TNR = \frac{TN}{TN + FP} = 1 - FPR \tag{4.4}$$

$$FNR = \frac{FN}{FN + TP} = 1 - TPR \tag{4.5}$$

where $TP$ is the number of true positive predictions, $TN$ is the number of true negative predictions, $FP$ is the number of false positive predictions, and $FN$ is the number of false negative predictions.[8,9] These results show that the TPR rate falls off as the SNR of the events decreases. This is due to the anomalous points residing closer to the background average such that less points oscillate outside of the threshold. To remedy this the detection threshold can be lowered which results in a higher TPR. However, as shown in Figure 4.3b, decreasing the threshold results in an increase of the FPR. For example, using a threshold of 1 standard

deviation results in a FPR of 0.30. Another observation is the independence of the FPR to the SNR. This is also expected as the SNR of the events does not affect the nature of background data points. Indeed, thresholds are calculated from the background meaning the same degree of false positives will be returned regardless of the nature of the events.

An overall score for each technique was calculated by considering the performances across all threshold values. This was achieved by taking the TPR and FPR for each threshold value to plot a receiver operating characteristic (ROC) curve. ROC curves allow a classification algorithm to be characterised by investigating the accuracy and error of a classifier at various sensitivities.[10] Starting from a sensitivity where no events are flagged the TPR and FPR will both be 0. For a successful detection algorithm, the desired effect of increasing the algorithms sensitivity would be for the TPR to increase whilst the FPR remains at 0. Once the TPR rate reaches 100%, then the FPR should begin to rise. Alternatively, a poor detection algorithm would have the inverse behaviour with the FPR rising before the TPR increases. If both the TPR and FPR were to increase at the same time, this would also be a poor result as it implies that finding true events also comes with false positives that cannot be distinguished. Considering this logic, the final performance metric is introduced. This is called the area under the ROC curve or simply the area under curve (AUC). As its name suggests, it is calculated by finding the area under the ROC curve. As such, this metric scales from 0 to 1 where a score of 1 corresponds to the perfect detection case, and a score of 0.5 and below indicate poor detection cases.[11]

This metric was applied to the amplitude thresholding algorithm. Using event detection behaviour at each threshold value a ROC curve was populated for every SNR. With a ROC curve for every SNR, the AUC was subsequently calculated and compared. The results of this analysis

are shown in Figure 4.4. The AUC scores show that as the SNR of the events decrease so do the corresponding scores. This is expected as previously it was shown that decreasing the SNR only affects the TPR. The FPR is tied to the algorithms threshold and has no relation to the SNR. Because of this the AUC score drops as well due to the ROC curve achieving high TPRs at higher algorithm sensitivity where the FPR is also high.

In summary, the performance of the amplitude thresholding technique becomes less than acceptable when the events have an SNR of one or less. With this technique characterised, alternative techniques were subsequently investigated and compared.

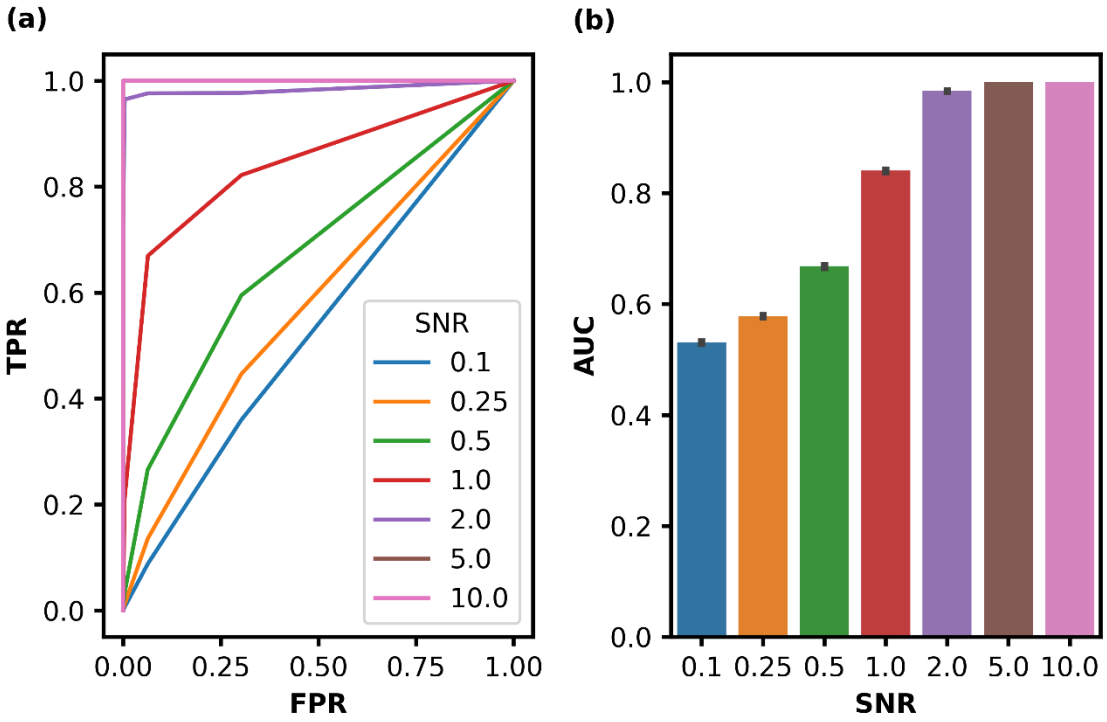



*Figure 4.4*

*Anomaly detection results via the amplitude thresholding method. Shown are the ROC curves for each SNR, (a), and the corresponding AUC scores, (b).*

### 4.2.3. Zero-Crossing Rate

Considering the nature of the events that are to be detected, assumptions can be made regarding value-crossings. Specifically, as the events manifest as pseudo-square wave pulses in amplitude, it would be sensible to infer that during these events, there is a reduced probability that the inherent oscillations will cross the mean of the trace. In this instance, the mean of the simulated traces has been set at zero. Using this logic, an anomaly detection technique can be developed with the criteria that anomalous sections of traces will harbour a reduced zero-crossing probability. This forms the basis of the zero-crossing rate (ZCR) method.

For this technique to function, a single value cannot be sampled when iterating through a trace, as a sample size of at least two is required to provide a number of zero-crossing instances. Therefore, this detection algorithm requires iteration of a sample window through a trace such that a ZCR can be calculated at each window position. Because of this, the ZCR detector requires two additional parameters; the window size and the step size. The window size controls the sample size that is used to calculate a zero-crossing probability at every position governed by the step size.

To carry out a ZCR event search, the baseline behaviour was characterised such that suitable thresholds could be selected for flagging up anomalous window positions. The ZCR values for each window position were calculated on the eventless, background data. These values were subsequently plotted on a histogram where a Gaussian fit was applied. Various percentile thresholds were extracted from this fit to yield a range of desired algorithm sensitivities. To do this, the resultant Gaussian was normalised such that the total area under the fit equalled unity. The normalised fit thus functioned as the background ZCRs probability density

function.[12] To acquire the value that corresponds to a desired percentile, the probability density function is converted to the cumulative distribution function via integration.[13] This

*Table 4.1*

*Summary of chosen percentiles used for thresholding and their corresponding ZCR values.*

| Percentile | Value |
|---|---|
| **0.00 %** | 0.0000 |
| **0.01 %** | 0.3086 |
| **0.10 %** | 0.3411 |
| **1.00 %** | 0.3808 |
| **5.00 %** | 0.4159 |
| **10.00 %** | 0.4346 |
| **25.00 %** | 0.4658 |
| **50.00 %** | 0.5004 |
| **75.00 %** | 0.5351 |
| **100.00 %** | 1.0000 |

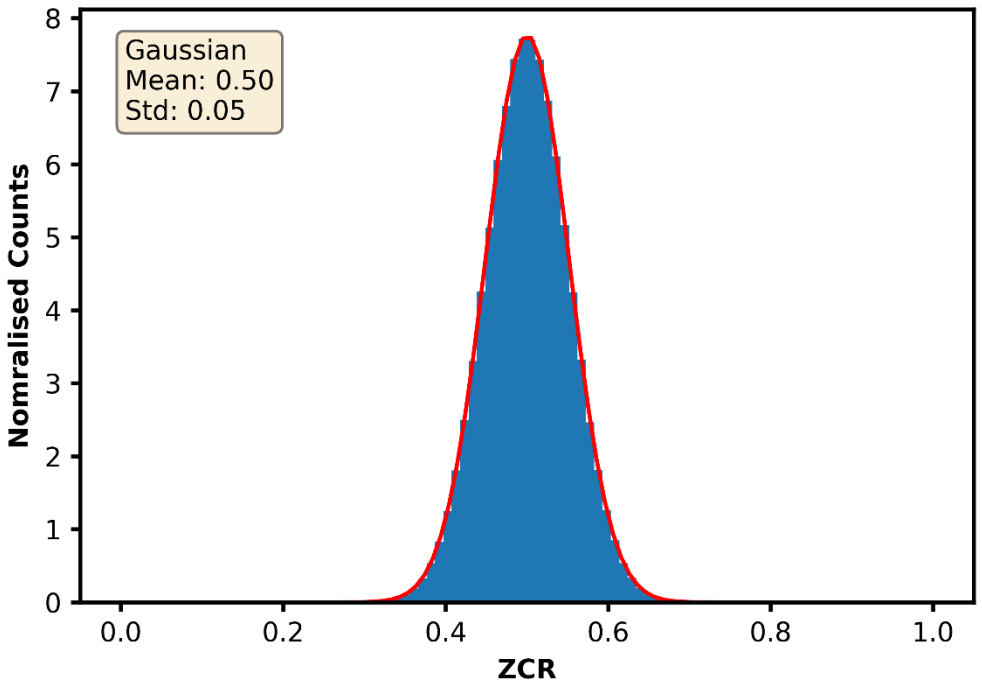



*Figure 4.5*

*Distribution of ZCR values when performing background analysis using a window size of 100 and a step size of 10. The resultant distribution is Gaussian with a mean and standard deviation of 0.50 and 0.05, respectively.*

cumulative distribution function is then solved to find the ZCR value that corresponds to any percentile. This procedure was carried out on the background data with a window size of 100 to yield a distribution demonstrated in Figure 4.5. From this distribution, various percentile values were calculated to trial the detection performance at different sensitivities. These values are summarised in Table 4.1.

Subsequently, the ZCR method was implemented on the simulated data using the predetermined thresholds. Initially, all possible window positions for a given window size and step size were extracted. The number of extracted window positions can be calculated given the following expression:

$$n = \frac{N - (w - s)}{s} \tag{4.6}$$

where $n$ is the number of possible window positions, $N$ is number of data points within a trace, $w$ is the window size, and $s$ is the step size. In this case, each trace contained 1,000,000 datapoints with a window size of 100 and a step size of 10. This yielded a total of 99,991 different window positions. Each window position was then used to calculate the ZCR at each position. With this, each data trace was converted into a new trace of ZCR vs window position (Figure A1.6). Additionally, histogram analysis of these ZCR values reveals the presence of outliers that were not present in the background distribution (Figure 4.6).

The start and end positions of anomalies were found by employing change-point detection using a similar process to the amplitude thresholding technique. Here, the ZCR values were iterated through sequentially until a ZCR value was encountered that fell short of the chosen percentile threshold. Again, from here a subroutine was called that scoped out the

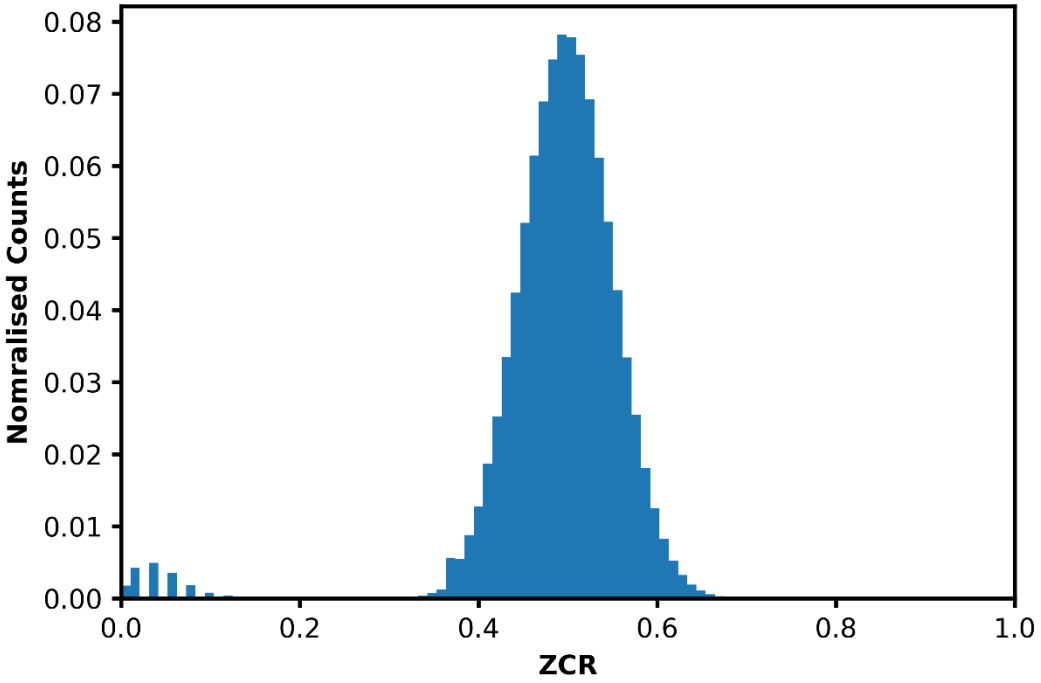



*Figure 4.6*

*Distribution of ZCR values calculated from event traces with an SNR of 2. ZCRs were calculated with a window size of 100 and a step size of 10. The distribution of values below 0.2 are a result of anomalies that were not present in the background trace.*

surrounding values until values were found that rose back above a return threshold, namely the same percentile threshold that was used to first find anomalous windows. This process returns a list of start and end window positions. Equation *4.6* is then used to transform these values back to index positions in the original signal space before all the points within event ranges are flagged as anomalous.

After returning detected events from the algorithm, analysis again moved to the calculation of a quantitative measure of accuracy using the same metrics used previously (Section 4.2.2). Equations *4.2* to *4.5* were used to calculate the TPR and FPR of the algorithm at each sensitivity which were subsequently used to plot ROC curves and provide AUC scores (Figure 4.7). Here

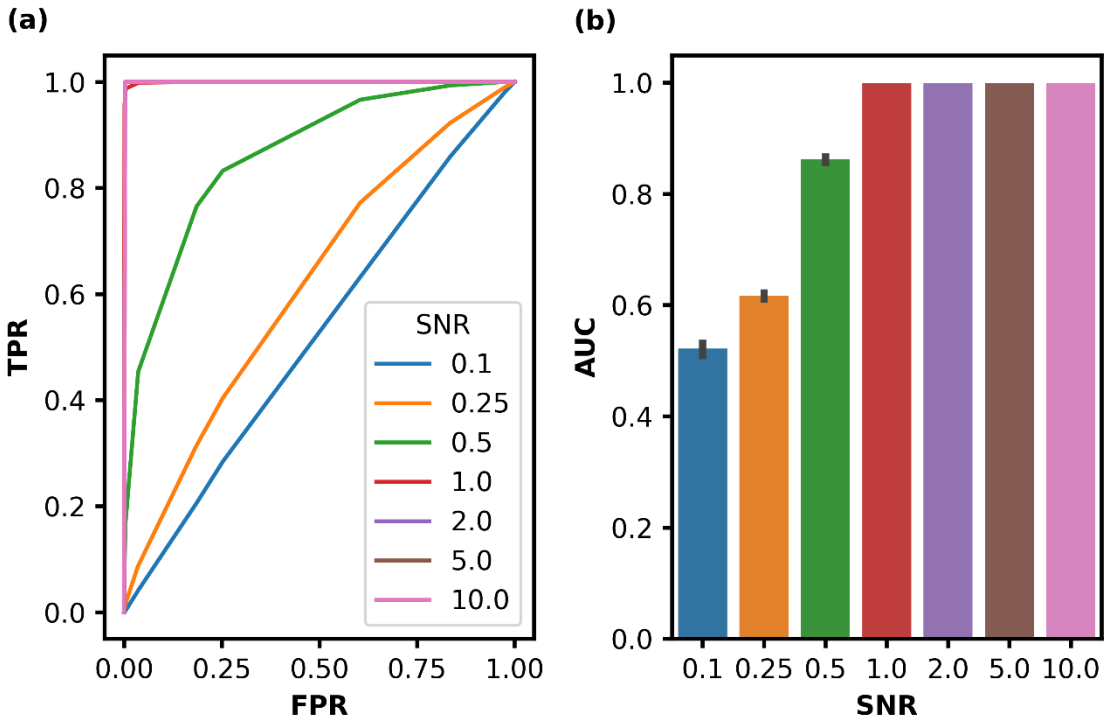



*Figure 4.7*

*Anomaly detection results via the ZCR method. Shown are the ROC curves for each SNR, (a), and the corresponding AUC scores, (b).*

it can be seen that the ZCR anomaly detection technique performs with a score of 1.00 for the SNRs of 1.0 and above. In comparison, the previous amplitude thresholding technique only achieved a score of 1.00 on the SNRs of 5.0 and 10.0 and had a score of 0.98 with an SNR of 2.0. Additionally, the ZCR algorithm achieved an AUC of over 0.8 on the SNR of 0.5 whereas, the previous method scored below. However, the ZCR method did not perform well on the lower SNRs of 0.25 and 0.1.

### 4.2.4. Convolutional Autoencoder

Currently, anomaly detection is also being explored from a DL perspective. Indeed, many investigations into the use of AEs for anomaly detection have been carried out.[14–16] As mentioned in Section 2.3.5, these neural networks are trained to reconstruct input data via a

lower dimensional vector space. Once these AEs are trained, there will be a reconstruction error that is associated with each input. Anomaly detection via these AE agents uses this reconstruction error to detect outliers, as anomalous inputs would result in larger than normal reconstruction errors.

For the implementation of an AE-based anomaly detection pipeline, the same percentile strategy that was implemented with the ZCR method has been adopted. More precisely, the simulated background data were split into windows of size 100 with steps of 10 between each window position. The AE was then trained on these windows with the goal of accurately reconstructing each segment of data. After training, the MAE loss for each window was calculated before being plotted on a histogram. From here, the same strategy of fitting a Gaussian distribution to the data such that percentile values could be calculated was also adopted. The resultant histogram and Gaussian fit are shown in Figure 4.8. It can be seen that the resultant distribution of losses is positively skewed. Because of this, the Gaussian fit does not accurately reflect the distribution. A possible reason for this skew would be the incorporation of both the training and validation datasets. As the neural network was trained on the training data, a similar but different reconstruction accuracy will arise when processing the validation data. This difference depends on the degree to which the network overfitted to the training data.

As the population of the validation dataset is much smaller than the training set and the loss difference between these sets is quite small, the resulting distribution could be the additive result of the two distributions for each respective dataset. To remedy this fitting issue, there are a few possible solutions. One is to retrain the network on both the training and validation

sets once validation is passed, another is to discard either the validation or training set from the background analysis process once validation is passed, another is to fit a skewed

*Table 4.2*

*Summary of chosen percentiles used for thresholding and their corresponding MAE values.*

| Percentile | Value |
|---|---|
| **0.00 %** | 0.0670 |
| **25.00 %** | 0.0948 |
| **50.00 %** | 0.0992 |
| **75.00 %** | 0.1035 |
| **90.00 %** | 0.1074 |
| **95.00 %** | 0.1098 |
| **99.00 %** | 0.1142 |
| **99.90 %** | 0.1191 |
| **99.99 %** | 0.1228 |
| **100.00 %** | 0.1443 |

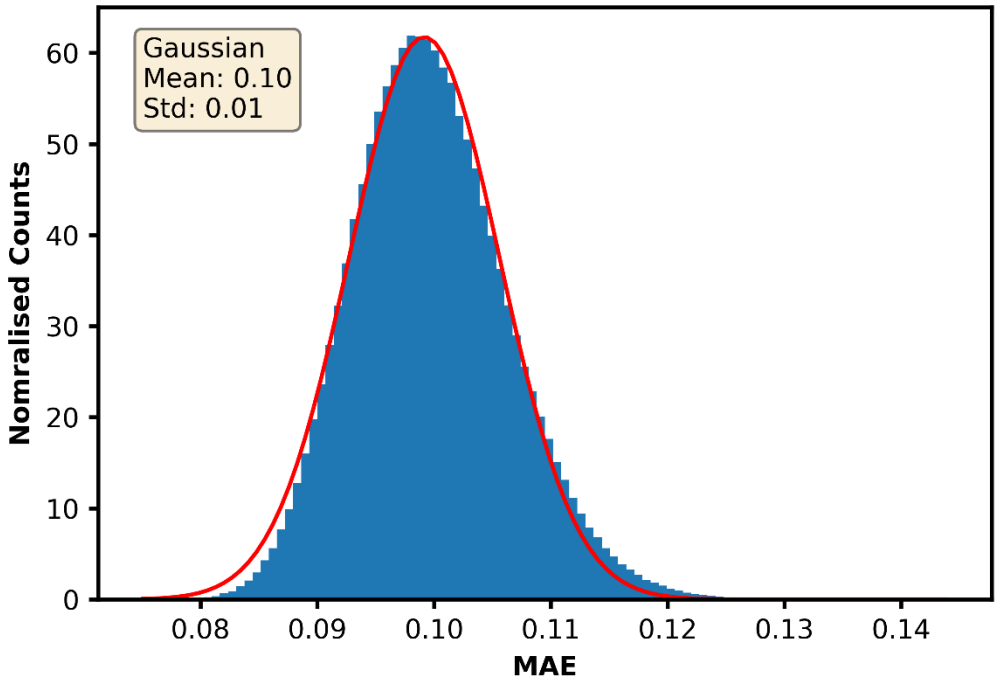



*Figure 4.8*

*Distribution of MAE values when performing background analysis using a window size of 100 and a step size of 10. The resultant distribution is positively skewed. However, a Gaussian fit is applied with a mean of 0.10 and standard deviation of 0.01.*

distribution such as a gamma distribution, or another is to fit distributions to both the validation and training sets separately before summing them and normalising.

However, for the purpose of populating a ROC curve, the selection of sensitivities does not require specific percentiles for trialling the AE algorithm. The percentile approach for the sensitivity thresholds was chosen as it can provide insight into the performance of statistical confidence boundaries. These sensitivity thresholds do not need to be chosen by taking percentile thresholds from a background distribution as the only requirement for populating a ROC curve is for the chosen sensitivities to span the range of values within the background. Given that the background distribution for this AE method is skewed and that the use of percentiles is not required, analysis continued with the fit shown in Figure 4.8 to generate the algorithm sensitivities. The sensitivities chosen are outlined in Table 4.2. Now, it would be inappropriate to refer to these sensitivities as percentile thresholds. Therefore, they are now referred to as sensitivity thresholds.

Following background characterisation, the different sensitivity thresholds were tested on the different SNR datasets that were analysed previously. Here the MAE losses for reconstructing windows from the event-containing datasets were calculated (Figure A1.7, Figure A1.8). These losses had the previous sensitivity thresholds applied to flag anomalous windows where anomalies constituted windows with MAEs that exceeded the thresholds. After event detection, analysis of the data was carried out in the same way as previously. Several ROC curves were produced and their AUCs calculated. These are summarised in Figure 4.9 where a slight improvement in event detection performance, when compared to the ZCR detector, can be seen. Whilst the ZCR algorithm was able to achieve an AUC score of 0.86 when

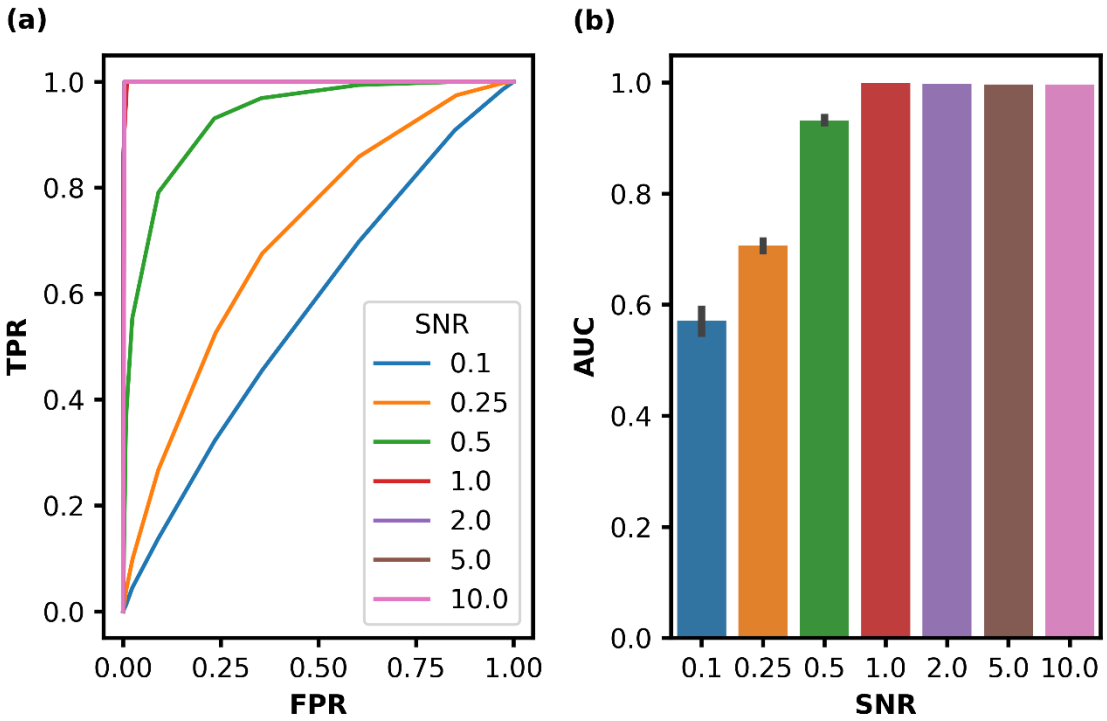



*Figure 4.9*

*Anomaly detection results via the convolutional AE method. Shown are the ROC curves for each SNR, (a), and the corresponding AUC scores, (b).*

analysing data with an SNR of 0.5, the convolutional AE technique was able to achieve a score of 0.93. Improvements were also seen when considering the datasets with SNRs of 0.25 and 0.10. Where the ZCR technique was able to score 0.62 and 0.52, respectively, the convolutional AE scored 0.71 and 0.57. This makes the convolutional AE the optimal technique for anomaly detection so far.

### 4.2.5. Moving Average

After testing the ZCR detector and convolutional AE techniques, an explanation as to why they were superior was desired. One of the potential reasons for the improvement was speculated to be the implementation of the moving window. The use of the moving window means a larger sample size is used for each position in a trace such that a narrower distribution could

result. This is explained by the standard error of means which is described in Section 4.2.6. With this logic, the idea of combining the moving window with the amplitude thresholding technique was created. This technique is already an established strategy for smoothing noisy

*Table 4.3*

*Summary of chosen percentiles used for thresholding and their corresponding average values.*

| Percentile | Value |
|---|---|
| **0.00 %** | -0.4655 |
| **25.00 %** | -0.0673 |
| **50.00 %** | 0.0000 |
| **75.00 %** | 0.0673 |
| **90.00 %** | 0.1279 |
| **95.00 %** | 0.1641 |
| **99.00 %** | 0.2322 |
| **99.90 %** | 0.3083 |
| **99.99 %** | 0.3841 |
| **100.00 %** | 0.5038 |

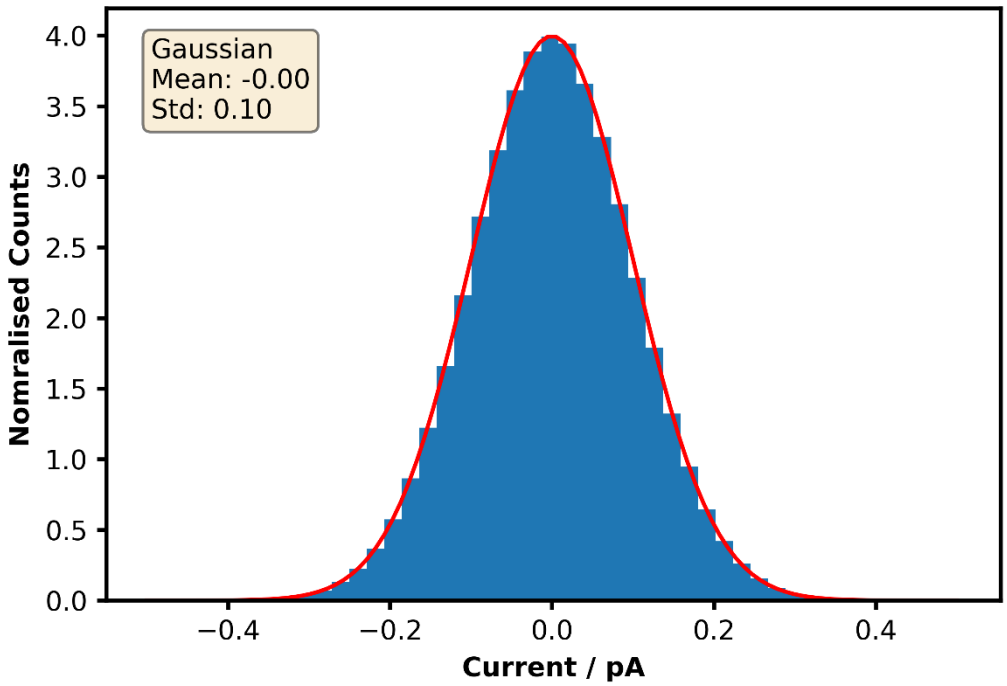



*Figure 4.10*

*Distribution of average current values when performing background analysis using a window size of 100 and a step size of 10. The resultant distribution is Gaussian with a mean of 0.00 and standard deviation of 0.10.*

data known as the moving average.[17] The moving average technique has also found use in forecasting challenges.[18,19] Therefore, given the previous applications of this technique to time series challenges, the moving average was implemented in this anomaly detection problem set.

The same first step of splitting the background dataset into a matrix of possible window positions was carried out. Each window was then used to calculate the window's average. These window averages were then analysed in the same manner as the ZCR and convolutional AE techniques. Namely, the window averages were plotted on a histogram before a Gaussian fit was applied (Figure 4.10). Using the Gaussian fit, percentile values were extracted. These

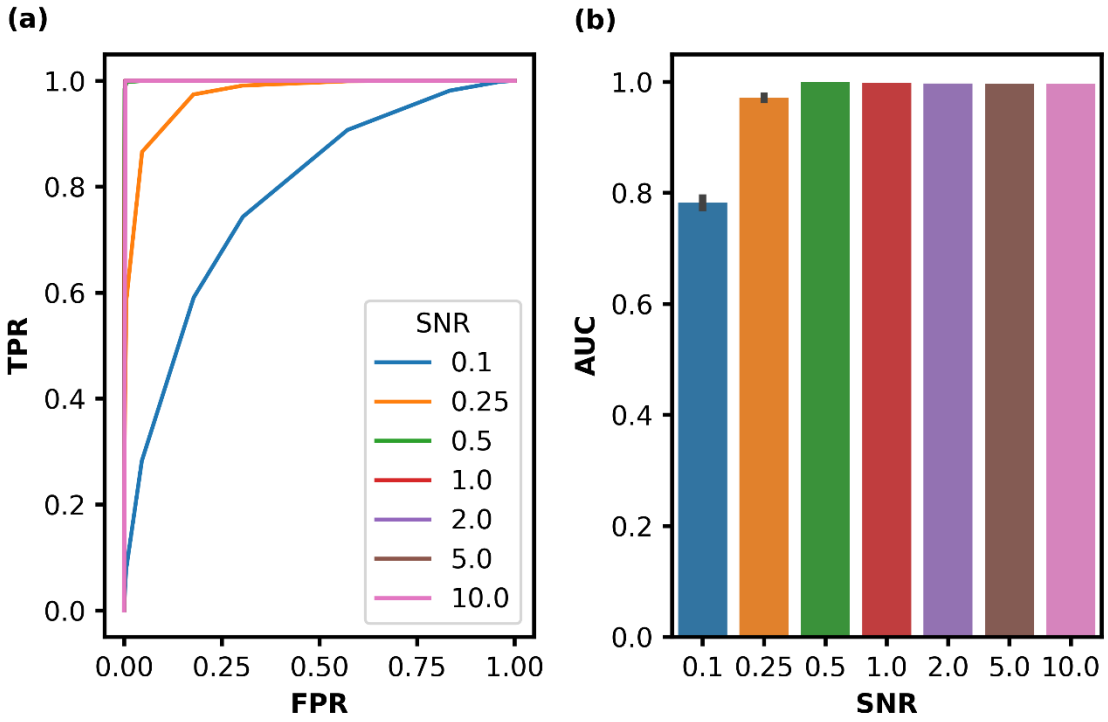



*Figure 4.11*

*Anomaly detection results via the moving average method. Shown are the ROC curves for each SNR, (a), and the corresponding AUC scores, (b).*

are summarised in Table 4.3. The same window size and step size, of 100 and 10 were implemented for this procedure, respectively.

Once again, traces were converted into traces of window position values (Figure A1.9, Figure A1.10), before the thresholds were used to flag up anomalous windows in the event-containing data. Windows with an average value greater than the used threshold were labelled as outliers. Once all anomalies were flagged the same ROC curve and AUC procedure was then carried out (Figure 4.11). Here it can be seen that there is a drastic improvement than all previous techniques. When comparing this technique to the convolutional AE, the event detection performance of the moving average technique was able to achieve higher AUC scores even with the data with low SNRs. Whilst the convolutional AE achieved scores of 0.93, 0.71, and 0.57 for the SNRs of 0.5, 0.25 and 0.1, respectively, the moving average technique achieved scores of 1.00, 0.97, and 0.78.

### 4.2.6. Parameter Optimization

To ascertain the effects that the window size and step size parameters may have on the AUC performance of the three window-based anomaly detection methods, additional parameter

*Table 4.4*

*Summary of parameters to be tested. Each combination of window size and step size multiplier are used to calculate the shown step sizes.*

| | | **Window Size** | | | | |
|---|---|---|---|---|---|---|
| | | 100 | 500 | 1000 | 1500 | 2000 |
| **Step Size Multiplier** | 0.1 | 10 | 50 | 100 | 150 | 200 |
| | 0.5 | 50 | 250 | 500 | 750 | 1000 |
| | 1.0 | 100 | 500 | 1000 | 1500 | 2000 |

optimisation was carried out. The above procedures were repeated with every combination of the window size and step size parameters outlined in Table 4.4. To summarise, there are 5 different values for window size, and 3 for the step size. Given that there are 7 different SNRs that are explored with each class containing 10 traces, this means that a total of 1050 event detection runs were performed per technique to analyse their performances. With 3 different methods using the window-based approach, this gives a total of 3150 algorithm runs with each producing an AUC score.

Figure 4.12 shows how the AUC score for each technique changes with relation to the step size whilst keeping the window size constant. This is shown for four of the trialled SNRs. Overall, it can be seen that, for SNRs below 1.0, better anomaly detection results can be seen when utilising a smaller step size. However, this improvement lessens the higher the SNR of the anomalies are. When considering the performance of the moving average algorithm, it is

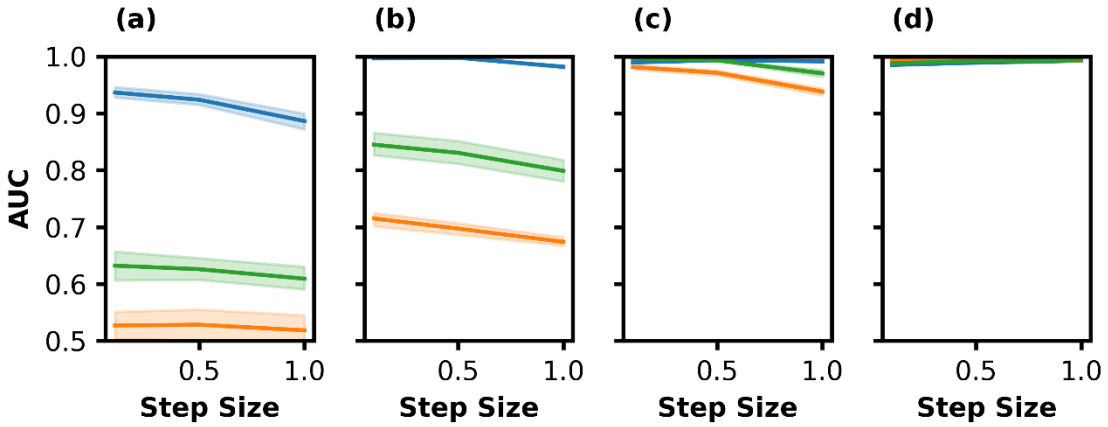



*Figure 4.12*

*Comparison of the effect of different step sizes on the AUC performance. The effect is shown for each detection method: ZCR (orange), convolutional AE (green), and moving average (blue). In addition, this comparison is shown for four different event SNRs. These are the SNRs of 0.10, 0.25, 0.50, and 1.00 for (a)-(d), respectively.*

evident that by increasing the step size from 0.1 to 1.0 times the window size, the performance drops by 0.05 for the 0.1 SNR dataset. This effect is lessened when considering the 0.5 SNR dataset. Here there are no considerable changes in performance.

Figure 4.13 shows how the performance of each technique changes with regards to the window size. Firstly, when considering the performance of the moving average algorithm, the best window sizes are 1000, 500, 100, and 100 for the SNRs 0.1, 0.25, 0.5, and 1.0, respectively. For the SNRs of 0.1 and 0.25, it can be seen that a window size greater than or less than the optimum results in poorer performance with the fall off being greater when decreasing window size. A similar behaviour is observed when considering the ZCR algorithm when changing the window size. In this instance the best window sizes are 1000, 1000, 1000, and 100 for the SNRs 0.1, 0.25, 0.5, and 1.0, respectively.

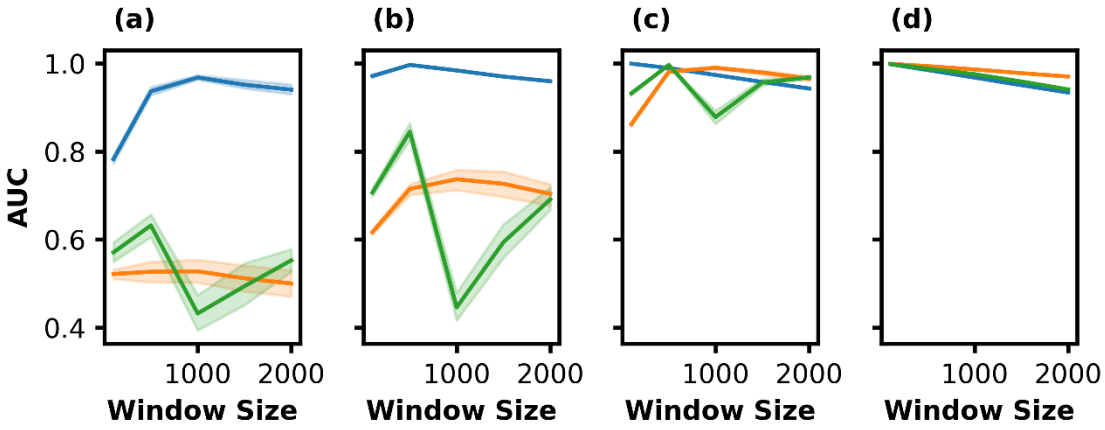



*Figure 4.13*

*Comparison of the effect of different window sizes on the AUC performance. The effect is shown for each detection method: ZCR (orange), convolutional AE (green), and moving average (blue). In addition, this comparison is shown for four different event SNRs. These are the SNRs of 0.10, 0.25, 0.50, and 1.00 for (a)-(d), respectively.*

For both the moving average and the ZCR detector the optimum AUC is never achieved with a window size greater than 1000 data points. This can be explained by the average event duration. The mean event duration used during the simulation was 1000 data points. When considering a window size greater than the event durations, there is no orientation of the window over the events such that no background points are captured. Because of this, as the window size becomes very large compared to the event durations, the calculated metrics will approach the background values. This makes anomalous events less likely to be detected.

At the other extreme, the decreasing performance of detection when reducing the window size could be explained by statistics. More precisely, this behaviour can be explained by the standard error of means. In statistics it is widely known that sampling larger sample sizes from a distribution will result in a smaller standard deviation of means.[20] This can be compared to both the moving average and ZCR detectors where the window size chosen acts as the sample size in terms of standard error. Figure 4.14 shows how the background analyses for these techniques can produce distributions with different widths based on the chosen window size. From this it can be speculated that having a larger window size will also result in a tighter distribution for anomalous points. Because of these tightening distributions, there will exist a window size where the distributions between events and background do not overlap and thus increase their separability.

The behaviour of the convolutional AE is more difficult to interpret. Here, the performance of the convolutional AE is at its best for a window size of 500 when applied to the 0.1, 0.25, and 0.5 SNR datasets. In addition to this, the performance drops to a minimum when using a window size of 1000 followed by increasing performance as the window size increases. This

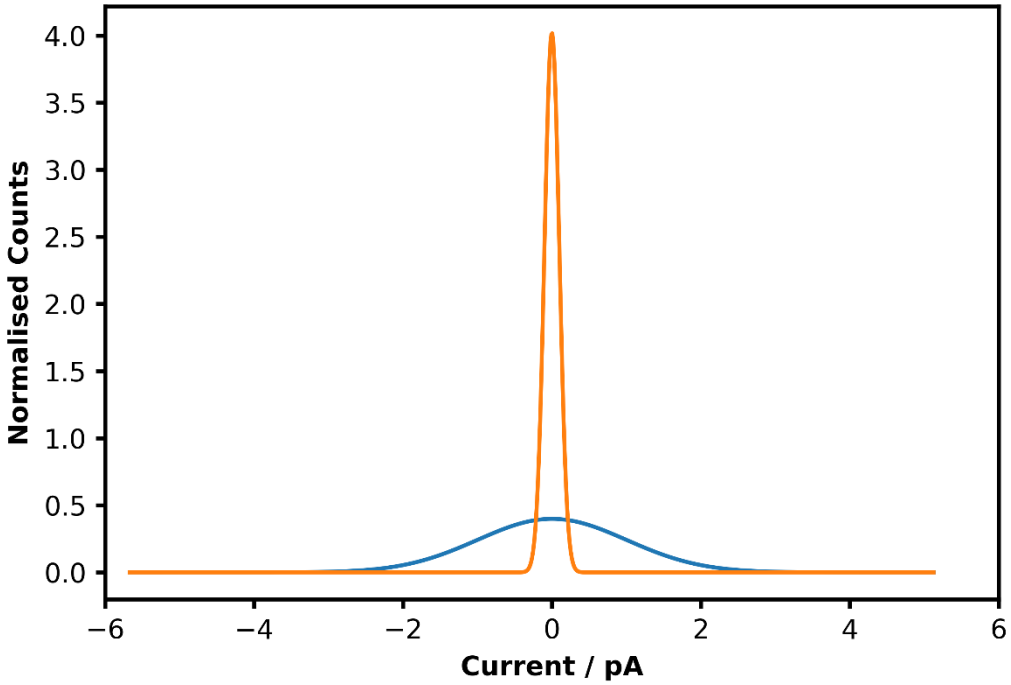



*Figure 4.14*

*Demonstration of the effect of standard error of means with the moving average technique. The distribution of background values (blue) is compared to the corresponding distribution of means with a window size of 100 (orange).*

behaviour cannot be explained by the previous logic associated with the moving average and ZCR algorithms. These results could be due to the non-Gaussian distributions associated with the MAE losses, but it is likely that another reason causes this behaviour.

To make the convolutional AE work, the input size of the neural network needs to be fixed to a certain number of features. Because of this limitation, when using a different window size to one that has already been tested, the input layer to the neural network needs to be changed. As the fundamental architecture of the neural networks is being altered, the new network will require training again. Therefore, each window size that was tested, was tested with an independently trained AE with similar architecture. The architectures of each network are

summarised in Appendix 2. Because each AE is trained independently, this makes their performances less comparable as their weight initialisation and updating will have happened stochastically.[21] Therefore, the observed fluctuations in the AUC score are likely attributed to the retraining of networks.

In summary, the best parameters for anomaly detection for the moving average and ZCR detectors can be determined by considering the standard error of means and the duration of the events to be found. Consideration of these leads to the conclusion that the best window size is one that is as large as possible up to the limit of the event durations. In addition to this, the best step size parameter would be a value that is as small as possible. This allows all possible window positions to be analysed such that the change points can be more precisely located. With regards to the performance of the convolutional AE, the best window size depends on the accuracies achieved during the training process of the neural network.

With the optimum parameters having been ascertained, a fair comparison between the techniques could be made. Figure 4.15 shows the AUC performance of all three techniques when applied to all 7 of the different SNR datasets whilst utilising their optimum parameters. Before parameter optimisation, a window size of 100 with a step size of 10 was used. Now it is known that the best parameters are a window size of 1000 with a step size of 100 for the moving average and ZCR techniques, whilst the AE performs best with parameters of 500 and 50 for window and step sizes, respectively. With the optimum parameters selected it can be seen that all three techniques perform better than previously. Most impressively the score of the moving average technique has grown from 0.78 up to an AUC score of 0.97 for the SNR of

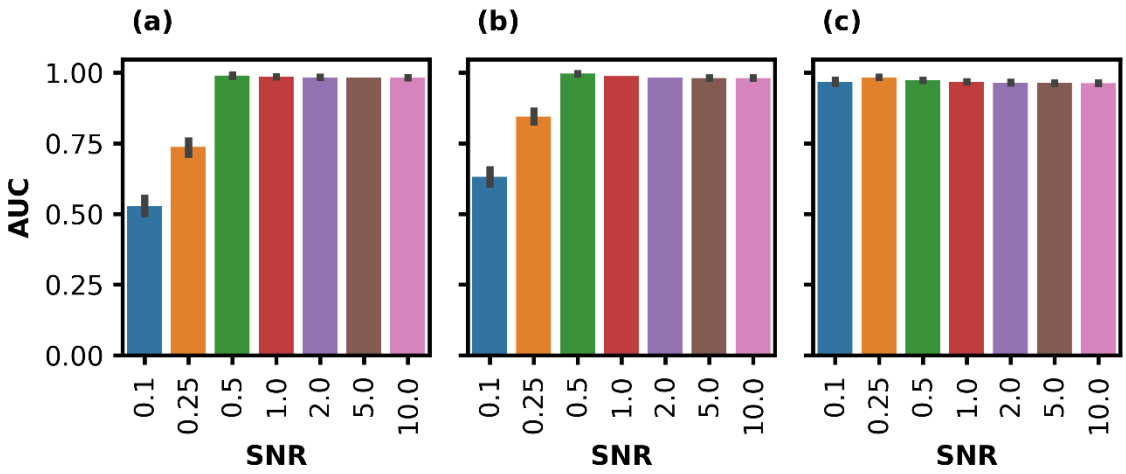



*Figure 4.15*

*Anomaly detection results of the ZCR, (a), convolutional AE, (b), and moving average, (c), techniques when using their optimum parameters.*

0.1. Therefore, the best technique is still attributed to the moving average with the AE coming in second position followed closely by the ZCR algorithm.

### 4.2.7. Event duration

To further explore the dependence of the detection performance on window size, a second set of simulated data was produced. The same SNRs were used to generate traces but in this case the duration of the events was greatly reduced. Here the average event duration was set to 10 with a standard deviation of 3. Given that the event duration is much smaller, the window sizes for the moving average and ZCR algorithms will be limited. When setting a window size greater than 10, both of the techniques will be susceptible to the performance decreases shown previously.

Using what was learnt from the parameter optimisation task, the shorter event dataset was analysed using a window size of 10 with a step size of 1 for the moving average and ZCR techniques. For the convolutional AE, the best performance was achieved with a window size

of 20 and a step size of 20 (Figure A2.6). The ZCR, moving average, and convolutional AE algorithms were trialled on all SNRs of the shorter event duration datasets. Here the same sensitivity/percentile thresholds were applied to yield ROC curves with AUC scores. The results of this are summarised in Figure 4.16. Here it can be seen that the performance of these event detection strategies is less when compared to the previous performance with longer events (Figure 4.15). For instance, previously the moving average technique was able to achieve an AUC score of 0.97 on an SNR of 0.1. Now, however, with the shorter event durations, the performance was only able to achieve a score of 0.57. The algorithm ranking remains the same, with the best being the moving average technique and the worst being the ZCR detector.

The overall drop in performance, when comparing the shorter event detection to the longer events supports the previous assertion that the performance depends on the standard error associated with the window size. The drop in the AEs performance suggest that the above

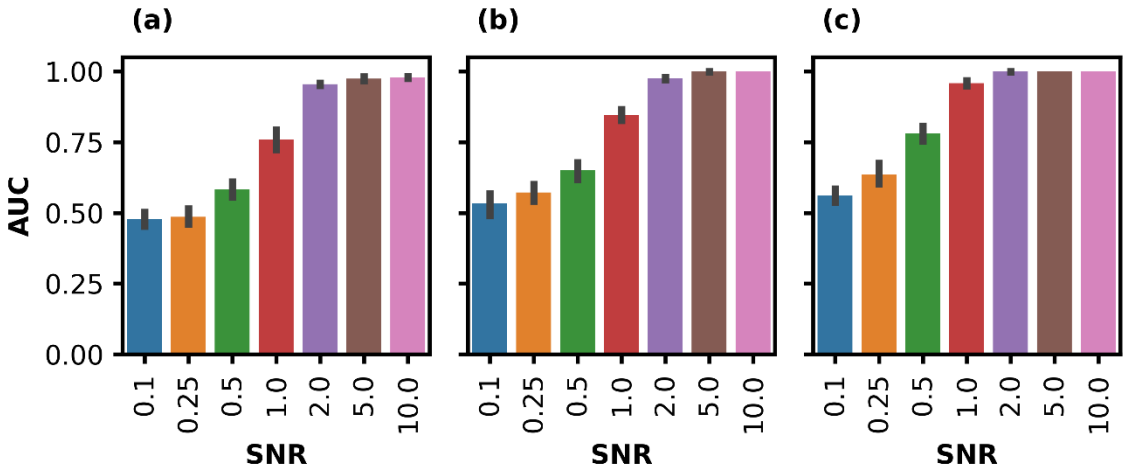



*Figure 4.16*

*Anomaly detection results of the ZCR, (a), convolutional AE, (b), and moving average, (c), techniques when using their optimum parameters on the detection of short events.*

reasoning will also affect its performance to a certain extent. However, with the AE, there is also the effect of randomness during training that could result in poorer performance.

### 4.2.8. Experimental Data

Lastly, the moving average technique was applied to a dataset consisting of sampled experimental data. A trace was provided by another member of the Albrecht research group that had been sampled by applying the RPS technique to a solution containing strands of DNA. Following the works of Loh et al.[22], a procedure was established to reliably detect DNA translocation events. Here 300 pM of 4 kilobase double stranded DNA molecules were prepared in a solution of 4M lithium chloride with 10% Tris-EDTA. To this solution, electrodes and a nanopore with a pore diameter of approximately 15nm were immersed. Voltage was then applied, and the corresponding current was measured. This system was sampled over 10 seconds at a rate of 1 MHz to yield a trace of 10,000,000 data points (Figure 4.17). Initial inspection shows a sloping feature that represents the electronics warming up. Therefore, the initial 600,000 data points were discarded. From here, the remaining 9,400,000 data points were analysed using the moving average algorithm.

Typically, this algorithm would start with the analysis of background data to acquire the appropriate thresholds to accurately determine what is outside the normal range of window averages. However, due to the nature of this experiment, it is not possible to sample the solution without the presence of the DNA structure prior to the measurement of DNA translocations. If another solution was used to measure the background, then these measurements may not be appropriate due to any small changes in the LiCl medium or any changes in the electronic noise. Because of this, the background analysis was carried out on

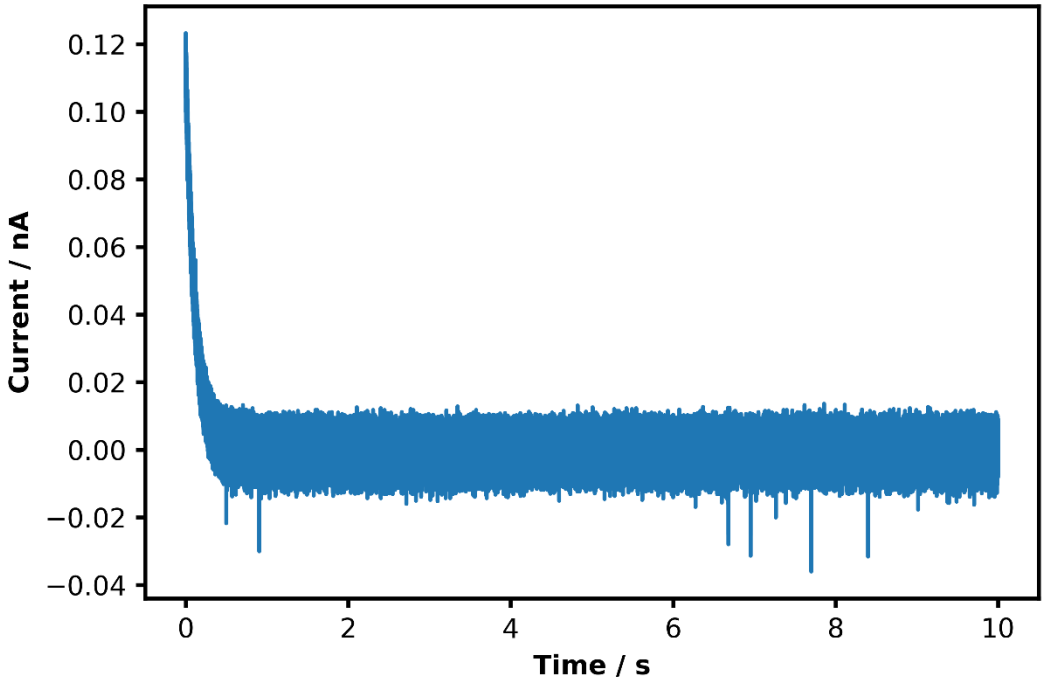



*Figure 4.17*

*Experimental trace sampled from the translocation of DNA. The electrical current was measured at a frequency of 10MHz over the span of 10 seconds.*

the trace that contains translocation events. To do this an assumption was made with regards to the translocation events. This assumes that the translocation events occur relatively infrequently such that the resultant histogram of window position values is dominated by the background behaviour. Continuing with this assumption, the background analysis was carried out in this manner. The parameters chosen were window sizes of 1000, 500, and 100 with step sizes being a 0.1 multiple of the used window size. The reason for trialling different window sizes is that there is no prior knowledge of how long the translocation will last. The translocation event could take place over 1ms or even 0.1ms. Because of this the above parameters were selected. Figure 4.18 shows the background histograms that were obtained from each of the parameter choices. Looking at these distributions it can be interpreted that

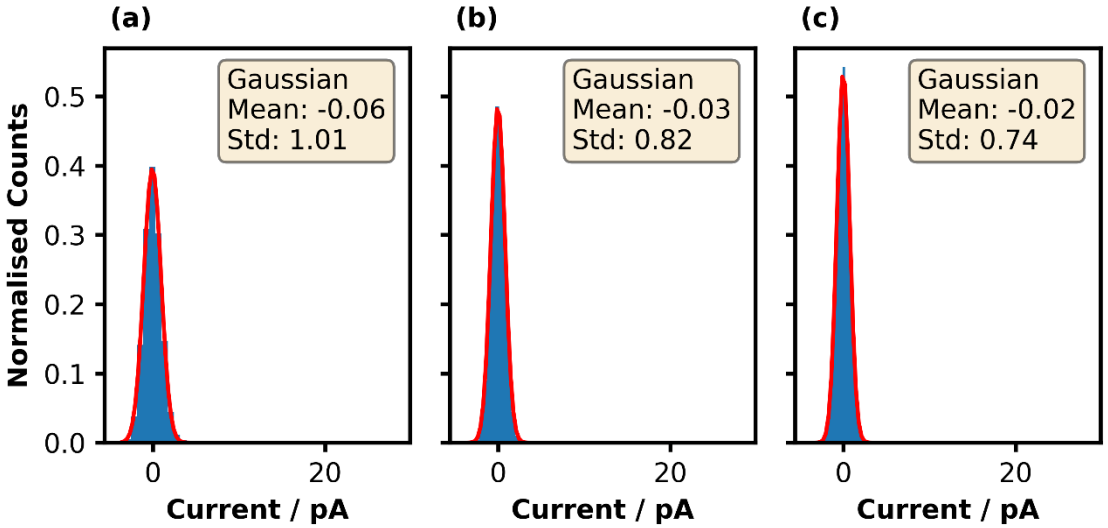



*Figure 4.18*

*Distributions of average current values when performing background analysis using window sizes of 1000, (a), 500, (b), and 100, (c) on the experimental data trace.*

the previous assumption is true. This is supported by how well each of the Gaussian fits performed. There appears to be no skew in the resultant histograms as consequence of the presence of translocation events. Therefore, the event searching algorithms continued to the next step of determining a threshold.

The percentile of 99.99% was chosen to extract threshold values from the background distributions. Using this value means that there will be events detected that are likely to correspond to false positive event detection. This will require the events returned to be manually inspected to see if there are any noticeable differences in appearance between events that might allude to the presence of false positives. The event searches were carried out using the above thresholds and the results of which are summarised in Figure 4.19. One of the first observations that can be made is that there appears to be a duration offset between the three different window sizes. For instance, when considering the five tallest

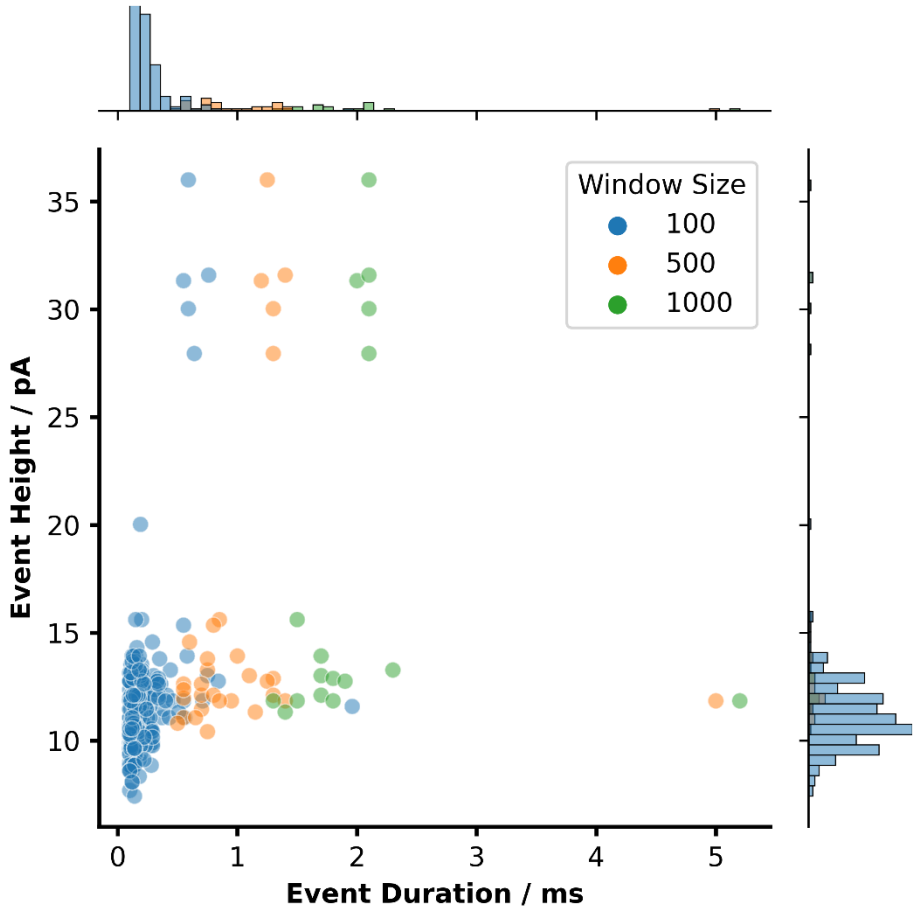



*Figure 4.19*

*Scatter plot of event heights vs durations for events found using the moving average technique on experimental data. Results are shown for each of the three trialled window sizes: 100, (blue), 500, (orange), and 1000 (green).*

events, there is as much as a 1ms difference between events found using the 1000-point window size versus the 500-point window size. This is a result of the chosen algorithm parameters causing differences in time resolution. Because of this duration offset between anomaly detection runs, there will be duplicate events when comparing the different runs. As such, the only comparison that can be made between the runs with different window sizes, is that of the number of anomalies found. The number of events found are summarised in Table 4.5. Based on prior assumptions related to the standard error, these experimental results

*Table 4.5*

*Number of detected events when utilising the corresponding window size parameter with the moving average technique on experimental data.*

| Window Size | Number of Events |
|---|---|
| **1000** | 17 |
| **500** | 31 |
| **100** | 309 |

make sense as the run with the smallest window size returned the greatest number of events, and the run with the largest window size returned the fewest. The additional events found with a smaller window size could be either attributed to the detection of more false positive events, but it could also be events whose duration better matched the smaller window size.

Overall, there are three distinct clusters to the events found when considering the window size of 500. The cluster of five tall events, the single event with a duration of 5ms, and the main cluster of events with durations between 0 and 2ms and heights between 5 and 20pA. When considering the cluster of five tall events and the one long event, it would be a fair assumption to state that these correspond to true-positive detected events. For the five tall events, their magnitude is so large that they are statistically independent with regards to the background distribution of current values (Figure 4.18). In addition, it is likely that the event with duration around 5ms is also a true positive due to the event being distinguishable from the other events in terms of duration. For both of these groups of events, the labelling of true positives is supported by the result that they fall into clusters separate from the majority of the other events. The final cluster with shorter height and duration is likely to contain some true positives but mostly false positives. As the event heights and durations are short, the detection is limited by the window size. Within this cluster, it is more likely for events to be false positives as their heights are closer to the values of the background. Overall to be able

to distinguish between events within this ambiguous cluster, additional cluster analysis and DR would be appropriate to further study whether the false positives and true positives within this cluster are separable. Example events from each cluster are presented in Figure A1.11.

## 4.3. Conclusion

In summary, for events of this nature, those which manifest as square wave pulses of increased magnitude, the best technique for detecting anomalies is the moving average technique. This is followed by the convolutional AE and closely by the ZCR technique. This is apparent by the fact that the moving average technique achieved the highest ROC AUC score of 0.97 for detecting events with an SNR of 0.1.

However, this result requires that the events that are being detected have a sufficient number of data points sampled throughout their duration. Indeed, it was shown that when applying these anomaly detection techniques to events with an average duration of 10 datapoints, the overall performance of all techniques decreased. The highest score for detecting events with an SNR of 0.1, which was still achieved by the moving average technique, was 0.56. This finding, therefore, implies that effective event detection requires a heightened sampling rate such that events contain hundreds of data points instead of tens.

In support of this, anomaly detection using moving average technique was trialled on an experimental dataset that was sampled at 1MHz. Here, where events typically last on the order of 1ms, the sampling rate allows for events of around 1000 datapoints. The results of this analysis showed that various events could be pulled out with different clusters of event heights and durations. To better understand how effective the anomaly detection techniques developed here are, additional studies on the application to experimental data are required.

In addition, further steps in clustering the found events would be required to better separate potential false positives from the true events.

Lastly, whilst this case study showed that statistical outlier detection was superior to DL methods, it is worth noting that other cases will likely result in different results. For instance, the moving average technique may not perform as well as the convolutional AE if the events did not involve purely a magnitude shift. If anomalies were to be changes in patterns, the convolutional AE may likely outperform the moving average.[23] Additionally, if anomalies manifested as changes in noise RMS, the ZCR or other value crossing rate detectors could also outperform the moving average technique.

## 4.4. Methods

Construction of the simplified I(t) simulation with events of user defined SNRs was written in the Python programming language. Additionally, all implementations of anomaly detection algorithms were written in this language as well. This project also utilised the Anaconda virtual environment management utility. Therefore, the same packages and version dependency as used in Chapter 3 were implemented here.

## 4.5. References


1 Mingyan Teng, in *2010 IEEE International Conference on Progress in Informatics and Computing*, IEEE, 2010, vol. 1, pp. 603–608.

2 H.-S. Wu, in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, 2016, pp. 426–431.

3 M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, *IEEE Access*, 2019, **7**, 1991–2005.

4 A. A. Cook, G. Misirli and Z. Fan, *IEEE Internet Things J*, 2020, **7**, 6481–6494.

5 A. Blázquez-García, A. Conde, U. Mori and J. A. Lozano, *ACM Comput Surv*, 2022, **54**, 1–33.

6 W. Haiss, R. J. Nichols, H. van Zalinge, S. J. Higgins, D. Bethell and D. J. Schiffrin, *Physical Chemistry Chemical Physics*, 2004, **6**, 4330.

7 H. Bayley and C. R. Martin, *Chem Rev*, 2000, **100**, 2575–2594.

8 A. Swift, R. Heale and A. Twycross, *Evidence Based Nursing*, 2020, **23**, 2–4.

9 D. Chicco and G. Jurman, *BioData Min*, 2023, **16**, 4.

10 T. Fawcett, *Pattern Recognit Lett*, 2006, **27**, 861–874.

11 Jin Huang and C. X. Ling, *IEEE Trans Knowl Data Eng*, 2005, **17**, 299–310.

12 E. Parzen, *The Annals of Mathematical Statistics*, 1962, **33**, 1065–1076.

13 M. P. Deisenroth, A. A. Faisal and C. S. Ong, *Mathematics for Machine Learning*, 2020.

14 M. Sakurada and T. Yairi, in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, ACM, New York, NY, USA, 2014, vol. 02-December-2014, pp. 4–11.

15 C. Zhou and R. C. Paffenroth, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, 2017, vol. Part F129685, pp. 665–674.

16 Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, in *2018 Wireless Telecommunications Symposium (WTS)*, IEEE, 2018, vol. 2018-April, pp. 1–5.

17 N. Vandewalle, M. Ausloos and P. Boveroux, *Physica A: Statistical Mechanics and its Applications*, 1999, **269**, 170–176.

18 S. Hansun, in *2013 Conference on New Media Studies (CoNMedia)*, IEEE, 2013, pp. 1–4.

19 D. J. Robb and E. A. Silver, *Journal of the Operational Research Society*, 2002, **53**, 1281–1285.

20 D. G. Altman and J. M. Bland, *BMJ*, 2005, **331**, 903.

21 Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436–444.

22 A. Y. Y. Loh, C. H. Burgess, D. A. Tanase, G. Ferrari, M. A. McLachlan, A. E. G. Cass and T. Albrecht, *Anal Chem*, 2018, **90**, 14063–14071.

23 T. Finke, M. Krämer, A. Morandini, A. Mück and I. Oleksiyuk, *Journal of High Energy Physics*, 2021, **2021**, 161.

# Chapter 5

## Unsupervised Classification in Single-Molecule Science and Electrochemistry

Contents

## 5.1. Introduction

The ability to detect subpopulations within datasets is typically conducted by analysis of statistical distributions. Previously in Sections 1.3 and 2.2.2.1, the complexities of working with high-dimensional datasets were discussed. Statistical analysis is limited by these intricacies due to the large number of interrelated variables, and the various phenomena that coincide with the CoD. Due to these difficulties, alternative techniques are desired to make the task of finding subpopulations in high-dimensional bases more easily achievable. Previously, the ML strategy of DR was introduced to remedy these issues.

Therefore, following the above notion, and the previous examples of DR in STM-BJ analyses discussed in Section 1.1.1, this work will incorporate DR and clustering analysis within the scope of STM-BJ analysis. The purpose of this is to explore the potential that DR has when used for the extraction of subpopulations within complex datasets. The aims here are to alleviate the challenges of working with BJ datasets, as well as to showcase the versatility of these techniques by applying DR in an electrochemical sensing setting.

## 5.2. Scanning Tunnelling Microscopy Break Junctions

BJ analysis, like with other SM strategies, requires a large ensemble of measurements to provide accurate insights into molecular properties.[1] Each BJ measurement produces traces like the one presented in Figure 1.1. Because of this, the resultant datasets are typically highly dimensional and complex. One of the challenges in these complex datasets is how to overcome inherent noise to extract the desired information. When measuring thousands of BJ traces, it is very common for a subset of these traces to possess features that are a result of a “poor pull”. The reasons for these “poor” traces were introduced in Section 1.1 where

error contribution in SM science and STM were discussed. Because of the presence of “poor” traces, measured datasets may require cleaning before analysing molecular properties. To do this, those traces where molecular characteristics are hidden need to be separated. However, manually filtering through traces can be a difficult and time-consuming task. Instead, an automated process involving DR and clustering could be an easier tool for the cleaning and separation of traces within a set.

In addition to data filtering, automated subpopulation detection using DR and clustering could also provide a useful method for more general clustering of high dimensional behaviours. For instance, in Section 1.1.1, the case of a bimodal conductance distribution was presented as an example of the limitations of the traditional statistical approach to BJ analysis. Here DR could also have an application to separate these different populations of subtle complex shape differences and better elucidate the cause of resultant bimodal distributions.

### 5.2.1. Conductance Measurements of Compounds

To explore the aforementioned use that DR may provide to the field of BJ analysis, a selection of molecules was chosen. In total three molecules were selected to be experimentally measured using the STM-BJ technique. These molecules include two well studied molecules known as OPE3 thioacetate, and 4,4’-bipyridine. These will be referred to as molecules 1 and 2, respectively. In addition to this, a third, novel, molecule was studied which will be referred to as molecule 3. This molecule is formally introduced in Section 5.2.4. Molecules 1 and 2 were used to provide a suitable testbed for exploring the effectiveness of DR techniques in the cleaning and sub-population analysis tasks. Using what is learnt from these molecules, the DR

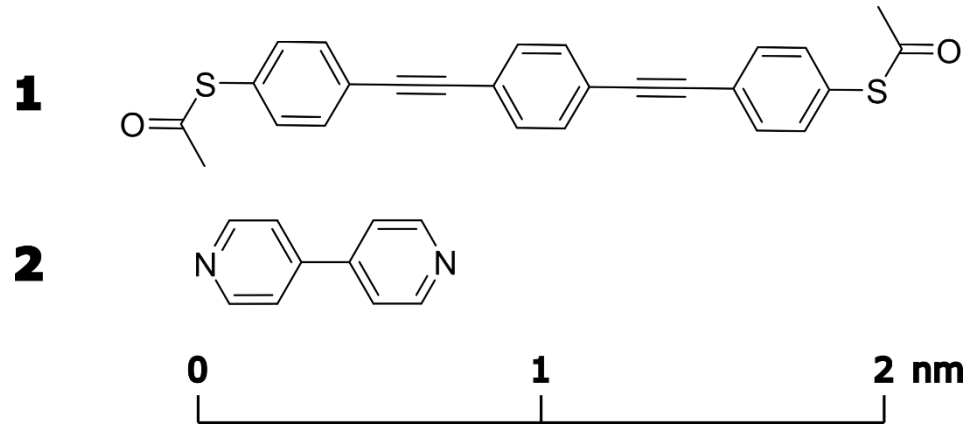



*Figure 5.1*

*Chemical structures and length comparison of molecules 1 and 2. The distance between the anchoring groups within each molecule are modelled to be 2.01nm and 0.72nm for molecules 1 and 2, respectively.*

strategies were then applied to molecule 3 to aid in understanding the behaviour of a novel molecule.

Initially, molecules 1 and 2 were measured. The structures of each of these molecules are demonstrated in Figure 5.1. Following the procedures outlined in Section 5.2.6, BJ traces for these two molecules were sampled. Traditionally, BJ results, as previously discussed, are presented using histogram analysis techniques. A large amount of BJ traces akin to the one shown in Figure 1.1 are collected and analysed together to elucidate probabilistic behaviours. These analyses include 1D histograms of conductance behaviour and molecular length, as well as a 2D histogram for characterising the conductance-distance behaviour. The raw measurements are presented in this fashion in Figure 5.2 for molecule 1. As a side note, the dataset for molecule 1 subsequently features in published literature.[2] This histogram analysis was also carried out for molecule 2 (Figure A1.12). It is also worth noting that the trace demonstrated in Figure 1.1 was taken from molecule 2's dataset. Therefore, the other traces present in this dataset will be very similar to the shown trace. When considering the 2D histogram of conductance vs displacement (Figure 5.2a), undesirable traces are present. Most

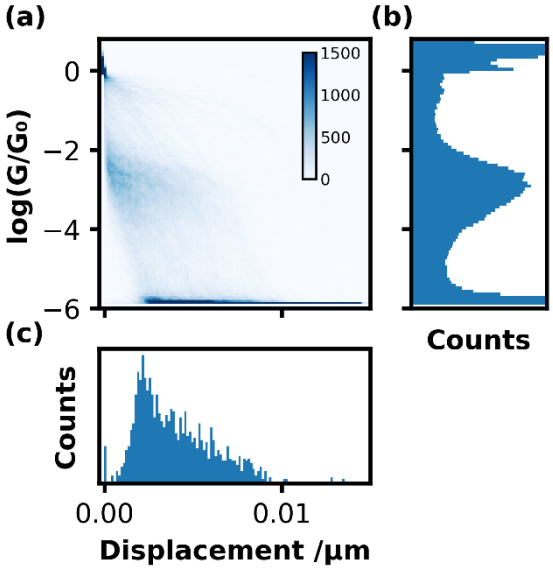



*Figure 5.2*

*Histogram analysis of molecule 1 using uncleaned data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. A total of 1529 traces were sampled.*

notable are the points present with a conductance greater than -2 log($G/G_0$) after the initial break point at 0nm. The existence of points within this region likely results from BJ traces that were unable to break quickly during tip withdrawal. This can be due to a strong coupling of the molecule with the electrode surfaces,[3] or due to the presence of contaminants bridging the electrodes. Another undesirable feature that is present is the large range of plateau lengths. When considering the molecular structure of molecule 1, the expected plateau length would be around 2nm. However, in this case this value ranges from 0 up to 10nm. These results are also likely due to the same reasons as the high conductance values from before.

### 5.2.2. Data Cleaning and Selection

To improve the quality of these results, a traditional cleaning method was then applied to both result sets. This cleaning method, as outlined in Section 5.2.6, involves selecting thresholds

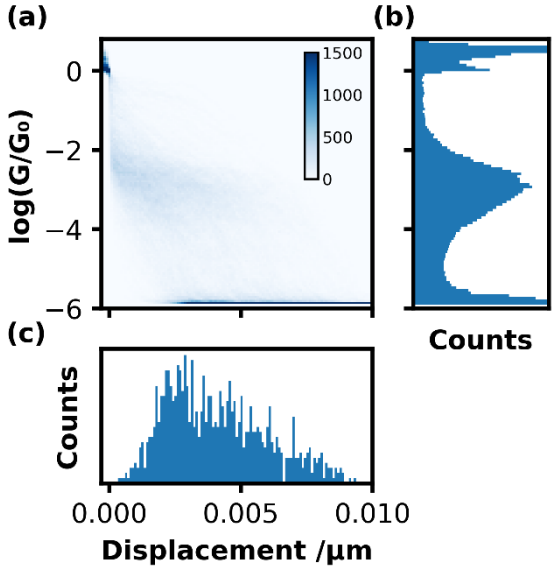



*Figure 5.3*

*Histogram analysis of molecule 1 using filtered data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. After this filtering, 946 traces remain.*

such that prohibited traces are removed from the results. These thresholds include minimum and maximum limits on trace properties such as the average conductance of each trace, the plateau length of the trace, and the noise level in the traces tail. The values used for each threshold for each molecule are summarised in Table 5.3 in Section 5.2.6. Once this filter was applied, the clean datasets were analysed using the same histogram process (Figure 5.3, Figure A1.13). After filtering the traces of molecule 1, it can be seen that the offending traces with high conductance values have been removed. This, in addition to making the 2D histogram cleaner, also improves the cleanliness of the 1D conductance histogram. However, the trace plateau lengths could not be pruned effectively. Whilst the most extreme cases have been removed, the overall range of values is still too broad. Further pruning via value thresholding is undesirable as this would result in a plateau length histogram with sharp cut-

offs at the value where the pruning would have been applied. This would impact the quality of any subsequent analyses of this property's distribution.

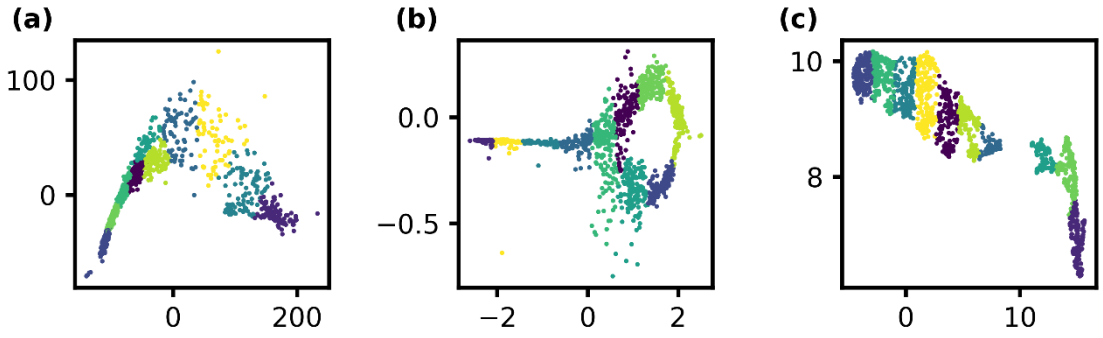



*Figure 5.4*

2-dimensional embeddings of molecule 1's uncleaned BJ data. Embeddings were produced using PCA, (a), t-SNE, (b), and UMAP, (c). Points are separated into ten clusters using the k-means clustering algorithm and are coloured accordingly.

Due to the aforementioned limitations with this cleaning method, a separate method was used to clean the raw BJ results. This second method utilised DR techniques to compress each individual BJ trace from a 2000-point trace into a 2-dimensional vector. From these low-dimensional representations, clustering was then applied to split up BJs into groups with similar vector values. These clusters were then analysed using the traditional histogram visualisation process to find those clusters that contain the desired clean traces. The raw datasets for molecules 1 and 2 were processed in this manner using PCA, t-SNE and UMAP for the purpose of DR. The low-dimensional representations for molecule 1 and molecule 2 are shown in Figure 5.4 and Figure A1.14, respectively. Upon initial inspection of each of the DR embeddings for molecule 1, it can be seen that there are very few separate clusters. Indeed, when considering t-SNE, the argument can be made that there are no distinct clusters.

Similarly, the embeddings of PCA and UMAP contain a maximum of two distinct clusters. Consequently, the trends within the clusters were then explored.

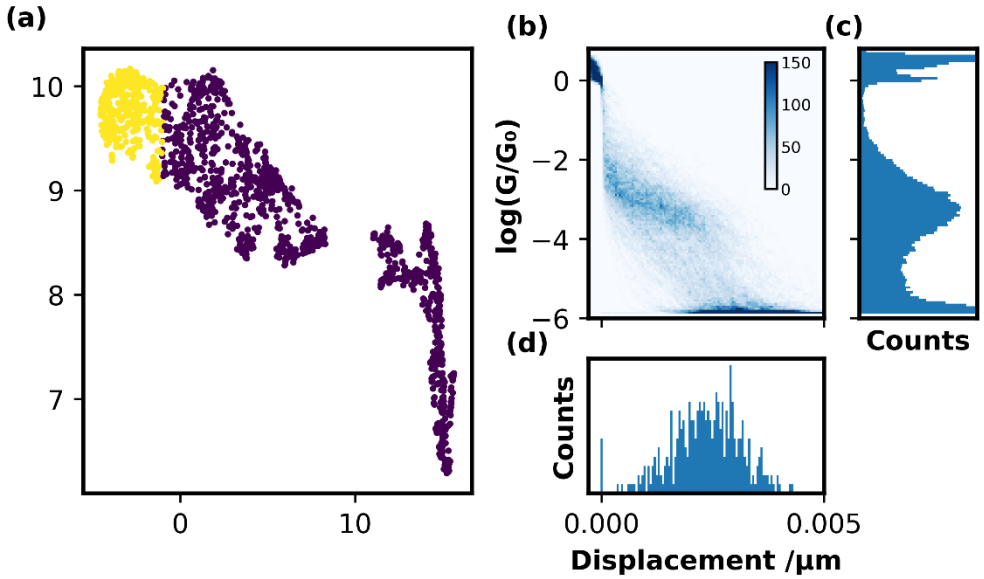



*Figure 5.5*

*Results of data cleaning using UMAP embeddings for molecule 1. (a) shows the embedded data space where the 368 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

The clusters were split up into individual sections to demonstrate how different segments within the low-dimensional space differ with regards to the high-dimensional BJ shape. The k-means clustering technique was applied to each of the three different DR technique outputs for each molecule using 10 as the number of clusters. This number of clusters was used such that the low-dimensional representations could be sufficiently split up allowing for trends in the vector space to be visualised. Each of these cluster results on each of the different DR technique results were then explored manually. Ultimately, the clusters with the cleanest corresponding BJ histograms were selected as the output of this cleaning method.

The best cluster across all three embeddings was subsequently selected. The results of this selection for molecule 1 and molecule 2 are shown in Figure 5.5 and Figure A1.19, respectively. To make this selection several criteria were used. The main histogram properties that were desired were the presence of a sharp peak at 0 on the 1D conductance histogram, a narrower plateau length distribution centred around the molecular length, and the absence of high conduction points within the 2D histogram after the initial break. In this case, a cluster selection was made from the UMAP embedding of molecule 1's BJs. Figure 5.5a shows the points within low-dimensional space that were selected to yield the corresponding histograms of Figure 5.5b-d. After this cleaning procedure was complete, there is an obvious improvement of this cleaning method from the previous filtering method. Here, there are no traces with conductance values too high, but there is also now a suitable distribution of plateau lengths which better aligns with the length of molecule 1.

The traditional approach and the DR approach to data filtering were subsequently combined to explore any combinatory effects. In this method, the data was cleaned initially using the

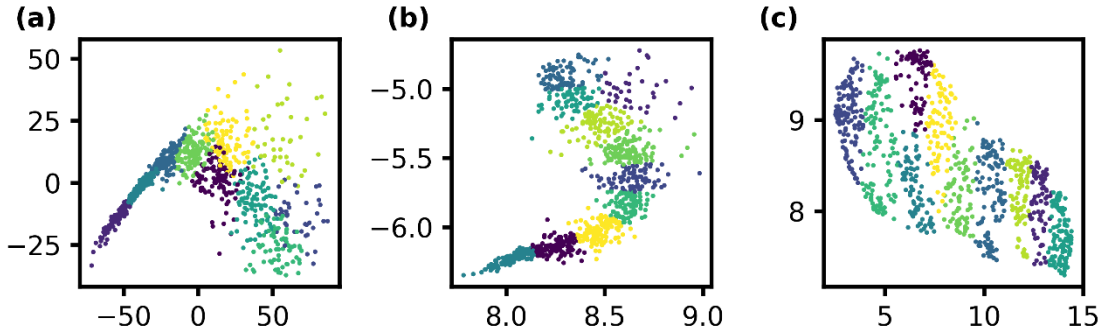



*Figure 5.6*

*2-dimensional embeddings of molecule 1's filtered BJ data. Embeddings were produced using PCA, (a), t-SNE, (b), and UMAP, (c). Points are separated into ten clusters using the k-means clustering algorithm and are coloured accordingly.*

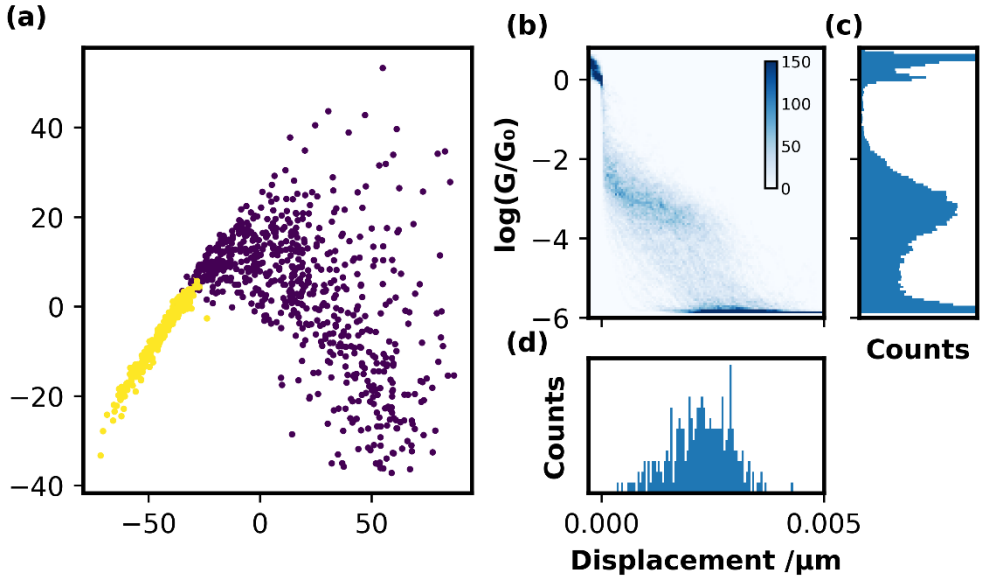



*Figure 5.7*

*Results of data cleaning using PCA embeddings on prefiltered data for molecule 1. (a) shows the embedded data space where the 269 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

thresholding method. After this, additional cleaning was then performed on the pre-filtered data using the DR method. The 2D embeddings produced are shown in Figure 5.6 and Figure A1.20 along with their cluster labels when applying k-means clustering with 10 clusters. As before, the results of this combination cleaning method are demonstrated in Figure 5.7 and Figure A1.25 in the same format. In this case, the PCA embedding of molecule 1's BJs was selected with the highlighted cluster being chosen. It is worth noting here, that although PCA was selected here for molecule 1, each of the three DR algorithms produced results that were nearly indistinguishable. While not immediately obvious, there are some improvements of this result from the previous DR only cleaning method. The most obvious change, although still subtle, is the removal of the traces with a plateau length around 0nm. These traces likely correspond to BJs where molecular binding was not captured.

Overall, the final cleaned datasets for both molecules 1 and 2 were produced. A comparison of these two results is shown in Figure 5.8. Considering molecule 1, the resultant plateau characteristics align with previous information. The plateau length of 2.30±0.68nm roughly aligns with the molecular structure, and the molecular conductance of -3.28±0.72 log($G/G_0$) aligns with other works involving this molecule.[4–6] The molecular conductance of molecule 1 is notably smaller than the conductance associated with molecule 2. Here, there are two molecular peaks present. These peaks are located at -2.93±0.42 and -3.64±0.28 log($G/G_0$) for the high and low bands, respectively. Additionally, the plateau length of molecule 2 was

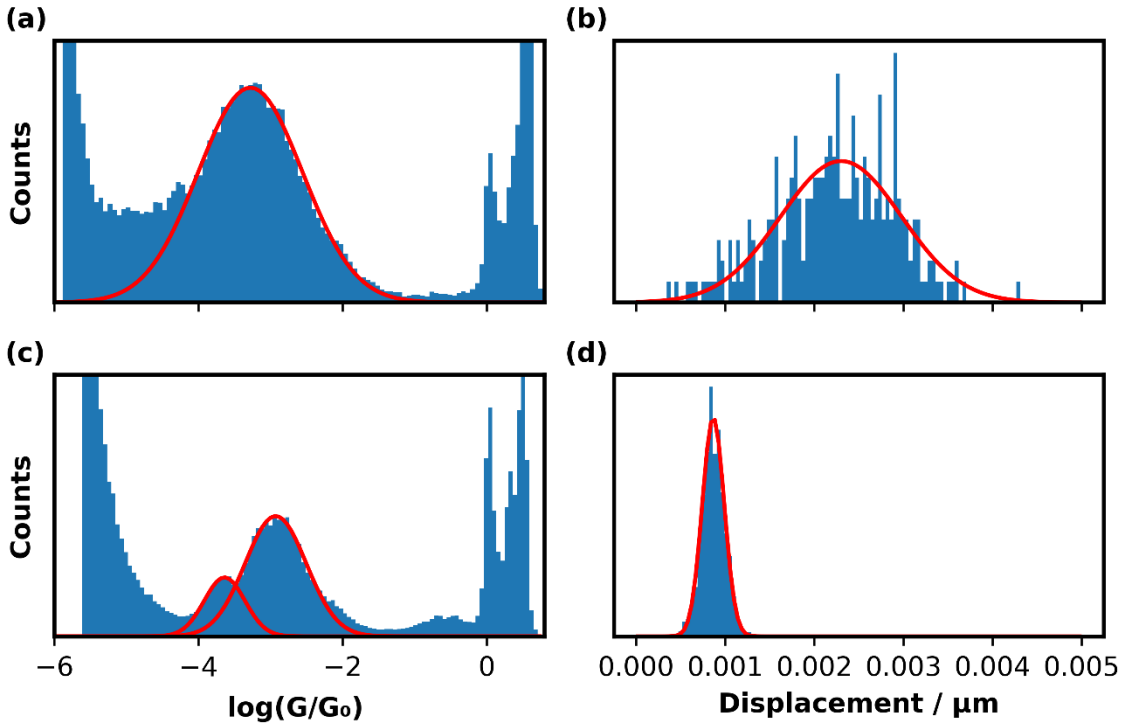



*Figure 5.8*

*Conductance and plateau length distributions for molecules 1 and 2. Molecule 1 was measured to have a conductance of -3.28±0.72 log($G/G_0$), (a), and a plateau length of 2.30±0.68nm, (b). In comparison, molecule 2 has two conductance bands at -2.93±0.42 and -3.64±0.28 log($G/G_0$), (c), and a plateau length of 0.87±0.12nm, (d).*

measured to be 0.87±0.12nm. Both of these values agree with other works involving this molecule.[7,8] However, what is not immediately obvious is the reason for this double peak feature.

### 5.2.3. Cluster Analysis

When studying the conductance of molecule 2, it can be found that two different conductance peaks are present. The exact cause of this split of high and low conductance bands has been a challenge to elucidate. Despite this, the current theory behind these different conduction bands is that this feature arises due to different molecular orientations within the junction.[8–11] When considering the high conductance band, the electrode separation is not so large that the bound molecule within the junction is fully outstretched perpendicular to the STM substrate. Instead, the molecule is present at an angle in the junction. As the withdrawal of the STM tip continues, the molecule is then pulled into the perpendicular orientation where the conductance behaviour switches to the lower value.

When investigating conductance behaviours as an ensemble using the previous histogram analyses, it is difficult to ascertain whether the two bands are present due to there being two distinct plateaus within all traces or if there are two different populations of traces with different plateau heights. When considering the previous works surrounding this problem, the prediction is that the former case will be true. To investigate whether this is indeed the case using novel methods, DR was used to analyse all BJ shapes. This involved embedding each trace into low-dimensional space followed by cluster analysis. The case of a single population containing two peaks would manifest as a single cluster in the embedding space.

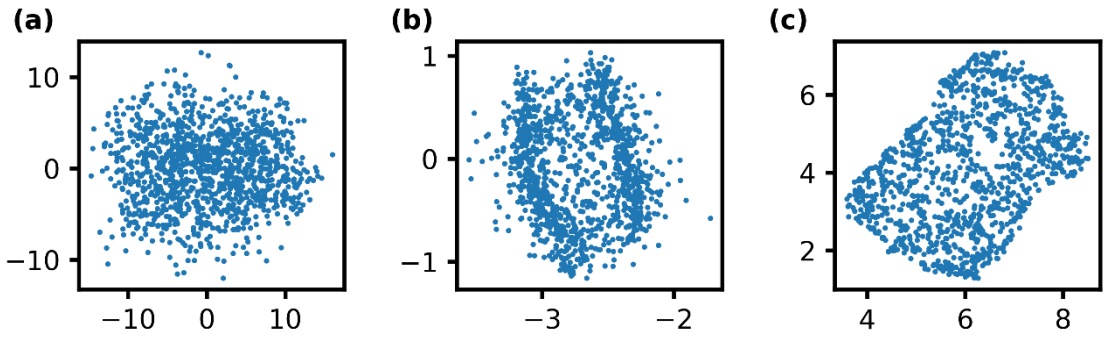



*Figure 5.9*

*2-dimensional embeddings of molecule 2's clean conductance data. Shown are embeddings produced when applying PCA, (a), t-SNE(b), and UMAP, (c).*

A clean dataset of molecule 2's BJs with the double conductance peak, found during the previous section regarding data cleaning, was taken and subsequently embedded using DR. Figure 5.9 shows the different embeddings when applying the three different DR techniques used previously. Upon first inspection of these embeddings, it can be seen that all BJs fall within the same cluster. However, close examination of the t-SNE embedding reveals that there are two areas of increased density. Because of this, the t-SNE embedding was further analysed.

The prevalent cluster of the t-SNE embedding was split into a series of subclusters. Here each sub cluster was analysed such that trends within the parent cluster could be visualised. Figure 5.10a shows how the t-SNE embedding from Figure 5.9b was divided up. In addition to this, Figure 5.10b shows the average traces generated by averaging over all traces within the corresponding subcluster. Here, when inspecting the BJs within the highlighted subclusters, it is revealed that there is little difference between the average trace of each section. Initially, this would mean that within the entire population of BJs, there are no discernible differences in features that would result from separable subpopulations. Instead, the 2D embeddings

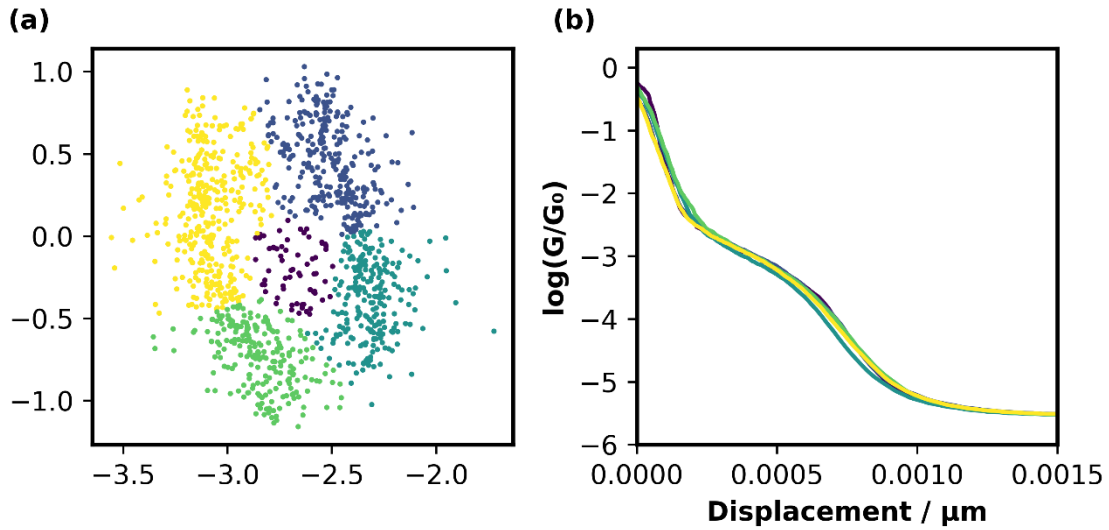



*Figure 5.10*

*Cluster analysis of the t-SNE embedding of molecule 2's BJ data. (a) shows how the embedding was split into subclusters for the comparison of average trace shape, (b).*

shown here form a cluster of traces with a distribution caused by slight deviations from an overall average trace shape.

However, in this cluster analysis, the entire conductance vectors of the clean dataset for molecule 2 were taken. Doing this has two effects on the performance of the DR algorithms. Firstly, as the conductance values were not kept with their corresponding, aligned distance vectors, each of the conductance traces were not aligned to a common reference point. This has a drastic effect on the DR performance. Because the traces are not aligned, there is a knock-on effect whereby each dimension within the high-dimensional vector space do not share the same semantic meaning. To clarify, without alignment of the conductance values, the $n$th point within conductance trace $i$, and the $n$th point within conductance trace $j$, will only relate with regards to their index position within their own vectors. With alignment, these indices would instead relate by their distance from a common reference, for example the

position of the $G_0$ conductance feature. Alignment would, therefore, provide a different low-dimensional vector space, where trends in deviations from a reference point would become accentuated. The second effect would be caused by taking all conductance values including values before and after molecule 2's molecular binding plateau. Incorporating the noisy data after the molecular plateau and before the $G_0$ plateau means these noisy datapoints will have an impact on the resulting DR embeddings. This is undesirable as the dimensions that correspond to noise locations may have a greater impact on the DR agent's training than the differences within the molecular plateau region. For instance, when considering PCA, the most important factor that affects training is the variance within dimensions.[12] If the dimensions in these noisy regions contain the highest degree of variance, then PCA would focus on maintaining this variance at the cost of hiding the variance information in other regions. Because the desire is to visualise variations in the molecular binding plateau, these areas of non-interest should be removed.

Therefore, each conductance trace was simultaneously aligned and focussed on an ROI by using their corresponding distance vectors. These distance vectors were already aligned such

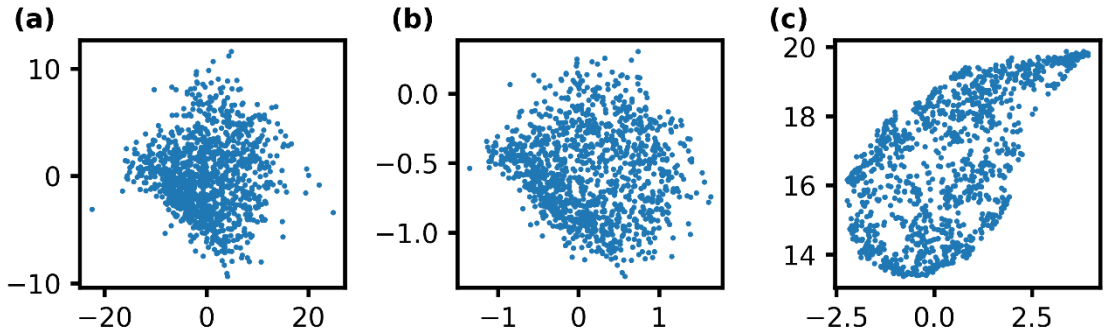



*Figure 5.11*

*2-dimensional embeddings of molecule 2's clean conductance data after alignment and ROI focussing. Shown are embeddings produced when applying PCA, (a), t-SNE(b), and UMAP, (c).*

that, within each trace, the same index at which the distance equals zero corresponds to the value of $G_0$ within the related conductance trace. Using these aligned distance vectors, the indices where the distances fell within distance limits were extracted. In this case the distances of 0.00 to 2.00nm were used. These indices were then used to extract the corresponding conductance values. The result is a new matrix of conductance values that are both aligned and focused on an ROI.

Following the alignment and focusing on the ROI, the same process as before was performed. The new traces were embedded into low-dimensional space using each of the three DR techniques used previously (Figure 5.11). Upon inspection, it can be seen that in the PCA and t-SNE embeddings there is a single region of increased density in the lower left with a range of points spreading out to the top right. Again, these embeddings point toward the conclusion that there is a single population of BJs with no subpopulations. However, the same subcluster analysis was carried out again with the intention of revealing any trends in BJ shape.

The PCA embedding was taken for this purpose. As shown in Figure 5.12a, this PCA embedding was split into three subclusters ranging from the bottom left to the top right of the vector space. For each subcluster, the corresponding average BJ was calculated and plotted (Figure 5.12b). In this instance, a difference was revealed when plotting the average traces of these three different subclusters. The average trace shape appears to have a displacement offset between different regions. More specifically, in the embedded space, those points within the upper right section of the cluster correspond to BJs with a greater displacement offset. This displacement offset results from a slower break from the $G_0$ plateau down to the molecular plateau. As consequence of this, the molecular plateau adopts a slanted shape on average

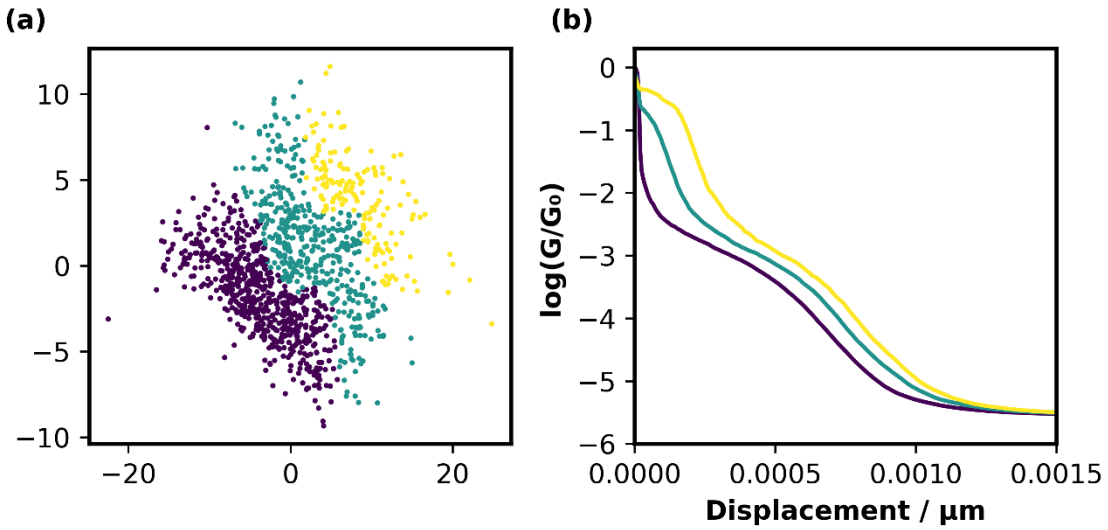



*Figure 5.12*

*Cluster analysis of the PCA embedding of molecule 2's BJ data after alignment and ROI focussing. (a) shows how the embedding was split into subclusters for the comparison of average trace shape, (b).*

instead of a flatter plateau feature. Alternatively, the lower right cluster possess BJs with a sharper break to the molecular plateau which results in a flatter plateau shape.

Overall, this DR approach to cluster analysis can reveal deviations in overall trace shapes. Here the use of PCA was able to reveal a trend of increasing displacement offset within a dataset of BJ measurements. These sorts of deviations would have been previously invisible due to the traditional analysis techniques using histograms. In addition, it was also demonstrated here how DR and clustering can aid in the cleaning and filtering of raw experimental measurements. This approach allowed undesirable traces to be filtered from the clean data while considering the entirety of traces instead of filtering out specific trace features.

### 5.2.4. Rotaxane Conductance

Using what was learnt from the previous sections regarding DR for data cleaning and cluster analysis, molecule 3 was subsequently measured and analysed using DR aided strategies. The

compound in question is similar to a so-called rotaxane, which is the name given to an arrangement of molecules such that a macrocycle becomes interlocked with a “dumbbell” shaped molecule.[13] These molecules are of particular interest as they can behave as molecular switches with multiple stable states.[14] For instance, chemical moieties can be present on a rotaxane axle such that it is energetically favourable for the interlocked macrocycle to be situated at these locations. With multiple locations, the macrocycle can switch between different stable sites and function as a molecular machine. This switching behaviour can also be controlled by external influences such as pH and result in changes to the molecular properties.[15]

It has been speculated for some time that these interlocking molecules would provide an interesting avenue of SM electronic development. Notably, it has been hypothesised that, due to the switching nature of rotaxanes, they could be suitable candidates for SM data storage.[16] Other arguments have proposed that the incorporation of a macrocycle to a long unstable molecule could result in increased chemical stability allowing for longer oligoyne-based molecular wires.[17] Indeed, one study by Milan et. al. studied a [2]-rotaxane molecule in comparison to its corresponding axle.[18] Whilst the rotaxane-protected molecule showed increased stability, the propensity of the corresponding rotaxane to form junctions was lower. In addition to this it was discovered that the axle molecule had a low conductance of -5.01 $\log(G/G_0)$. However, after incorporation of a macrocycle, the conductance increased to -4.78 $\log(G/G_0)$. This showed that incorporation of a macrocycle to a long oligoyne molecule could serve as a method of tuning molecular wire conductivity. Following this analysis, a novel pseudo-rotaxane host guest complex was synthesized and analysed using STM-BJ to further explore the potential of rotaxane-like molecules in SM electronics.

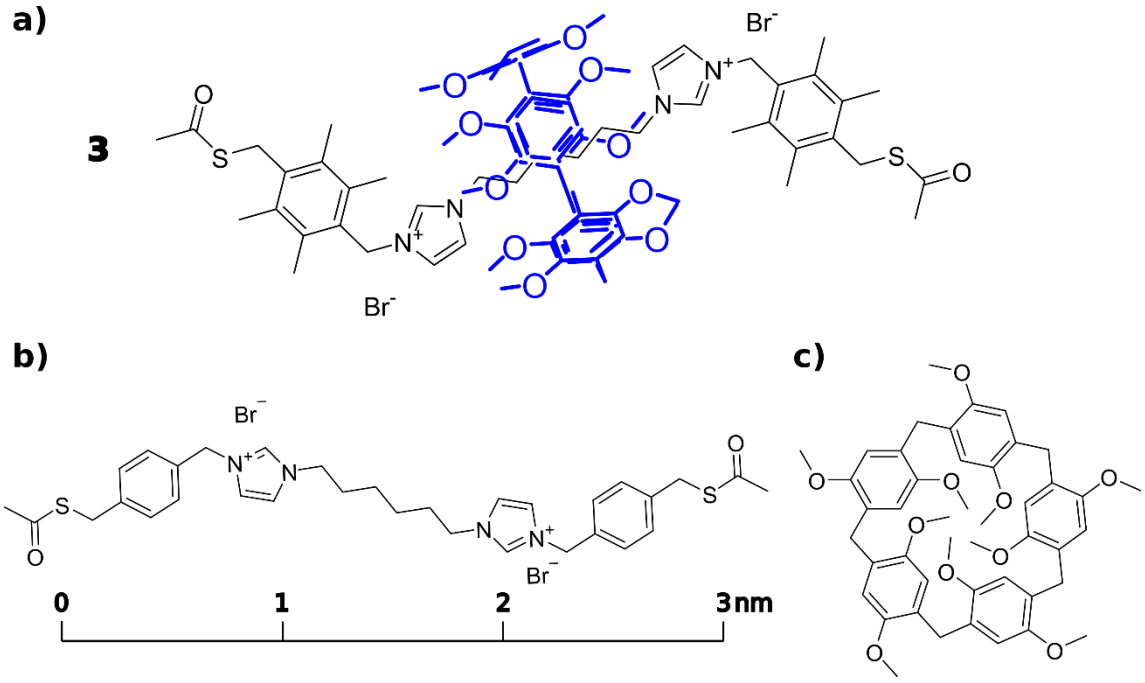



*Figure 5.13*

*Chemical structure of pseudo-rotaxane molecule 3. (a) shows the interlocked structure of the molecules axle, (b), and the macrocycle, (c). The distance between each anchoring sulphur group was measured to be 2.93nm, (b).*

The molecular structure of molecule 3, is shown in Figure 5.13. The molecule was synthesized by other colleagues from the Champness research group from the School of Chemistry. Once synthesized, the compound took the form of a solid powder. This powder was prepared for BJ analysis and measured using the steps outlined in Section 5.2.6. Following measurement, the results of BJ analysis of molecule 3 were cleaned using the traditional thresholding method. The results of which are summarised in Figure 5.14. The conductance behaviour captured here is unusual. The first notable feature is the large sloping drop in conductance shown in Figure 5.14a before the captured molecular plateaus. Additionally, after this sloping feature, the molecular plateaus themselves appear to have two levels. This is also shown in the 1D conductance histogram (Figure 5.14b). However, when considering the plateau length

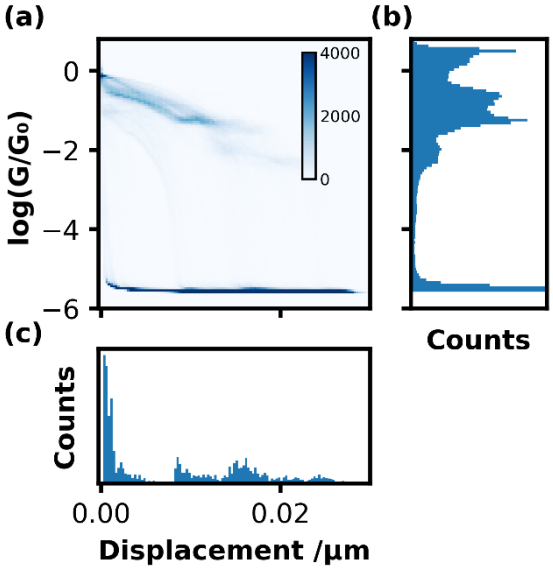



*Figure 5.14*

*Histogram analysis of molecule 3 using filtered data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. After filtering, 3937 of the 4000 sampled traces remain.*

histogram (Figure 5.14c), these results show a wide range of values with values as high as 25nm. Given that the measured molecular length is 2.93nm, this makes these results difficult to attribute to molecular binding alone as the plateau lengths are too large. The reason for these large plateau lengths is likely due to the large sloping feature mentioned previously as the plateau lengths are calculated using the distance from the datapoints following the conductance drop from 0 $\log(G/G_0)$ to the trace noise level measured below -5 $\log(G/G_0)$. Regarding the traces as individuals, their appearance will be drastically different to the example presented in Figure 1.1 due to the presence of the large sloping feature before the molecular plateau, and due to the presence of there being two distinct molecular plateaus.

The overall appearance of the break junction behaviour is likely due to the long immersion time of over six hours. Because of this, it is likely that multiple molecules will exist on the

surface near the region where the STM tip approaches such that multiple molecules can bridge the gap between the tip and substrate. This would explain the higher-than-expected conductance readings. Given the conjugation of the pseudo-rotaxane complexes, aggregates could form which would further facilitate the binding of multiple molecules in parallel to the tip and substrate. Due to the complexity of this system, determining the cause of each feature within the BJs is not currently possible without further experimentation. Whilst it is possible for the large sloping feature to manifest due to strong gold-sulphur bonds as with molecule 1 (Section 5.2.1), this is not the guaranteed cause. Additionally, the presence of two molecular plateau levels could be explained by changes to the orientation of bound molecular complexes similarly to molecule 2, (Section 5.2.3), however this cannot be proved without knowledge of the exact nature of the bound complexes.

Following this initial measurement, a repeat experiment was performed with a shorter immersion time of four hours with the intention of reducing molecular surface coverage. The results of this second experiment are summarised in Figure 5.15. Here, there is no sloping feature present within the 2D histogram (Figure 5.15a). Indeed, the appearance of this immersion's conductance-displacement behaviour aligns with the traditional BJ appearances shown previously. That is, there is a strong conductance peak present at the conductance quantum, and there is only one molecular plateau region present with a more consistent break off distance. This result presents itself as a more tidy and reliable measurement, however, when considering the plateau length histogram (Figure 5.15c), the value of 2.37±0.31nm does not align with the molecular dimensions in Figure 5.13. Whilst this result doesn't match, it could be explained by either the junction breaking before the molecule has been fully

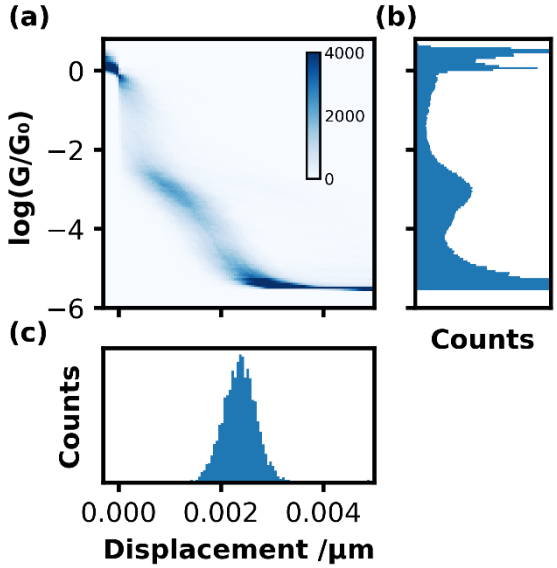



*Figure 5.15*

*Second histogram analysis of molecule 3 using filtered data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. After filtering, 3861 of the 4849 sampled traces remain.*

extended, or the electrode surfaces rearranging after the rupture of the $G_0$ junction. Overall, the behaviour of this second repeat is more easily interpreted than the first experiment.

With two datasets of very different appearance, DR was subsequently applied to simultaneously clean and filter out subclusters within the datasets that align with what is known from the molecular structure. Namely, subclusters within the 2D embeddings were extracted if they contained plateau length features that matched the molecular length. Additionally, subclusters were removed if they contained no molecular plateau, or possessed noise features that obscured the molecular plateau. From the molecular structure, predictions can be made as to the expected BJ molecular plateau length. Following the same procedure as with molecules 1 and 2, the first experimental dataset of molecule 3 was analysed. Here, PCA was used to produce the 2D embedding presented in Figure 5.16a. The highlighted region

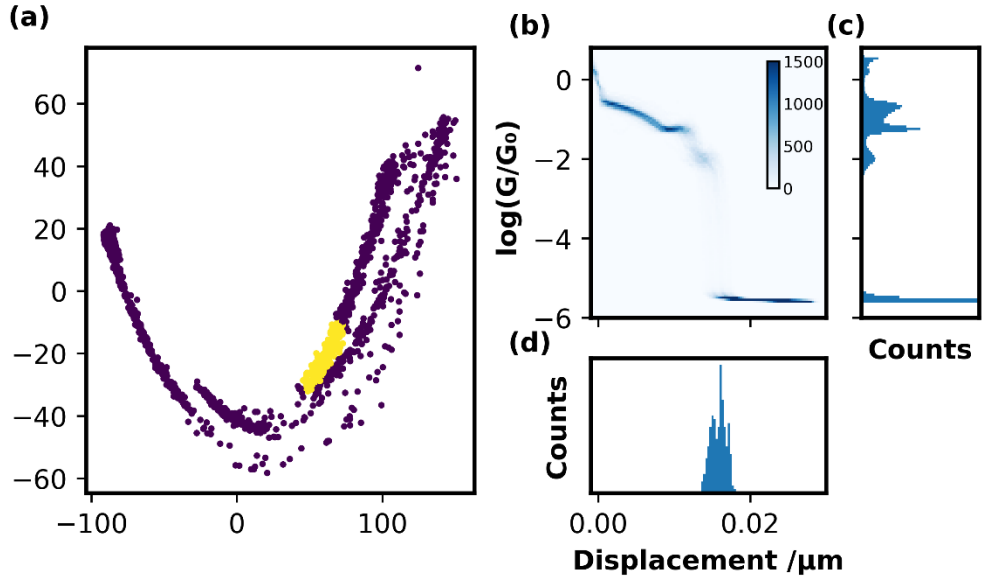



*Figure 5.16*

*Results of data cleaning using PCA embeddings on molecule 3's prefiltered data. (a) shows the embedded data space where the 233 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

was extracted to yield the conductance and displacement histograms shown in Figure 5.16b. This dataset of results returned from the ML cleaning strategy has a much cleaner and consistent conductance-displacement relationship. However, despite this cleaning the plateau length distribution is still offset from the undesirable sloping feature. To address this, the displacement requires alignment to a more appropriate reference point instead of the previously used $G_0$ distance. In this instance, each BJ trace should be aligned such that the displacement value of 0nm corresponds to the first point of the molecular plateau. This task is not trivial as this point differs between individual traces in terms of index position, displacement value, and even conductance value.

Locating the start of the molecular plateau for each trace is related to a common problem in time-series analysis known as change-point detection. This challenge has been previously

explored briefly in Chapter 4 when locating anomalies in I(t) data where the so-called "one-window" approach was incorporated. In this instance, however, to find the location of molecular plateaus, the "two-window" approach is used instead as this strategy can locate change-points more precisely. Whilst the one-window approach utilises the iteration and measurement of a windows value across a time-series, the two-window approach instead iterates and measures two windows with values. These two windows are always located together with one window representing the future and the other representing the past. From these two values a difference is calculated. When this difference exceeds some threshold, a change point is flagged.[20]

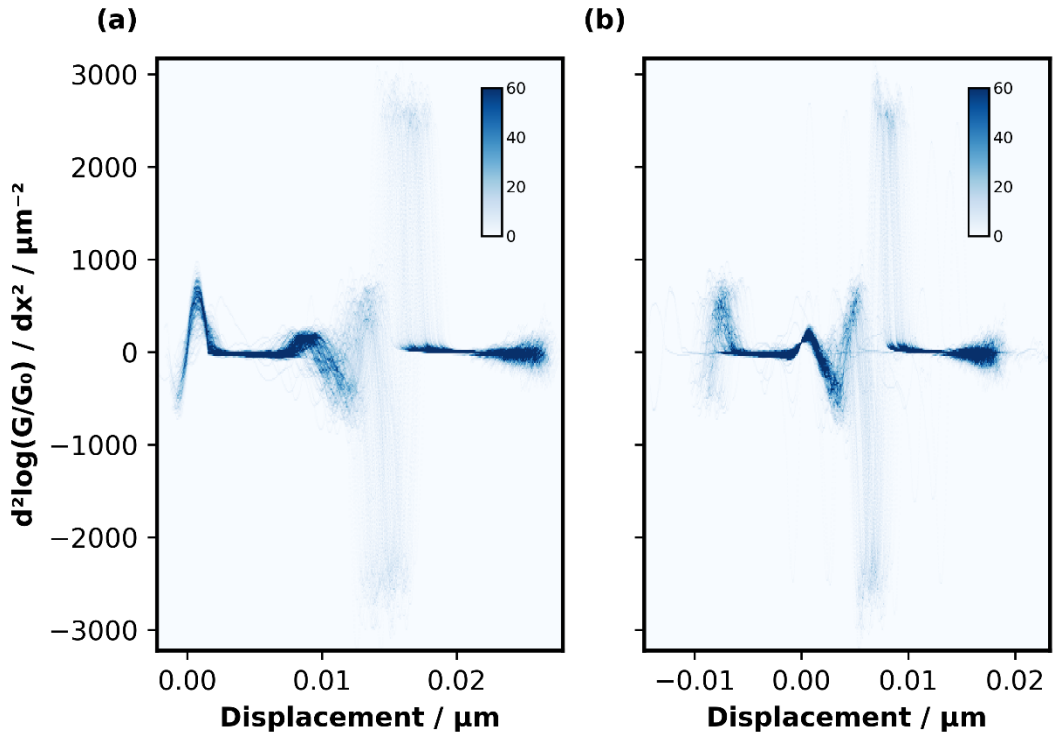



*Figure 5.17*

*Histograms of the conductance's approximate second derivative vs displacement. (a) shows the initial 2D relationship, whereas (b) shows the relationship after alignment at the chosen change point.*

This two-window technique was thus applied to every BJ trace within the first experimental dataset of molecule 3. Here two windows of size 100 were used to measure the gradients at different past and future positions within traces. The difference of these gradients for each trace as the windows iterated through were plotted on a 2D dimensional histogram. This is shown in Figure 5.17a. As these plots show differences in gradients, the resultant plots can be seen as numerical approximations of the second derivative of their corresponding BJs. Therefore, at the location where the sloping feature ends and the molecular plateau starts, the second derivative will have a positive value. However, the first notable instance within each trace where the second derivative becomes positive would be where the sloping feature starts after the $G_0$ plateau. The second instance where the second derivative becomes positive would be the location where the desired change point lies. Looking at the 2D histogram of the second derivative it can be seen that the second occurrence of positive values is consistent throughout the majority of traces. This makes this event an ideal target for change point detection.

Using these second derivative plots, the starts of molecular plateaus were found by locating the first point after the $250^{th}$ index within each trace that exceeded the value of $100\mu m^{-2}$. The skip of the initial 250 data points within each break junction meant that the initial spikes in the second derivative plots, resulting from $G_0$ plateaus, could be avoided whilst maintaining a high enough sensitivity to detect the start of the molecular plateau. Once each location was found, the corresponding displacement offsets were extracted and used to zero each trace at their own corresponding change-point. The resultant aligned second derivative traces are shown in Figure 5.17b. Here it can be seen that the features located after the new zero point, have become more pronounced on the 2D histogram. This is because the curve features after

this zero-point have become aligned and less variable. This will result in a more consistent plateau length measurement in the corresponding BJ traces.

After alignment in this manner, the desired displacement limits were applied to remove the undesirable sloping feature. More specifically, for each trace, the data before the molecular plateaus were disregarded. To do this, any conductance value with a corresponding displacement value less than zero was removed. The resultant conductance and displacement relationship was produced as shown in Figure 5.18. After this alignment and ROI focussing, the undesirable sloping feature has been removed (Figure 5.18a). Due to the removal of this feature, there is now a more consistent and visible 1D conductance histogram (Figure 5.18b). This makes measurement of the upper and lower conductance values easier to achieve. Also, when considering the plateau length distribution (Figure 5.18c), the values now range

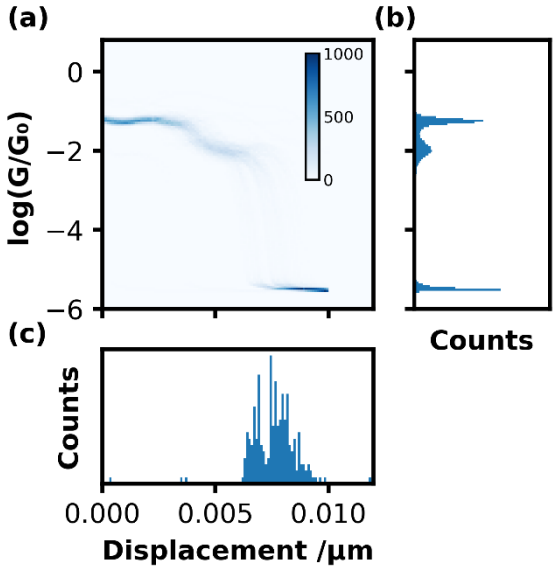



*Figure 5.18*

*Histogram analysis of molecule 3 after DR cleaning, alignment, and ROI focussing. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths*

between 6 and 10nm. This now aligns with the assumption that two molecules are present within the junction to yield two molecular plateaus. The combined length of two molecules would roughly equate to 6nm with variation resulting from the molecular junction breaking early, or the addition of multiple substrate atoms.

Overall, two different behaviours were observed with molecule 3. Experiment 1 produced BJ traces with a large sloping feature followed by two separate plateau regions. This dataset required extensive cleaning to produce the final result providing two mean conductance values of -1.25±0.08 $\log(G/G_0)$ and -1.96±0.21 $\log(G/G_0)$, respectively. Whilst this result had plateau lengths that corresponded to the molecular length, the presence of the sloping feature, and the omission of a clean $G_0$ peak in the conductance histogram calls for repeat measurements. Following this initial measurement, a second behaviour was observed. For experiment 2, the resultant BJ traces yielded a single molecular plateau at -3.08±0.57 $\log(G/G_0)$ which is lower than what was observed previously (Figure 5.15). This measurement, however, contained plateau lengths with an average of 2.37±0.31nm and are therefore shorter than the fully extended structure of the molecule. In this instance, however, there was a sharper, more obvious $G_0$ peak that indicates a cleaner experiment. Either way, both results revealed conductance behaviour that is larger than expected given the length and lack of conjugation of molecule 3's axle (Figure 5.13). In summary, molecule 3 presents interesting electrical behaviour that requires further investigation. This would involve repeat BJ measurements of molecule 3, as well as measurements of molecule 3's axle without the macrocycle. This would allow a better understanding of the electrical behaviour of rotaxane-like molecules that could result in a larger range of possible molecules for SM electronics.

### 5.2.5. Concluding Remarks

In conclusion, a total of three molecules have been analysed using STM-BJ. These three molecules include two well-known and previously measured molecules, as well as a third novel pseudo-rotaxane molecule. The two initial molecules were used to benchmark the use of DR for ML-enhanced BJ analysis. In the case of molecule 1, it was shown that DR can be an effective tool for aiding in the cleaning and filtering of large datasets whilst incorporating the entirety of measured traces. Here, a clean subset was extracted from a dataset containing a large proportion of undesirable traces. In addition to this, DR was then used to analyse a clean dataset of molecule 2's conductance measurements to analyse subpopulations within a parent population of BJ traces. Trends within a cluster of BJs were analysed by extracting subpopulations from a uniform cluster in a low-dimensional embedding. This allowed changes in overall trace shape across the population to be elucidated with relative ease.

After these two benchmark molecules were explored from a DR perspective, this ML analysis method was then applied to molecule 3 to aid in the analysis of a previously untested molecule. Here DR aided in extracting subpopulations of traces that agreed with models of molecule 3's length. Using this approach several conductance behaviours were discovered. All of which possess magnitudes that are greater than expected for the corresponding molecular length.

The overall properties for each molecule that were measured are summarised in Table 5.1. The acquisition of each of these results was greatly aided by the incorporation of ML techniques. These results, therefore, point to the usefulness of ML techniques in aiding with BJ analysis, as a novel molecule has been brought to light with relative ease.

*Table 5.1*

*Summary of BJ results for each molecule. Results show the measured molecular conductance of any plateaus present. Also shown are the plateau lengths compared with their expected lengths taken from molecular models.*

| Molecule | Conductance (High) / log(G/$G_0$) | Conductance (Low) / log(G/$G_0$) | Plateau Length / nm | Expected Length / nm |
|---|---|---|---|---|
| **Molecule 1** | -3.28±0.72 | | 2.30±0.68 | 2.01 |
| **Molecule 2** | -2.93±0.42 | -3.64±0.28 | 0.87±0.12 | 0.72 |
| **Molecule 3i** | -1.25±0.08 | -1.96±0.21 | 7.45±1.01 | 2.93 |
| **Molecule 3ii** | -3.08±0.57 | | 2.37±0.31 | 2.93 |

### 5.2.6. Materials and Methods

For this project the materials and methods used are largely the same as those stated in Section 3.4.1. The same STM setup is utilised, and the same sample preparation steps are adopted. However, when preparing STM tips for BJ experiments, the coating step is omitted as electrochemical control is not required. Because of this, the preparation of counter and reference electrodes is also omitted. Assembly of the STM sample plate and scanner is also the same with the only difference being the removal of the extra electrodes from the process.

The three molecules used in this chapter were prepared as follows. For molecule 1, 8.531mg of S,S'-[1,4-Phenylenebis(2,1-ethynediyl-4,1-phenylene)]bis(thioacetate) was dissolved in a 2ml solution of a 1:4 mix of tetrahydrofuran and mesitylene to yield a 10mM stock solution. This solution was then diluted to 100µM by mixing 20µl of the 10mM stock with 1.98ml of the 1:4 THF mesitylene mix. The resultant 100µM solution of molecule 1 was then aliquoted onto the STM sample plate prior to BJ sampling. The same approach was also used to prepare a 100µM solution of molecule 2. Here 3.124mg of 4,4'-bipyridine was dissolved in the same

solvent solution to yield a 10mM stock. From there the same dilution took place by mixing 20µl of this stock with 1.98ml of the solvent mix. This 100µM stock was then aliquoted onto the STM sample.

Molecule 3, however, required a slightly different approach. In this instance, a 100µM stock was prepared by diluting a 1mM stock. The initial 1mM stock was prepared by dissolving 3.199mg of the pseudo-rotaxane powder in 2ml of acetonitrile. This concentrated stock was further diluted by aliquoting 0.2ml into 1.8ml of acetonitrile. Following this, to functionalise the gold substrate for BJ analysis, the gold substrate was immersed in the prepared 100µM solution prior to STM assembly. After immersion for the desired time, typically between 4 to 8 hours, the sample is briefly washed with 1ml of acetonitrile to remove any excess residues. Assembly then proceeds as normal.

Once set up, BJs were sampled with the following logic. The STMs setpoint was set to a value higher than the conductance quantum. In this case, on the logarithmic preamplifier utilised, the setpoint of 2.5V was set at an applied bias of 0.05V. These parameters were used for all three molecules. Once the high setpoint is acquired, the BJ withdrawal distance and the withdrawal speed are set. A summary of these parameters for each molecule are presented in Table 5.2. BJs are then sampled in bursts of 2000 withdrawals. Between each withdrawal the system has an automatic settling time to allow time for the setpoint to be reacquired. This is set to 2 seconds for all experiments.

*Table 5.2*

*Summary of BJ parameters used for each molecule. These are the withdrawal distance and the withdrawal speed of the STM tip.*

| Molecule | Withdrawal Distance / μm | Withdrawal Speed / $\mu ms^{-1}$ |
|---|---|---|
| **Molecule 1** | 0.015 | 0.008 |
| **Molecule 2** | 0.010 | 0.010 |
| **Molecule 3-1** | 0.030 | 0.029 |
| **Molecule 3-2** | 0.010 | 0.010 |

Lastly, the ML analyses of all BJ results were carried out using the same python programming environment stated previously in Section 3.4.3. On top of this a new package was installed. This is the umap-learn (v0.5.3) package to provide access to the UMAP DR algorithm.

The manual cleaning algorithm was also written in the previous programming environment with the following logic. Firstly, all traces were aligned with the $G_0$ reference point. To do this a window of 10 samples was iterated through each trace. The first location where the average value of the window dropped below a conductance value of -0.1 $\log(G/G_0)$ was labelled as the zero-mark. The corresponding displacement traces were then offset to make the distance that corresponds to the zero-mark equal 0. From here, each trace's plateau length was calculated. This was achieved by using the same iterating window of size 10 approach. The first average conductance value to drop below the threshold of -5.0 $\log(G/G_0)$ was found to extract the corresponding displacement. After this alignment and plateau length calculation, traces were then filtered. Three properties were tested in total. These are the trace's average conductance, the traces plateau length, and the average value of the traces last 10 points. Traces were removed from the dataset as dirty traces if these properties were measured to be too large

or too small. The filters used for each of the three molecules are summarised in the following table (Table 5.3).

*Table 5.3*

*Summary of thresholds used for cleaning each molecule's BJs. These include thresholds for the trace's average current value, the trace's plateau length, and the noise oscillations within the trace's tails.*

| Molecule | High $\log(G/G_0)$ | Low $\log(G/G_0)$ | High Plateau Length / nm | Low Plateau Length / nm | Tail Threshold |
|---|---|---|---|---|---|
| **Molecule 1** | -3.0 | -6.0 | 10.0 | 0.0 | -4.0 |
| **Molecule 2** | -3.0 | -6.0 | 2.0 | 0.0 | -4.0 |
| **Molecule 3** | -0.5 | -6.0 | 30.0 | 0.0 | -3.0 |

## 5.3. Cyclic Voltammetry

The application of DR for subpopulation analysis and clustering is not limited to the previously demonstrated application of BJ analysis. The use of this analysis approach can be applied to any problem set where vast amounts of high-dimensional data are produced. To demonstrate this versatility, DR and trend analysis was applied within the field of CV to quantify overall changes in CV shapes.

### 5.3.1. Unsupervised Classification of Voltammetric Data Beyond Principal Component Analysis

One of the challenges that are present within electrochemical analysis of compounds is the interpretation of CVs. These challenges, previously discussed in Section 1.2.1, demand a considerable amount of manual processing and analyses to understand the mechanical differences between individual CVs. Therefore, the idea was conceived that ML could potentially serve as an automated solution to organising CVs into groups of like characteristics

by incorporating their entire shape. As this challenge requires an unsupervised approach to analysing trends in groups, DR was aptly chosen for this task.

In addition, it is generally accepted that, for ML analyses to be successful large amounts of data are required.[21–23] Contrary to this, CV datasets typically do not involve thousands of individual repeats of the same environment.[24] Because of this, the idea to apply DR to a small dataset was coupled with the ML technique known as TL. TL is a technique that aims to improve learning with small datasets by incorporating pretrained models into a new problem set. This generally results in improved performance than training a new model alone on a small dataset.[25]

From these ideas, a CV study to explore the potential of TL and DR in an electrochemical setting was carried out by Weaver et al. (Appendix 3).[26] This study involved the generation of a dataset of 18 CVs using a commercially available simulation software. Within these 18 CVs, there existed two properties that were varied between each. These properties are the electrochemical reaction mechanism and the corresponding electrode radius. Three reaction mechanisms were selected. These are the E type, the EC type, and the ECE type mechanisms. E corresponds to an electron transfer reaction, whereas C corresponds to a chemical reaction. Therefore, ECE corresponds to a three-stage reaction of an E reaction followed by a C reaction then another E reaction. Within these three groups there exist six CVs each with a separate electrode radius, which range uniformly on a logarithmic scale from $1x10^{-1}$ to $1x10^{-6}$cm. These CVs were used to create three separate datasets of 18 CVs that each correspond to different representations of the initial traces. This includes a vector, an image, and a features representation. The vector representation corresponds to the raw CV trace taken as a 600-

dimensional vector. The image representation manifests as the plotted CVs saved as 1080 by 1080 RGB images. Finally, the features representation corresponds to the images dataset after having features extracted using a pretrained VGG-16 convolutional head.

For each of the three different datasets of pre-processed CVs, DR was applied using PCA, t-SNE, and UMAP. Once the low-dimensional embeddings were produced, the resultant 2D vectors were plotted and labelled with their corresponding reaction mechanism and electrode radius. For many of the representations, there was a clear trend of increasing electrode radius across the low-dimensional vector spaces. To quantify how well CVs with like electrode radii clustered in the embeddings, the silhouette score was implemented. Here the silhouette scores for each point in a cluster of like electrode radii were calculated and averaged to provide a measure of accuracy for the clustering. Overall, it was discovered that applying t-SNE to the vector dataset of CVs was best at expressing the likeness of electrode radii in low-dimensional space.

Additionally, another metric was incorporated. This metric was simply the sum of the three distances between each CV's 2D vector at a specific electrode radius. This metric, called the perimeter score, provides a degree of separation between CVs of different electrode mechanism. Within each representation there were six perimeter scores calculated and subsequently averaged to yield the overall score for an embedding. Ultimately, it was found that applying t-SNE to the vector representations also yielded the highest average perimeter score.

Given that t-SNE was able to yield the highest silhouette and perimeter scores, it was concluded that this DR technique was best for representing the CVs in 2D space whilst

demonstrating the mechanical differences between the reaction conditions. It is also worth noting that additional investigations into the effect of noise on the performance of each DR technique was carried out. Alongside this, there was also additional steps taken to ensure that each of the different methods utilised the optimum parameters for their chosen tasks. Both of these, as well as additional figures were presented in this study's supporting information, (Appendix 3).

Overall, the use of DR on a small dataset of CVs was able to express trends in complex data shapes without the need for any prior assumptions or manual data processing. Additionally, these trends matched the trends of the governing parameters, where CVs produced from the same electrode radius clustered together in the 2D embedded space. This demonstrates an additional application of ML techniques for unsupervised classification.

## 5.4. References


1 T. Fu, K. Frommer, C. Nuckolls and L. Venkataraman, *J Phys Chem Lett*, 2021, **12**, 10802–10807.

2 J. M. Hamill, C. Weaver and T. Albrecht, *The Journal of Physical Chemistry C*, 2021, **125**, 26256–26262.

3 D. Krüger, H. Fuchs, R. Rousseau, D. Marx and M. Parrinello, *Phys Rev Lett*, 2002, **89**, 186402.

4 H. Lissau, R. Frisenda, S. T. Olsen, M. Jevric, C. R. Parker, A. Kadziola, T. Hansen, H. S. J. van der Zant, M. Brøndsted Nielsen and K. V. Mikkelsen, *Nat Commun*, 2015, **6**, 10233.

5 R. Huber, M. T. González, S. Wu, M. Langer, S. Grunder, V. Horhoiu, M. Mayor, M. R. Bryce, C. Wang, R. Jitchati, C. Schönenberger and M. Calame, *J Am Chem Soc*, 2008, **130**, 1080–1084.

6 V. Kaliginedi, A. V. Rudnev, P. Moreno-García, M. Baghernejad, C. Huang, W. Hong and T. Wandlowski, *Physical Chemistry Chemical Physics*, 2014, **16**, 23529–23539.

7 T. Kim, P. Darancet, J. R. Widawsky, M. Kotiuga, S. Y. Quek, J. B. Neaton and L. Venkataraman, *Nano Lett*, 2014, **14**, 794–798.

8 S. Y. Quek, M. Kamenetska, M. L. Steigerwald, H. J. Choi, S. G. Louie, M. S. Hybertsen, J. B. Neaton and L. Venkataraman, *Nat Nanotechnol*, 2009, **4**, 230–234.

9 Z. Yu, Y. Xu, J. Su, P. M. Radjenovic, Y. Wang, J. Zheng, B. Teng, Y. Shao, X. Zhou and J. Li, *Angewandte Chemie International Edition*, 2021, **60**, 15452–15458.

10 M. Kamenetska, S. Y. Quek, A. C. Whalley, M. L. Steigerwald, H. J. Choi, S. G. Louie, C. Nuckolls, M. S. Hybertsen, J. B. Neaton and L. Venkataraman, *J Am Chem Soc*, 2010, **132**, 6817–6821.

11 M. Baghernejad, D. Z. Manrique, C. Li, T. Pope, U. Zhumaev, I. Pobelov, P. Moreno-García, V. Kaliginedi, C. Huang, W. Hong, C. Lambert and T. Wandlowski, *Chem. Commun.*, 2014, **50**, 15975–15978.

12 J. Camacho, J. Picó and A. Ferrer, *Chemometrics and Intelligent Laboratory Systems*, 2010, **100**, 48–56.

13 H.-Y. Zhou, Q.-S. Zong, Y. Han and C.-F. Chen, *Chemical Communications*, 2020, **56**, 9916–9936.

14 C. A. Schalley, K. Beizai and F. Vögtle, *Acc Chem Res*, 2001, **34**, 465–476.

15 C. Clavel, C. Romuald, E. Brabet and F. Coutrot, *Chemistry - A European Journal*, 2013, **19**, 2982–2989.

16 P. Wu, B. Dharmadhikari, P. Patra and X. Xiong, *Nanoscale Adv*, 2022, **4**, 3418–3461.

17 S. Anderson, R. T. Aplin, T. D. W. Claridge, T. Goodson III, A. C. Maciel, G. Rumbles, J. F. Ryan and H. L. Anderson, *J Chem Soc Perkin 1*, 1998, 2383–2398.

18 D. C. Milan, M. Krempe, A. K. Ismael, L. D. Movsisyan, M. Franz, I. Grace, R. J. Brooke, W. Schwarzacher, S. J. Higgins, H. L. Anderson, C. J. Lambert, R. R. Tykwinski and R. J. Nichols, *Nanoscale*, 2017, **9**, 355–361.

19 A. Vladyka, M. L. Perrin, J. Overbeck, R. R. Ferradás, V. García-Suárez, M. Gantenbein, J. Brunner, M. Mayor, J. Ferrer and M. Calame, *Nat Commun*, 2019, **10**, 262.

20 C. Truong, L. Oudre and N. Vayatis, *Signal Processing*, 2020, **167**, 107299.

21 J. G. A. Barbedo, *Comput Electron Agric*, 2018, **153**, 46–53.

22 A. Vabalas, E. Gowen, E. Poliakoff and A. J. Casson, *PLoS One*, 2019, **14**, e0224365.

23 J. Prusa, T. M. Khoshgoftaar and N. Seliya, in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2015, pp. 96–102.

24 D. M. Morales and M. Risch, *Journal of Physics: Energy*, 2021, **3**, 034013.

25 F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He, *Proceedings of the IEEE*, 2021, **109**, 43–76.

26 C. Weaver, A. C. Fortuin, A. Vladyka and T. Albrecht, *Chemical Communications*, 2022, **58**, 10170–10173.

# Chapter 6

## Conclusions and Recommendations for Future Work

In Chapter 1, the difficulties of SM experimentation were introduced. Namely, the use of highly sensitive equipment to measure nanoscale physical objects is fraught with notorious variation in measurement that require vast amounts of data to be collected. The traditional approach to analysis of SM datasets relies on statistical analysis of distributions through histogram analysis. The example of STM-BJ and the STM I(t) technique were introduced to demonstrate this. However, the application of statistical methods to large, high-dimensional datasets with a wide variety of influencing variables is difficult. For such an analysis to proceed, assumptions are made regarding which variables are the most decisive factors for a given investigation. In the example of I(t) events, the event height and duration are the most commonly analysed properties. Analysis processes such as these result in a large loss of information to ease the analytical process, which can result in less than optimum results.

Typically, the goal of an I(t) technique study is to use the measured behaviour to make predictions about the nature of the captured single molecule. The most explored application of this is the classification of I(t) events for DNA sequencing. Here, ML techniques are better suited for the purposes of data classification as these techniques typically possess higher predictive capabilities when compared to statistical techniques. Indeed, Chapter 3 presented a simulated I(t) dataset of five classes which possessed events with properties that overlapped considerably. Traditional statistical methods would be unable to separate the individual classes, whereas the use of an RFC and a CNN could both produce classification accuracies of 95%. These results proved that ML techniques within the classification of I(t) events have superior classification capabilities when compared to distributional analyses.

Because of the success of the classification of these simulated I(t) events, the use of electrical tunnelling currents for the purpose of nucleotide classification appears more plausible as accurate classification was shown to be possible despite the overlap of event properties. Therefore, in an effort to ease experimentation with the I(t) technique such that an experimental dataset of nucleotide data could be acquired, Chapter 3 also presented a novel experimental procedure. Here, the "soft" I(s) technique was implemented to determine the tunnelling decay constant for the aqueous medium used in nucleotide measurements. With this constant, any measured setpoint could be mapped to a corresponding tunnelling distance. This mapping was used to calculate a setpoint that resulted in a tunnelling distance that matched the physical dimensions of the molecule to be measured. Overall, this approach, in tandem with electrochemical control, allowed I(t) events to be detected without the need for recognition elements or nanofluidic channels that are present in the literature.

Despite the successful measurement of rAMP using the STM I(t) technique, the next challenge of data analysis was made apparent. Whilst I(t) events could be measured, the extraction of these events from I(t) traces proved to be challenging as extracted events could be classed as either being true or false positives. Therefore, Chapter 4 explored various methods by which anomalies could be detected within time-series datasets. To provide a comparison of statistical methods against ML techniques both statistical and ML methods were incorporated. In summary, the highest event detection accuracy was achieved by a statistical technique whereby a moving average was applied to exaggerate true events from noise. This technique achieved a ROC AUC score of 0.97 when detecting events with an SNR of only 0.1. In comparison a convolutional AE achieved a lower score of less than 0.7. This result showcased an instance where a statistical method resulted in a higher predictive score than an ML agent.

Overall, Chapter 4 presented a thorough investigation into anomaly detection capabilities that can be achieved with I(t) event detection, which resulted in the successful capture of events with an SNR as low as 0.1, given that events contain a sufficient number of datapoints.

With effective methods for both nucleotide measurement via the I(t) technique and anomaly detection in time-series data having been achieved, subsequent experimentation should be carried out to acquire a full dataset of extracted I(t) events for each of the four DNA nucleotides. From here CNN classifiers should be employed for the classification of experimental data to ascertain if the same high accuracy can be achieved with data sampled from real world sources. Additionally, whilst high accuracies were achieved for both I(t) classification and anomaly detection, future work should also strive to achieve even more accurate results. To that end, alternative CNN architectures should be explored for classification, and alternative ML algorithms should be explored for anomaly detection. As an alternative to AEs for anomaly detection, regression and forecasting agents should be investigated.

In addition to the classification and anomaly detection studies, other experiments have been discussed where unsupervised ML techniques have been the focus. To demonstrate the versatility and clustering capabilities of ML techniques, DR techniques were showcased in an STM-BJ setting and a CV setting within Chapter 5. In both cases, application of DR techniques to datasets was able to effectively visualise trends in the overall shapes of both BJ traces and CVs. For STM-BJs, DR was effectively implemented for the purpose of data cleaning by separating undesirable traces from tidy BJ pulls. This separation ability was also used to explore variations in molecular plateau shapes and was able to visualise more complicated

trends that were obscured within conductance histograms. This unsupervised strategy was also used to aid in the characterisation of an original pseudo-rotaxane molecule. By clustering 2D representations of BJ measurements, a clean dataset was produced that displayed a higher-than-expected molecular conductance. This result demonstrates that rotaxane-like molecules may have the potential to serve as long molecular wires with high conductivity which would be invaluable to SM electronics.

The promising pseudo-rotaxane results were also made possible due to the incorporation of a unique alignment process. Using concepts from anomaly detection, change point detection was employed in an analysis pipeline where the start of molecular plateaus could be automatically located for each individual trace. This process meant that, instead of aligning at the distance where conductance breaks from $G_0$, traces can be aligned to the start of the molecular plateau. Because of this new alignment, the resultant plateau length distribution had a smaller variation resulting in a more accurate plateau length measurement. Alignment via these means also meant a more reproducible molecular plateau could be seen on a 2D conductance histogram.

With the development of an ML-enhanced analysis process, and an improved alignment process, continued measurement of the pseudo-rotaxane molecule should be carried out. Repeat experimentation would provide a more insightful view into the conductance behaviour of rotaxanes in general. Additional experimentation should also compare the conductance of the pseudo-rotaxane molecule to the conductance of the non-protected axel molecule to better understand the effect of the interlocking macrocycle. This would provide further insight into the electronic behaviour of rotaxane-like molecules.

To conclude, the use of ML techniques has been shown to yield predictive results that are superior to statistical analysis approaches. Due to this enhanced capability, new and previously challenging SM concepts have been explored with relative ease. As such, the application of ML techniques to data interpretation should be considered by chemical analyses in general.

# Appendix

## Appendix 1 Additional Figures

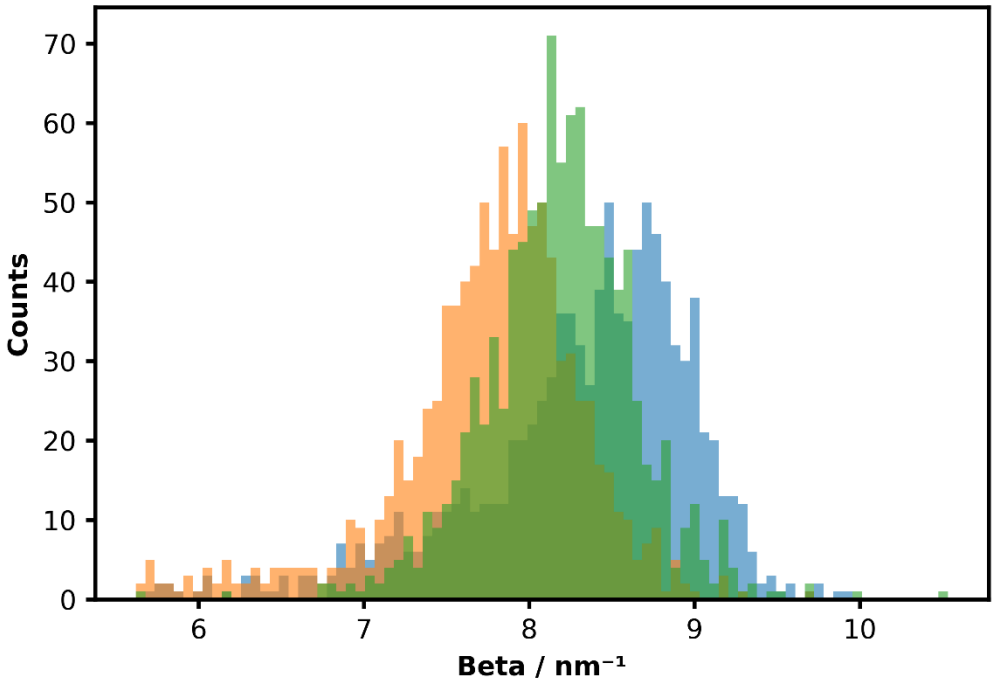



***Figure A1.1*** *Three repeat ensemble measurements of the tunnelling decay constant, $\beta$, as part of I(t) distance calibration.*

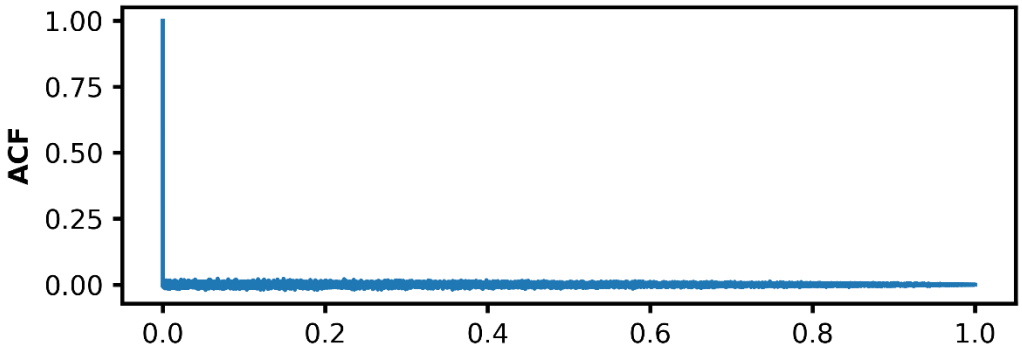



***Figure A1.2*** *Autocorrelation function for a simulated section of background trace. There appears to be no correlation between the trace and itself at any time delay other than 0. This implies the trace is largely random as no patterns have emerged.*

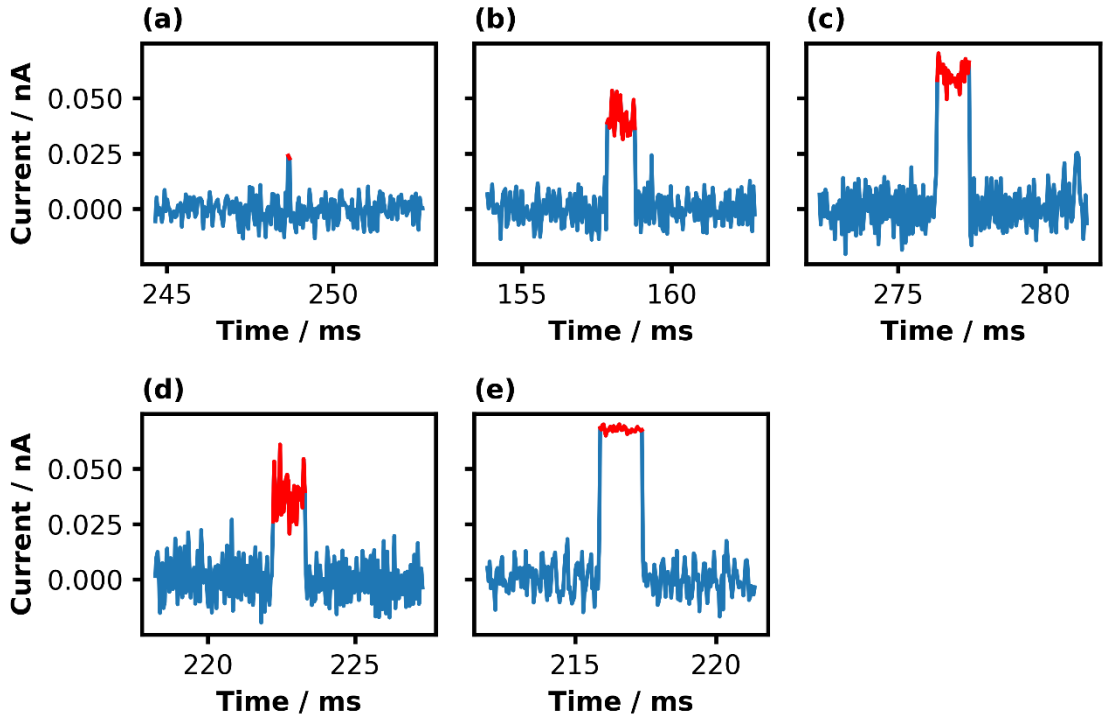



***Figure A1.3*** *Example simulated I(t) events for each class. Classes were labelled as phosphate, (a), rAMP, (b), rTMP, (c), rCMP, (d), and rGMP, (d), respectively.*

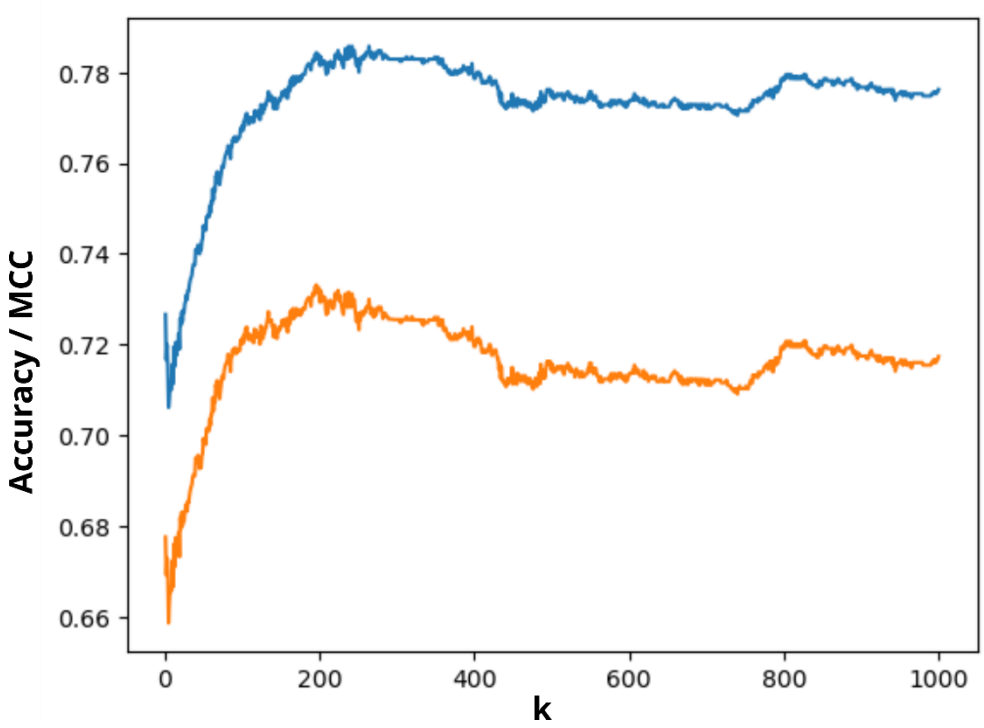



***Figure A1.4*** *Results of the parameter optimization for the KNN classifier. Shown are the accuracies (blue) and the MCC scores (orange) associated with each value for K.*

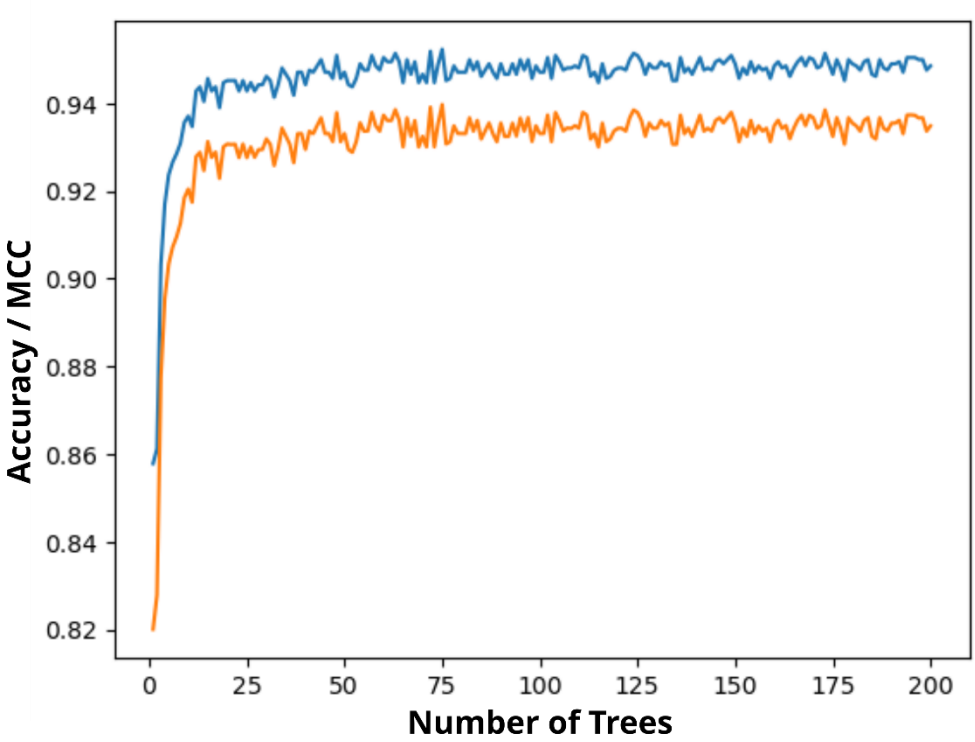



***Figure A1.5*** *Results of the parameter optimization for the RFC. Shown are the accuracies (blue) and the MCC scores (orange) associated with each value for the number of trees within the forest.*

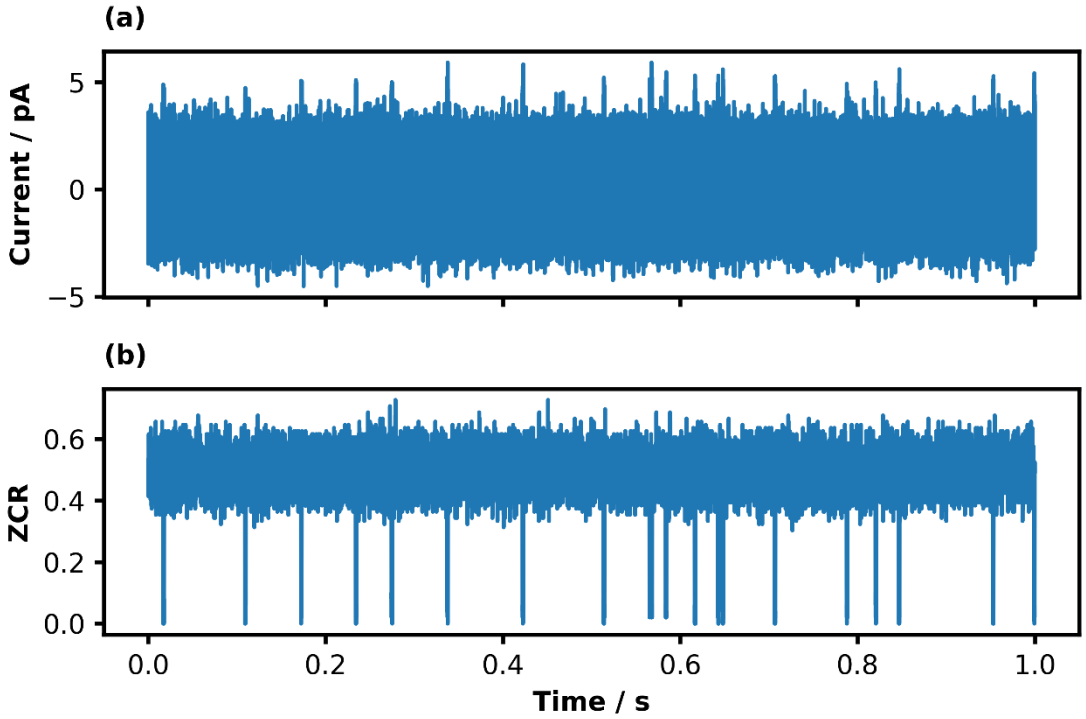



***Figure A1.6*** *Comparison of a simulated I(t) trace with anomalies, (a), to the corresponding trace of window ZCR values, (b), when using the ZCR method with a window size of 100 and a step size of 10.*

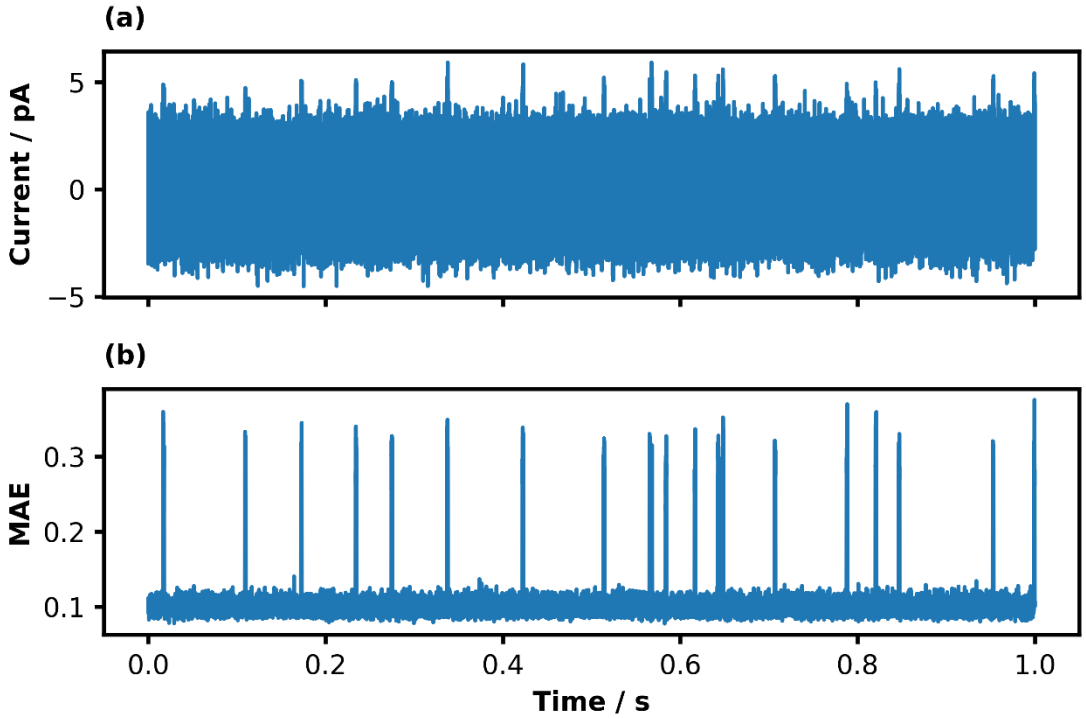



***Figure A1.7*** *Comparison of a simulated I(t) trace with anomalies, (a), to the corresponding trace of window MAE values, (b), when using the AE method with a window size of 100 and a step size of 10.*

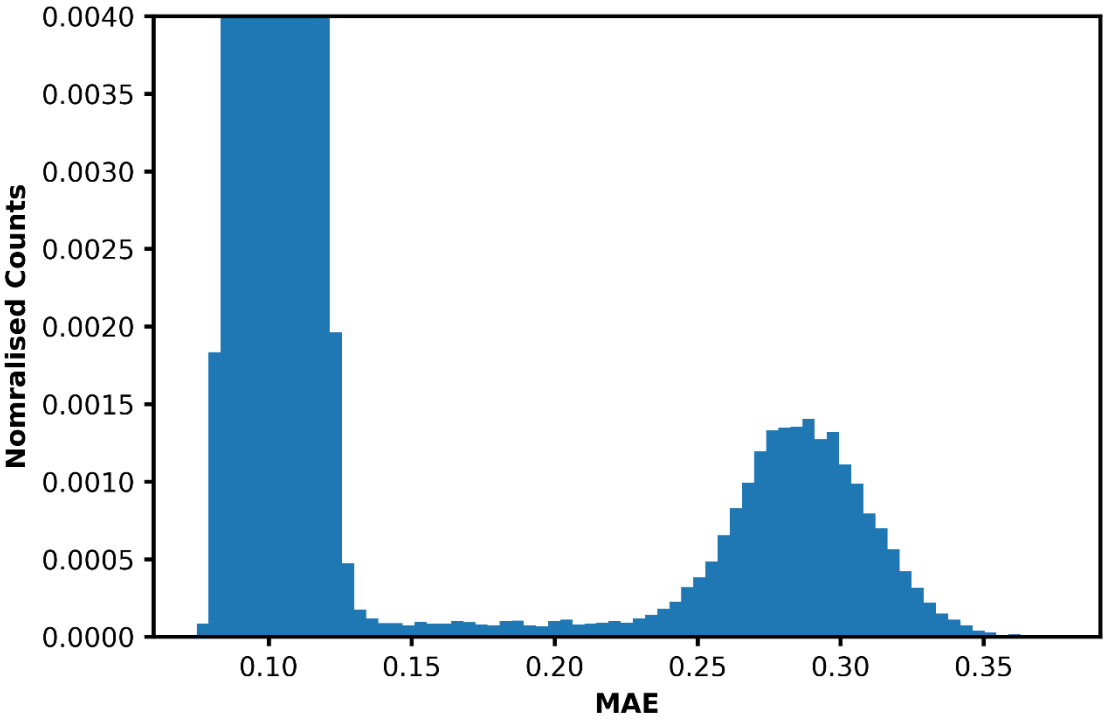



***Figure A1.8*** *Distribution of MAE values calculated from event traces with an SNR of 2. MAEs were calculated with a window size of 100 and a step size of 10. The distribution of values above 0.14 are a result of anomalies that were not present in the background trace.*

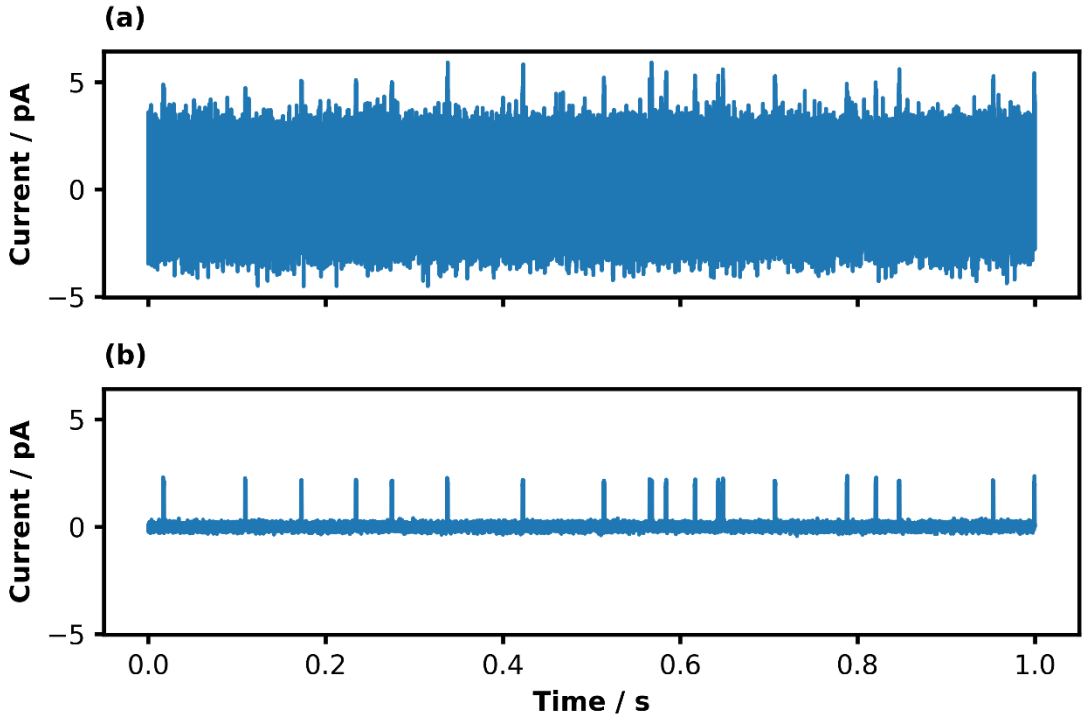



***Figure A1.9*** *Comparison of a simulated I(t) trace with anomalies, (a), to the corresponding trace of window average values, (b), when using the moving average method with a window size of 100 and a step size of 10.*

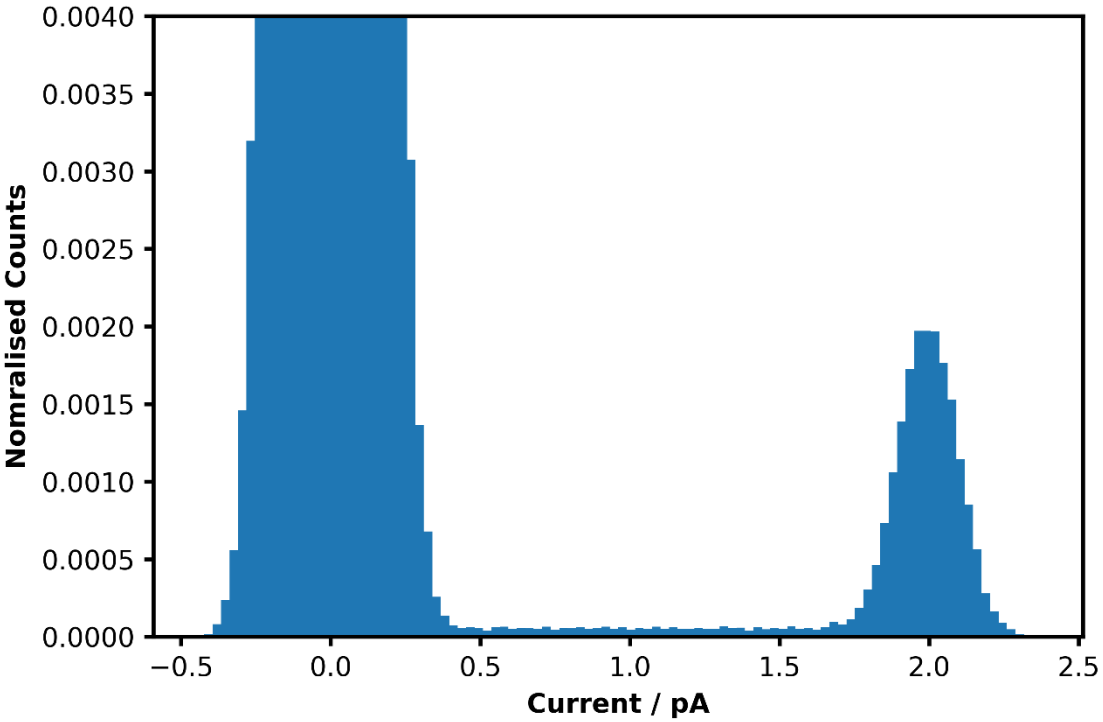



***Figure A1.10*** *Distribution of average values calculated from event traces with an SNR of 2. Averages were calculated with a window size of 100 and a step size of 10. The distribution of values above 0.4 are a result of anomalies that were not present in the background trace.*

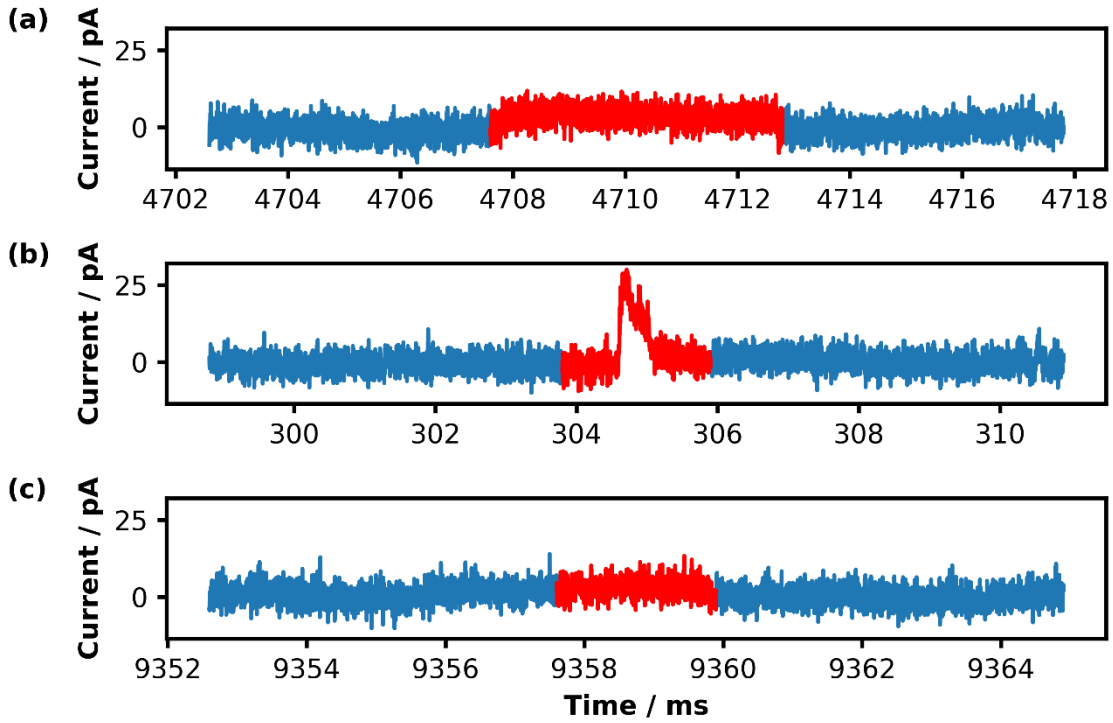



***Figure A1.11*** *Example events found using the moving average method on experimental RPS data. (a) shows an event with a long duration, (b) shows an event with a large height, and (c) shows an event with both short height and duration.*

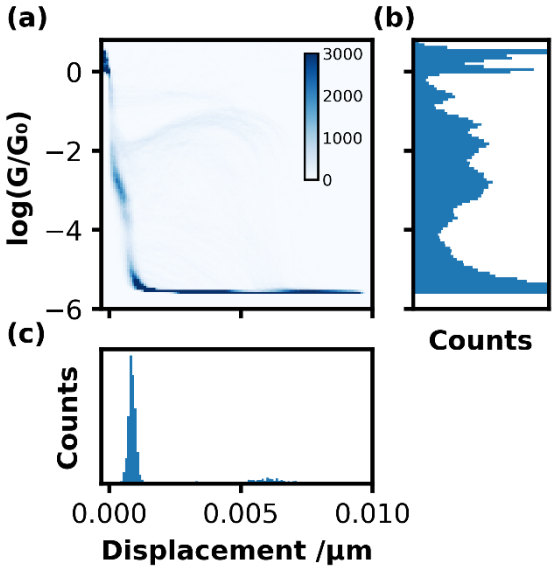



***Figure A1.12*** *Histogram analysis of molecule 2 using uncleaned data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. A total of 2000 traces were sampled.*

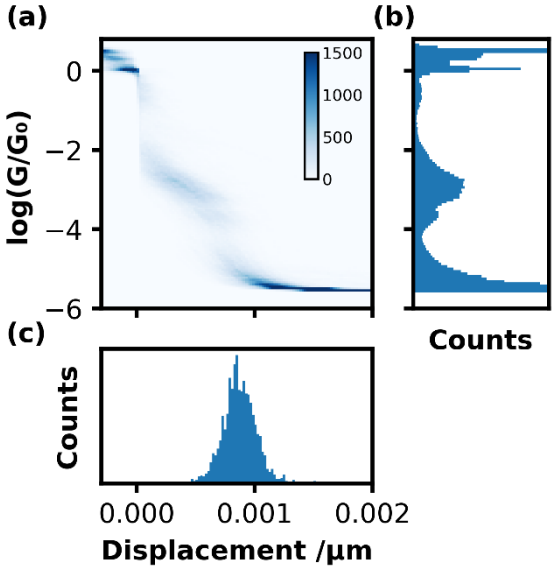



***Figure A1.13*** *Histogram analysis of molecule 2 using filtered data. (a) shows a 2D histogram of conductance vs displacement, (b) shows the 1D conductance histogram, and (c) shows a histogram of trace plateau lengths. After this filtering, 1575 traces remain.*

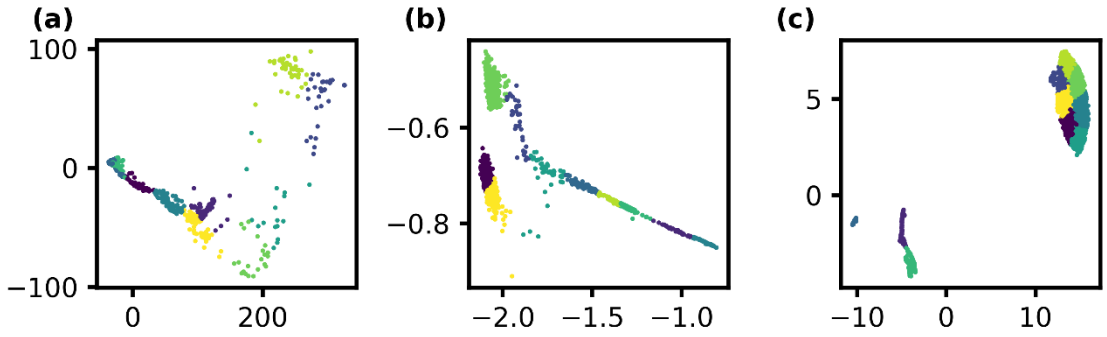



***Figure A1.14*** *2-dimensional embeddings of molecule 2's uncleaned BJ data. Embeddings were produced using PCA, (a), t-SNE, (b), and UMAP, (c). Points are separated into ten clusters using the k-means clustering algorithm and are coloured accordingly.*

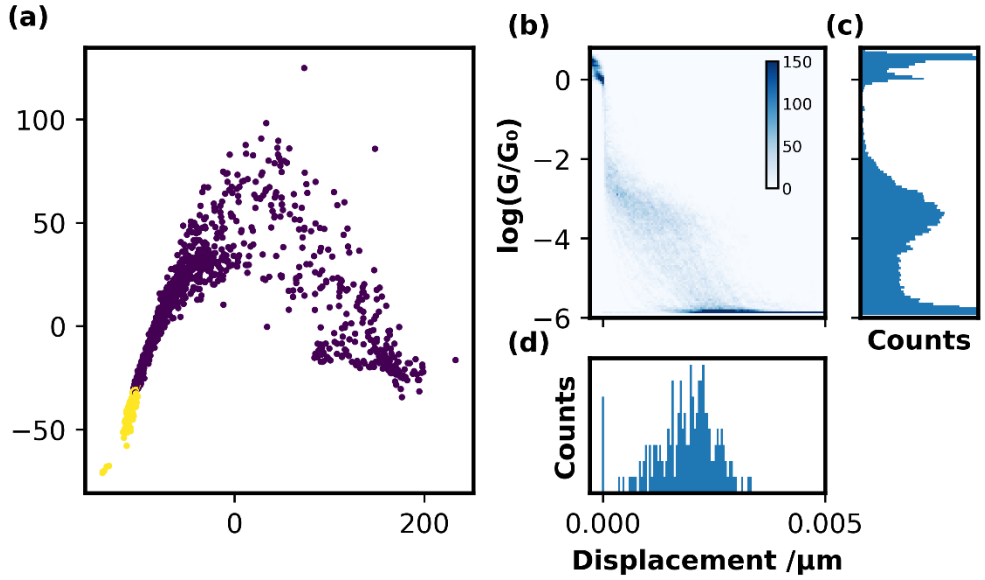



***Figure A1.15*** *Results of data cleaning using PCA embeddings for molecule 1. (a) shows the embedded data space where the 177 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

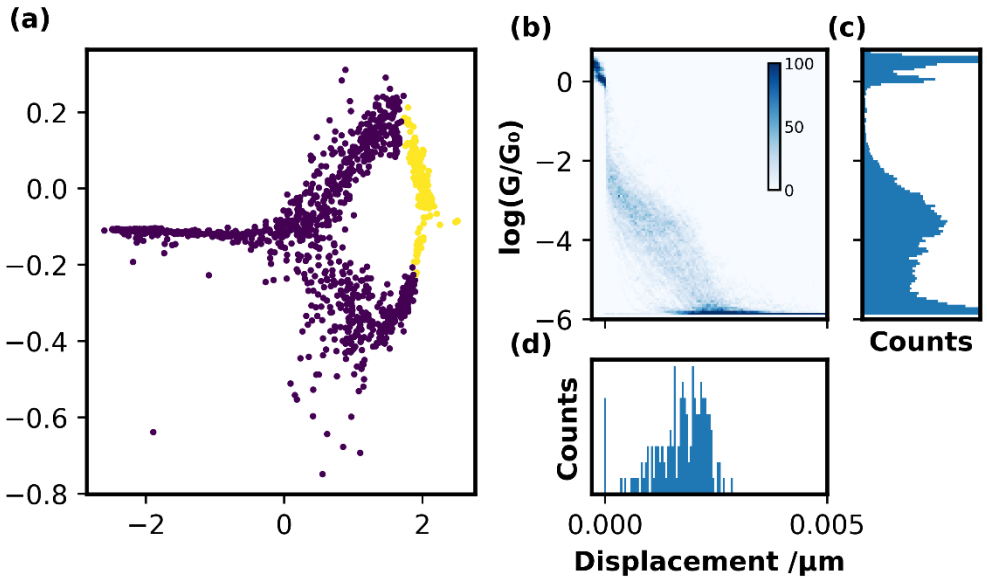



***Figure A1.16*** *Results of data cleaning using t-SNE embeddings for molecule 1. (a) shows the embedded data space where the 153 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

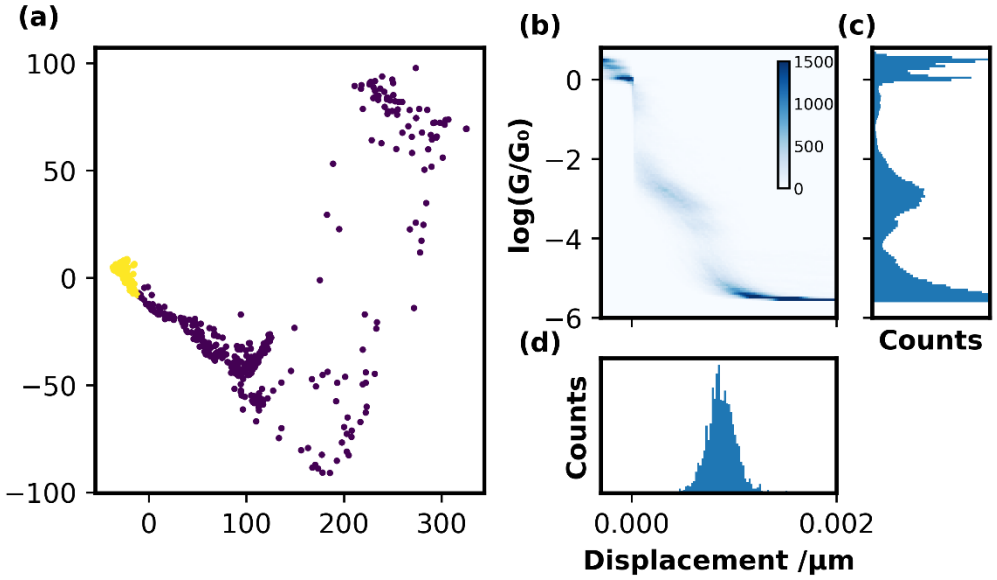



***Figure A1.17*** *Results of data cleaning using PCA embeddings for molecule 2. (a) shows the embedded data space where the 1609 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

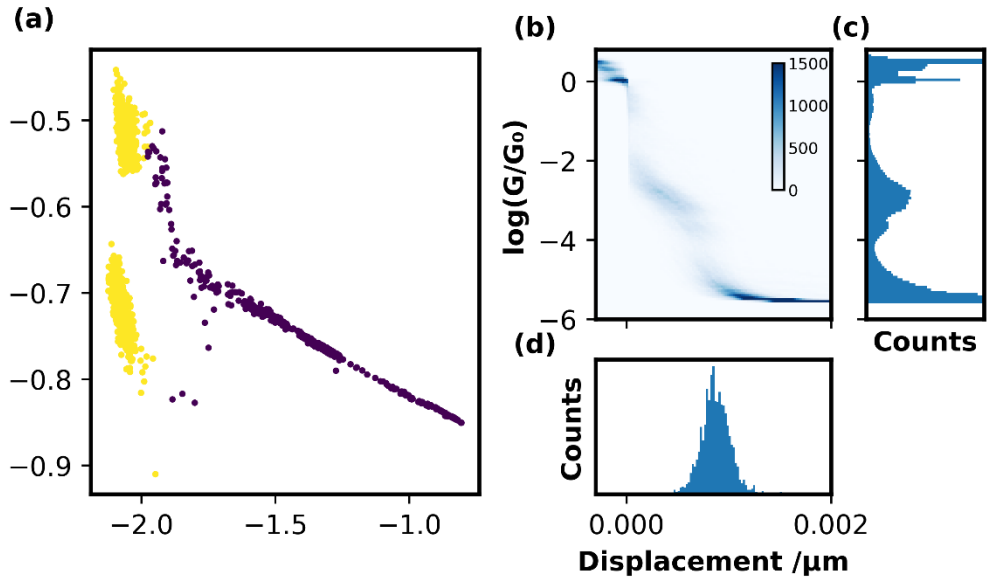



***Figure A1.18*** *Results of data cleaning using t-SNE embeddings for molecule 2. (a) shows the embedded data space where the 1582 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

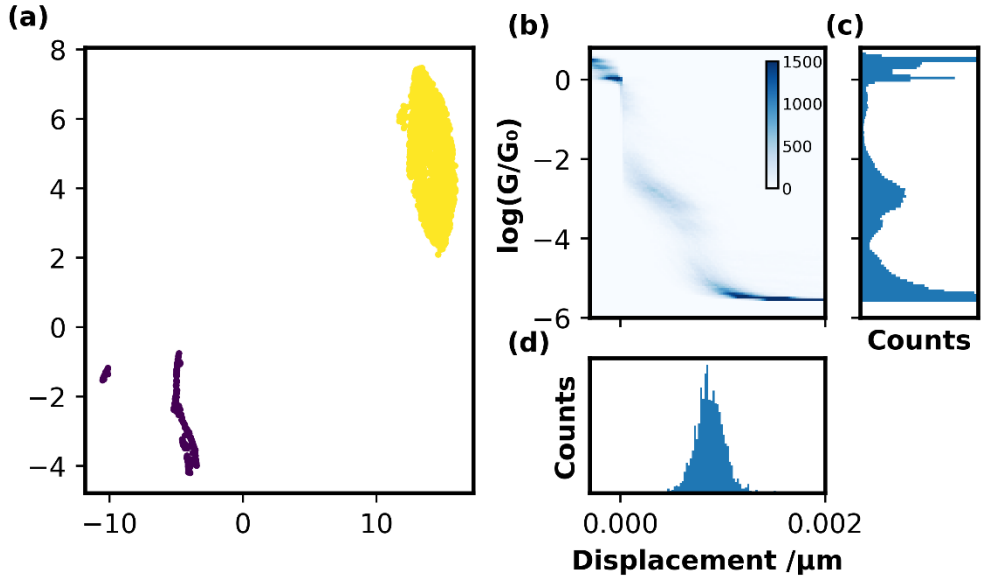



***Figure A1.19*** *Results of data cleaning using UMAP embeddings for molecule 2. (a) shows the embedded data space where the 1612 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

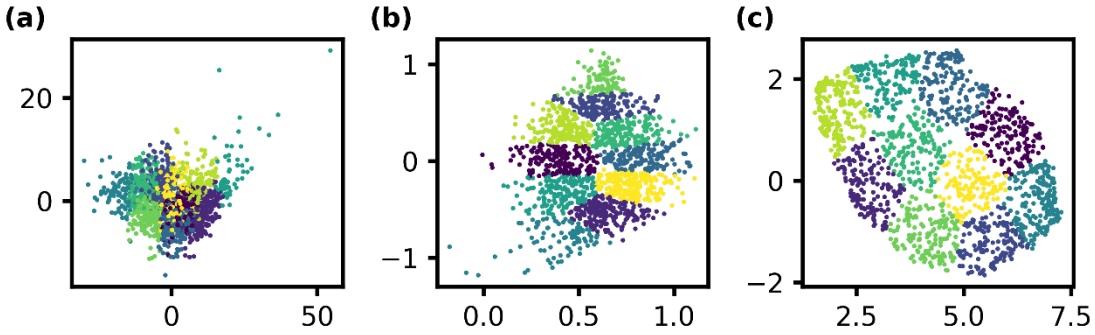



***Figure A1.20*** *2-dimensional embeddings of molecule 2's filtered BJ data. Embeddings were produced using PCA, (a), t-SNE, (b), and UMAP, (c). Points are separated into ten clusters using the k-means clustering algorithm and are coloured accordingly.*

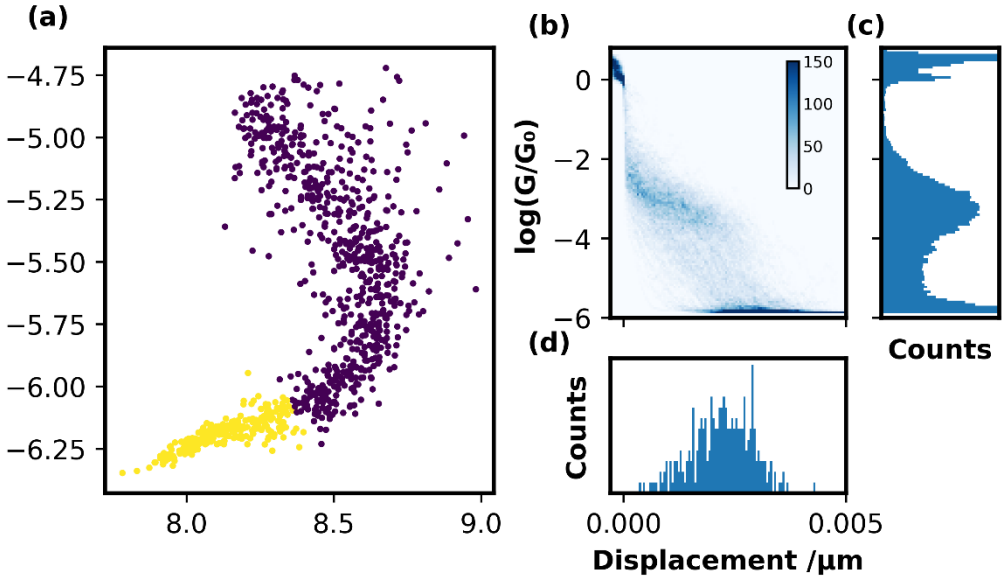



***Figure A1.21*** *Results of data cleaning using t-SNE embeddings on prefiltered data for molecule 1. (a) shows the embedded data space where the 272 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

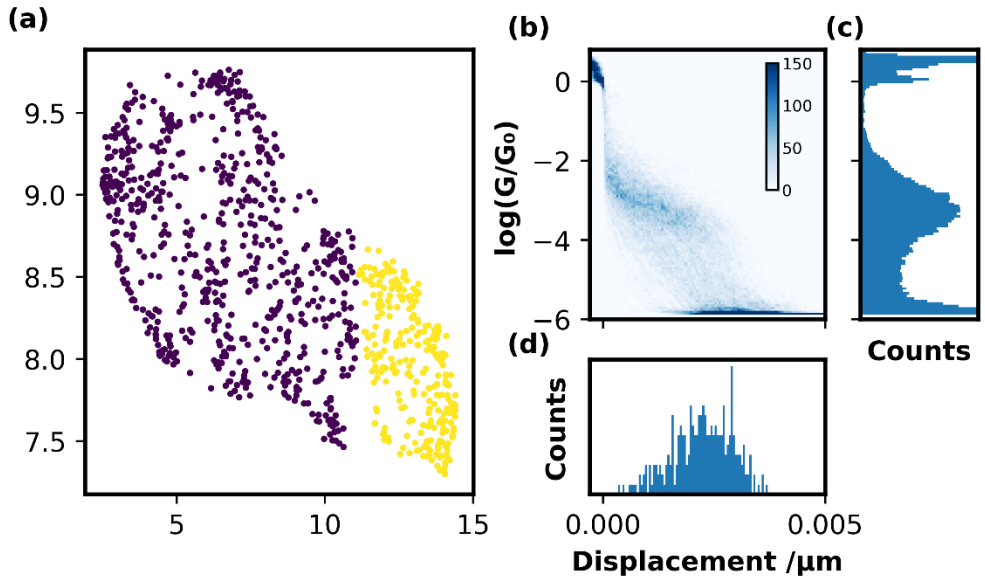



***Figure A1.22*** *Results of data cleaning using UMAP embeddings on prefiltered data for molecule 1. (a) shows the embedded data space where the 274 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

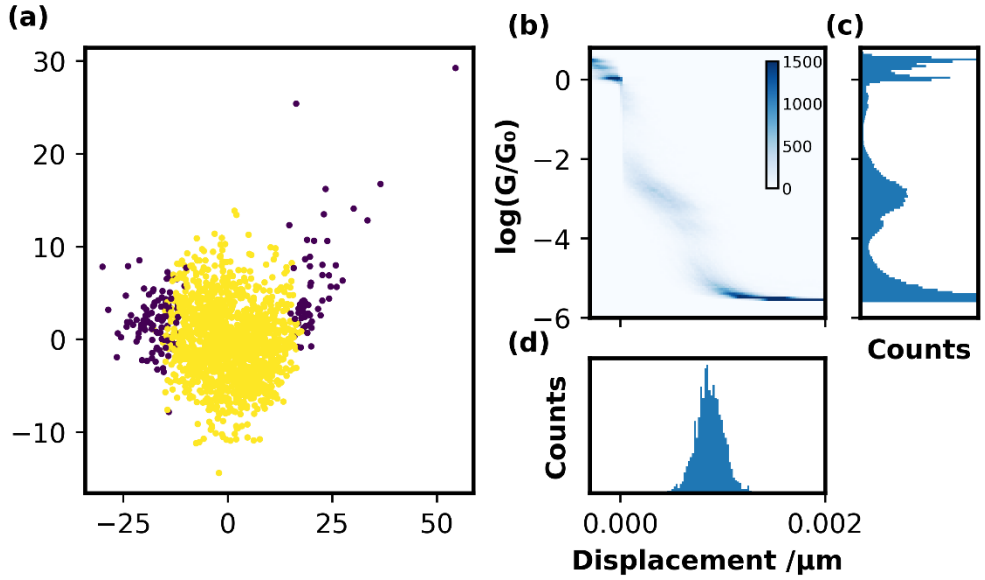



***Figure A1.23*** *Results of data cleaning using PCA embeddings on prefiltered data for molecule 2. (a) shows the embedded data space where the 1409 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

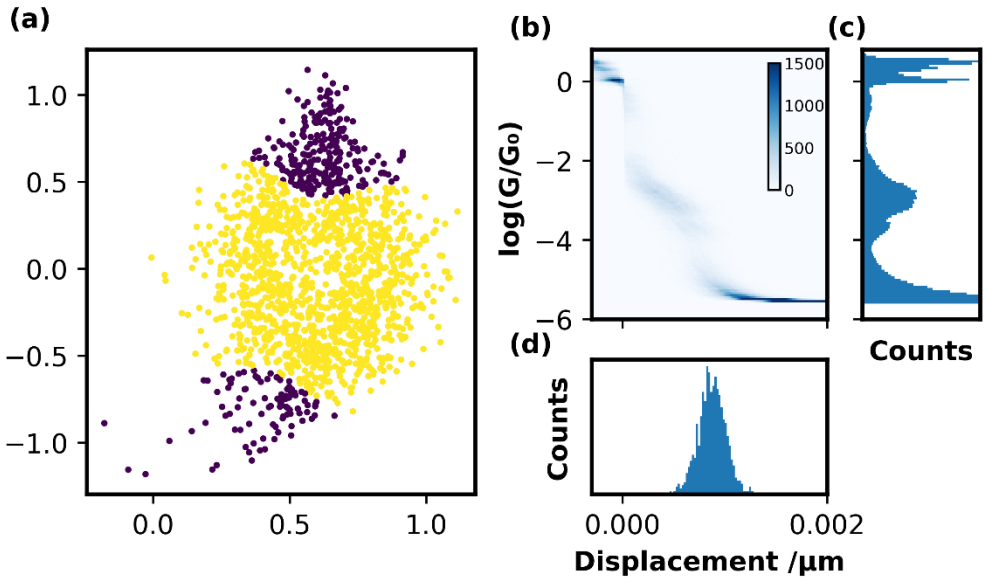



***Figure A1.24*** *Results of data cleaning using t-SNE embeddings on prefiltered data for molecule 2. (a) shows the embedded data space where the 1227 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

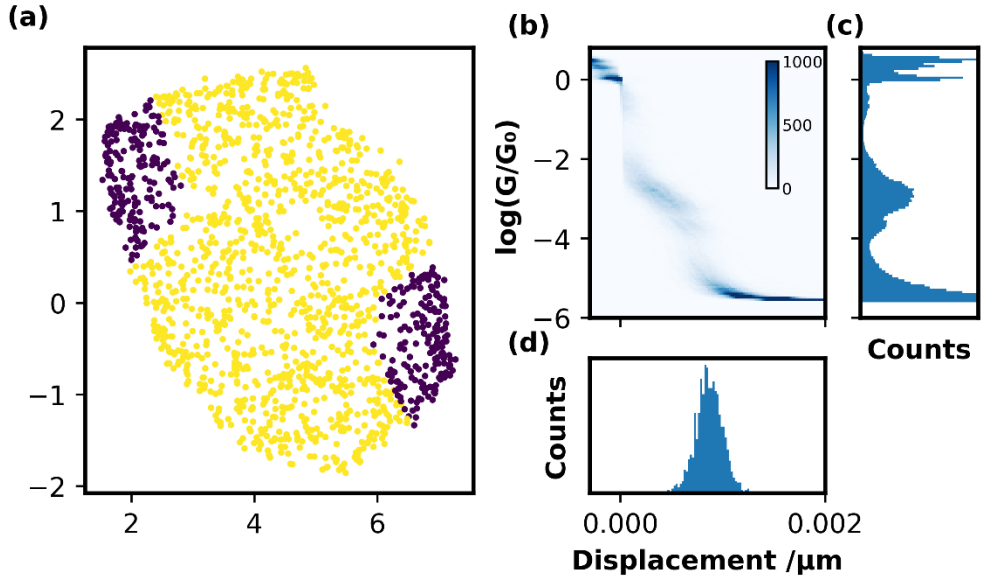



***Figure A1.25*** *Results of data cleaning using UMAP embeddings on prefiltered data for molecule 2. (a) shows the embedded data space where the 1194 highlighted points (yellow) were taken as clean traces. The histogram analysis of these clean traces is shown in (b-d).*

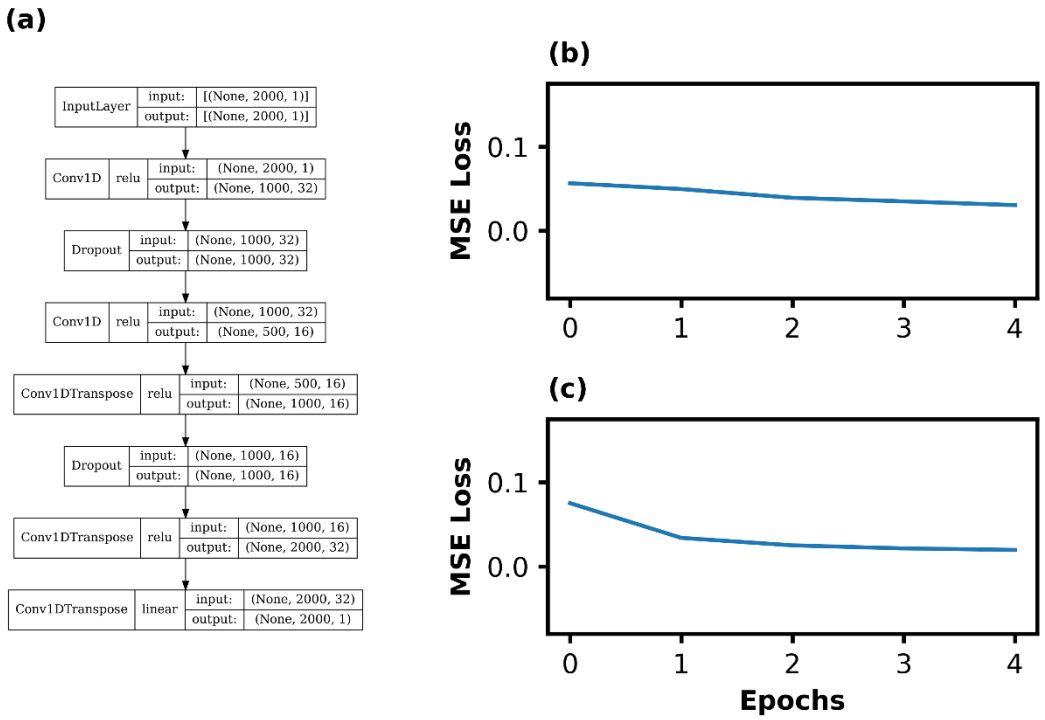



***Figure A2.1*** *Summary of the convolutional AE model with the 2000-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

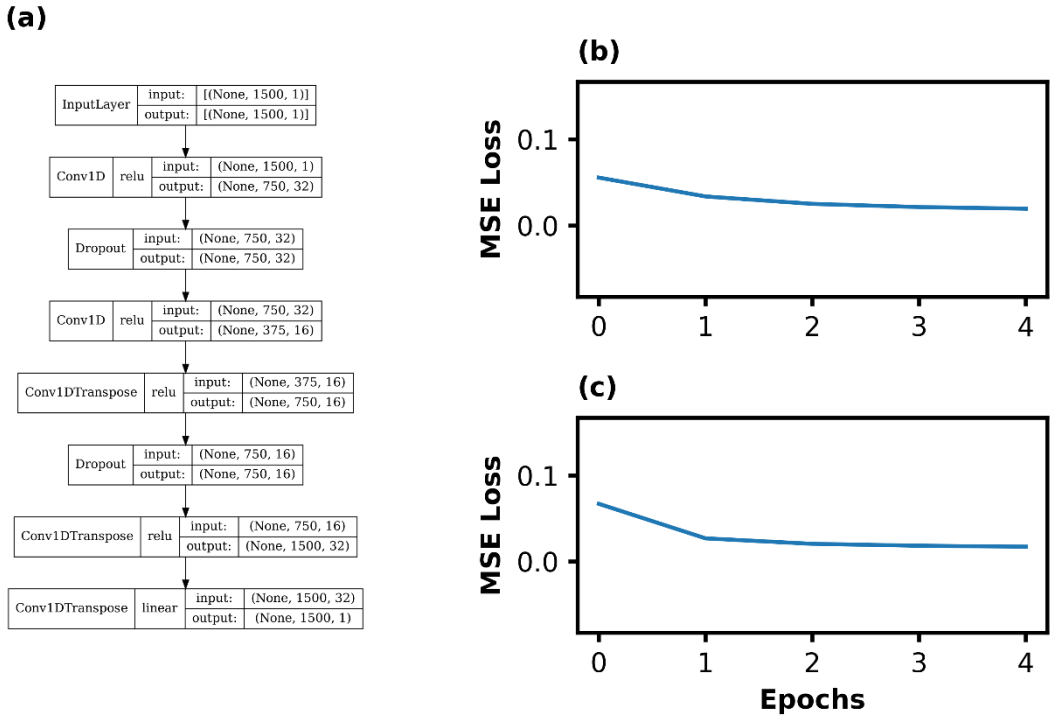



***Figure A2.2*** *Summary of the convolutional AE model with the 1500-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

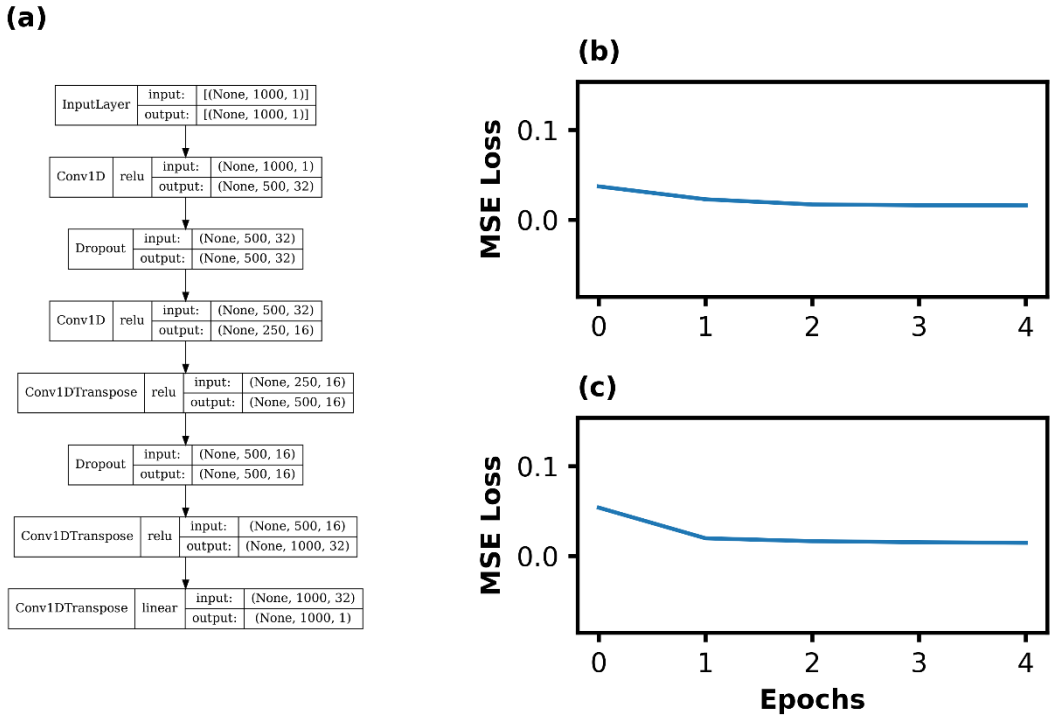



***Figure A2.3*** *Summary of the convolutional AE model with the 1000-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

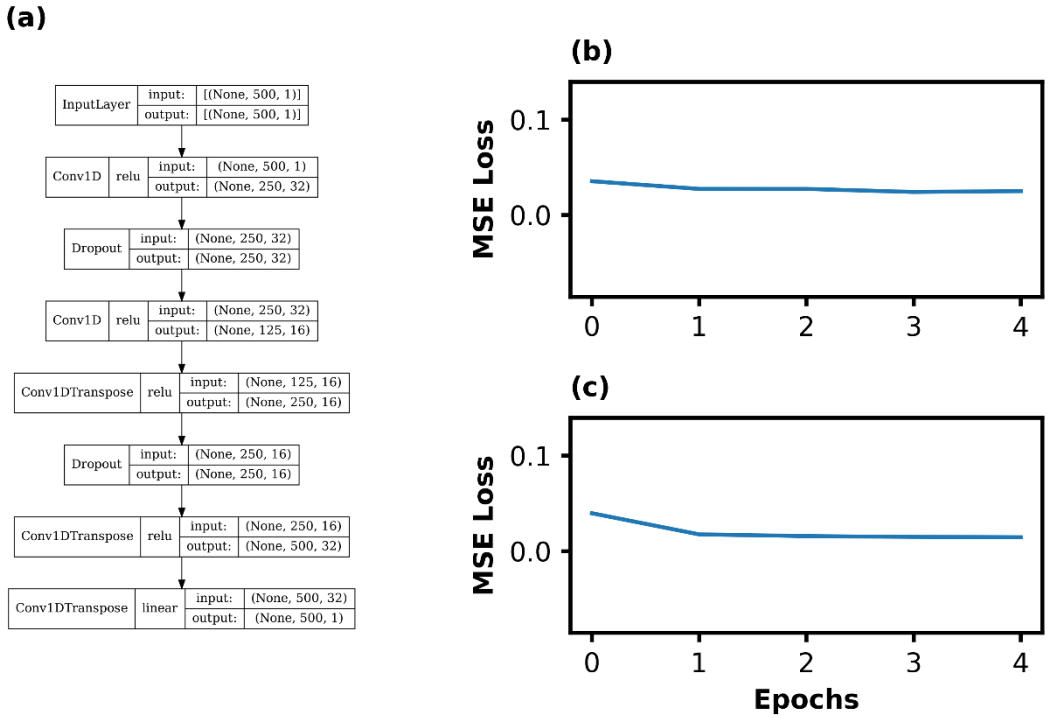



***Figure A2.4*** *Summary of the convolutional AE model with the 500-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

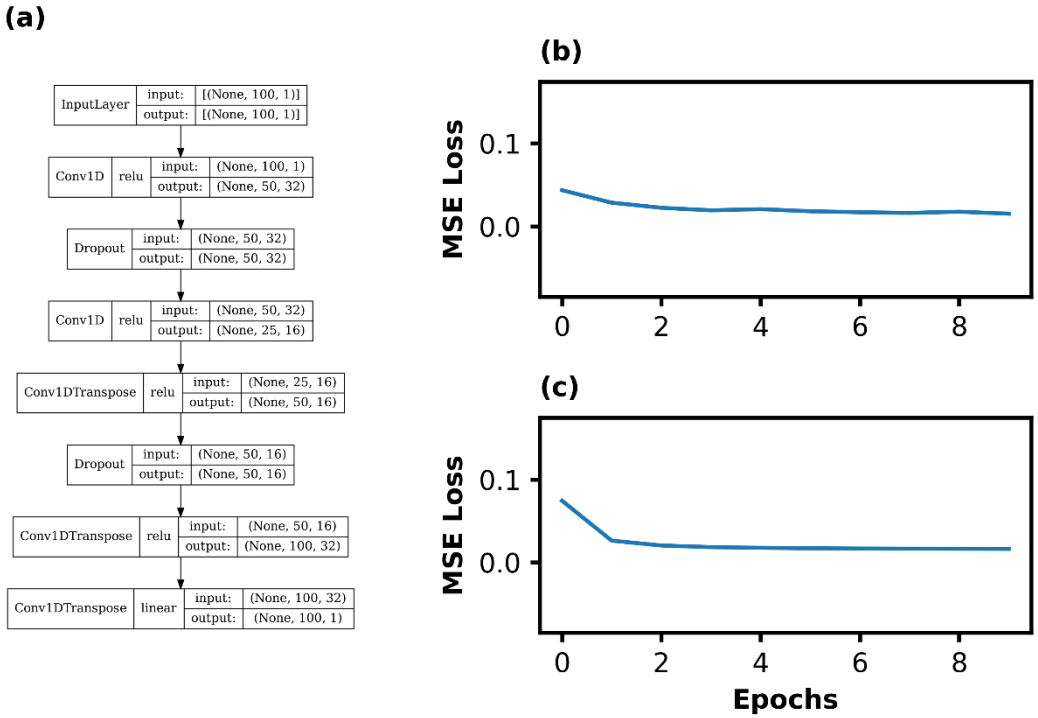



***Figure A2.5*** *Summary of the convolutional AE model with the 100-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

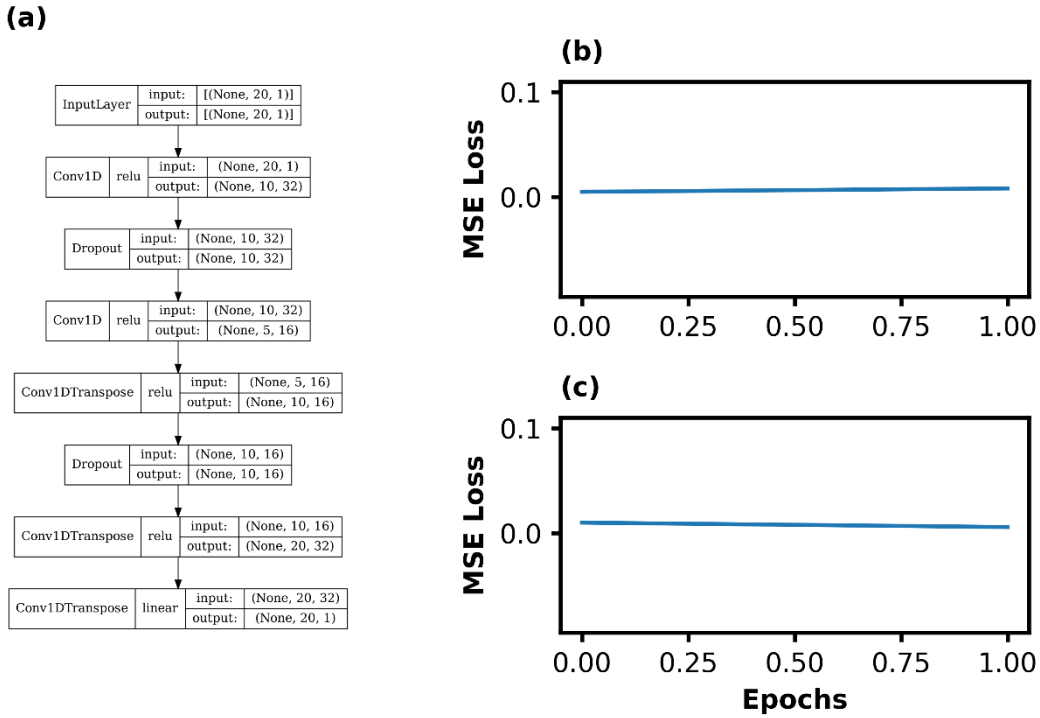



***Figure A2.6*** *Summary of the convolutional AE model with the 20-point window input size. (a) shows the network architecture of layers including activation functions and output dimensionality. (b) shows MSE loss and accuracy on the validation dataset during training. (c) shows the MSE loss and accuracy on the training dataset during training.*

# Appendix 3 Unsupervised Classification of Voltammetric Data Beyond Principal Component Analysis

Here the manuscript and supporting information of the paper “Unsupervised Classification of Voltammetric Data Beyond Principal Component Analysis” has been integrated this thesis. For this collaborative piece of work, the data analysis and figures were the main contribution of myself with additional contributions being made to manuscript writing and analysis pipeline development.

ChemComm

COMMUNICATION

View Article Online
View Journal | View Issue



Cite this: *Chem. Commun.*, 2022, 58, 10170

Received 6th June 2022,
Accepted 17th August 2022

DOI: 10.1039/d2cc03187f

rsc.li/chemcomm

# Unsupervised classification of voltammetric data beyond principal component analysis†

Christopher Weaver, [a] Adrian C. Fortuin, [ab] Anton Vladyka [a] and Tim Albrecht *[a]

**In this study, we evaluate different apoproaches to unsupervised classification of cyclic voltammetric data, including Principal Component Analysis (PCA), t-distributed Stochastic Neighbour Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP) as well as neural networks. To this end, we exploit a form of transfer learning, based on feature extraction in an image recognition network, VGG-16, in combination with PCA, t-SNE or UMAP. Overall, we find that t-SNE performs best when applied directly to numerical data (noise-free case) or to features (in the presence of noise), followed by UMAP and then PCA.**

Voltammetric data contain a wealth of mechanistic, kinetic, and analytical information about the system under study and are routinely used in a wide range of applications, including for the basic characterisation of redox-active materials,[1] surface characterisation[2] and the study of electrocatalytic processes.[3] In some cases, the amount of available data is small and their analysis can readily be performed 'by hand'. Typically, this information is extracted based on theoretical models, for example in relation to the dependence of peak currents on analyte concentration or scan rate, to mass transport and the effect of electrode geometry or the overall shape of the voltammetric signal.[4,5] Deviations from model behaviour can significantly enhance the complexity of the task, but may be addressed using numerical modelling or calibration.[6]

In other applications, however, data are recorded in an (semi-) automated way and are then much more abundant. Examples include high-throughput screening,[7] autonomous sensing and quality control,[8] and electrochemical surface imaging.[9] This calls for automated analysis methodologies, which is relatively straightforward, if the system behaviour is well-understood, robust and well-defined. In such cases, specific observables, such as the current at given potential or the peak current, may be used to extract the quantity of interest.

However, the analysis task becomes significantly more challenging, if this is not the case. For example, the data may reflect a mixture of different electrochemical processes, be recorded under varying geometrical conditions or be affected by device failure or contamination.[10,11] In such cases, unsupervised dimensionality reduction techniques such as PCA have been employed, which to some extent consider the overall appearance of the data.[12–15] Beyond PCA, there are however other, potentially superior dimensionality reduction techniques, such as non-linear, stochastic methods, which aim to reproduce neighbourhood relations in high-dimensional data space in a lower dimensional representation. Those have not found widespread application in electrochemistry yet and we will therefore explore two examples, namely t-SNE[16,17] and UMAP,[18,19] and benchmark those against an implementation of linear PCA. Supervised methodologies, which do require labelled training data, have been introduced to the field recently and show promise for some applications.[20–23] However, they will not be in focus here. In addition to the three dimensionality reduction techniques mentioned above, we will also consider three different data input formats and evaluate their effect on the classification performance. These are raw numerical data (*i.e.* (scaled) current-potential value pairs), b/w images of the CVs as well as the feature extractor output of an image recognition network, VGG-16, as illustrated in Fig. 1.[24,25] The latter is based on the idea that such networks are able to identify salient features in physico-chemical data, despite initially being trained on unrelated image data of everyday objects. We recently demonstrated this approach for single-molecule charge transport data using another image recognition network, AlexNet,[24,26] and were able to identify previously undetected sub-populations in the data. Interestingly, while this has elements of transfer learning or Artificial General Intelligence, the fact that the feature extractor does not require re-training also demonstrates that such Deep Learning architectures can be employed successfully in the absence of large amounts of, or indeed any, domain-specific data.

[a] *School of Chemistry, University of Birmingham, Edgbaston Campus, Birmingham B15 2TT, UK. E-mail: t.albrecht@bham.ac.uk*
[b] *Faculty of Mechanical Engineering, Helmut Schmidt University, 22043 Hamburg, Germany*

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d2cc03187f

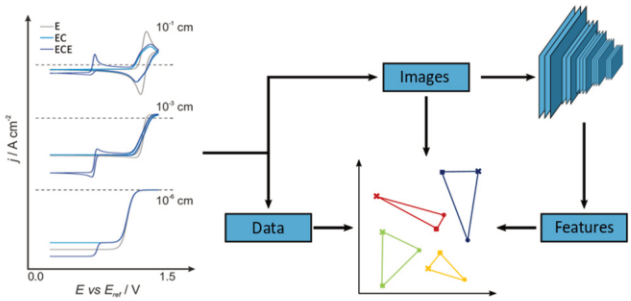



Fig. 1 Schematic of the unsupervised classification process (right) alongside examples of simulated CVs (left). CVs were simulated for 'E' (grey), 'EC' (light blue), and 'ECE' (dark blue) reaction mechanisms at varying electrode radii. In general, CVs are pre-processed in three formats: images; features; and data. The data and image sets are created by taking the datapoint or converting graphs into images, respectively. The Features dataset is created by utilising a pretrained VGG-16 CNN feature extractor. See ESI,† and main text for more details on the simulation and analysis pipeline.

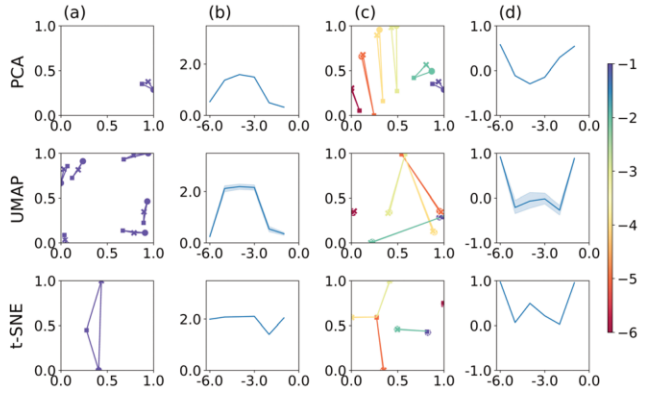



Fig. 2 PCA, UMAP, and t-SNE applied to raw numerical data (a) 2D projections of CVs at the $10^{-1}$ cm electrode radius showing six repeats of the dimensionality reduction process. Reducers were optimised for the perimeter metric. (b) Average perimeter scores at each electrode radius (log scale) when perimeter optimised. (c) 2D projections of one repeat of CVs for all electrode radii. Reducers were optimised for the silhouette score. The points hues in (a and c) are determined by the radius colour bar (right). (d) Average silhouette scores at each electrode radius when optimised for silhouette score.

In order to establish an unequivocal "ground truth", we use simulated data (Digisim® v3.0), in some cases with added noise (*vide infra*). A total of 18 CVs was generated for three different, classic electrochemical reaction mechanisms, namely electron transfer (E), electron transfer coupled with a homogeneous chemical reaction (EC) and a sequence of electron transfer/homogeneous chemical reaction/electron transfer (ECE), for electrode radii ranging from $10^{-1}$ to $10^{-6}$ cm, Fig. 1. This choice is to some extent arbitrary and mainly served to generate well-defined, distinct cyclovoltammetric responses. However, the specific thermodynamic and kinetic model parameters were taken from a well-known and well-characterised example, namely the electrochemical oxidation of aniline to polyaniline, see Sections S1 and S2 in the ESI† for further details.[27,28]

The raw numerical data values were scaled between −1 and 1 (comparable to the image data), in order to emphasize shape, rather than the magnitude of the current, and to facilitate a comparison with the analysis of other input formats used in this study. In practical applications, the electrode radius is normally given, but changes in appearance could conceivably have other origins.

After dimensionality reduction, each CV was represented in a two-dimensional projection, with each E/EC/ECE triplet spanning a triangle of perimeter $P$, where larger $P$ scores correspond to better separation between the three mechanisms (each component scaled from 0 to 1, so $0 \leq P \leq 2 + \sqrt{2}$; over all 18 CVs). In addition, to quantify the separation of triplets, we used the mean silhouette value of each triplet, as defined in eqn (1):

$$S = \frac{b - a}{\max(a, b)} \quad (1)$$

where $a$ is the average intra-cluster distance of a cluster and $b$ the average nearest-cluster distance over all samples. Hence, $S$ scales from −1 to +1, where +1 indicates perfect separation of the triplet clusters (all points assigned correctly).

Considering dimensionality reduction applied directly to the raw (value pair) data first, the different metrics are summarised in Fig. 2. Column (a) shows the value triplet for $r = 1 \times 10^{-1}$ cm, for PCA (top), UMAP (centre) and t-SNE (bottom), see Fig. 1. For this particular radius, PCA produces a point triplet that is roughly equally separated, even if the perimeter score is relatively small. It is larger for other electrode radii, column (b), and reaches a maximum for $r = 1 \times 10^{-4}$ cm. Under these conditions, the E and EC mechanisms produce very similar CVs, which is reflected in a rather large, but irregular triangle in the reduced dimensional representation, column (c). Based on the $S$ score and an optimised set of hyperparameters, column (d), separation by electrode radius works best for the small and large radii, less so for the intermediate range. This is in part a manifestation of the interdependence of the perimeter and silhouette score metrics, as large perimeter values are more likely to result in overlap between adjacent point triplets and hence reduced silhouette values. However, we have used this approach to facilitate the comparison across the entire dataset as well as between the different dimensionality reduction techniques. Notably, when optimised for maximum perimeter score, UMAP and t-SNE produce larger $P$ scores, compared to PCA, suggesting that those two techniques provide better separation between the three mechanisms. However, to account for the stochastic nature of both UMAP and t-SNE, where the outcome can show some variation from run to run, we show averages of $P$ and $S$, as well as associated 95% confidence intervals for UMAP and t-SNE (20 repeats each), as shown in Fig. 2(b) and (d). In terms of $P$ score and across all electrode radii, UMAP broadly follows the trend observed in the PCA results, while t-SNE appears to show a more consistent performance throughout.

The UMAP and t-SNE results shown in columns (c) and (d) are optimised for maximum $S$ score. Normalised over the entire dataset, it becomes apparent that some ability to differentiate between mechanisms is lost (small $P$ scores for large $r$, for example), but that those triplets are then well-separated from the others (relatively high $S$ score). Conversely, at intermediate radii, separation by mechanism is still satisfactory (relatively

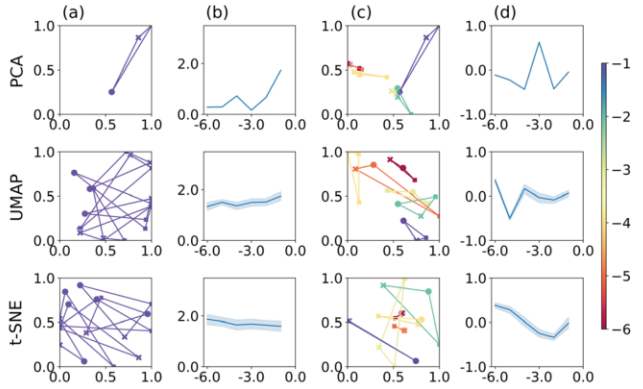



Fig. 3 PCA, UMAP, and t-SNE applied to b/w image data. Plots are arranged in the same way as in Fig. 2.

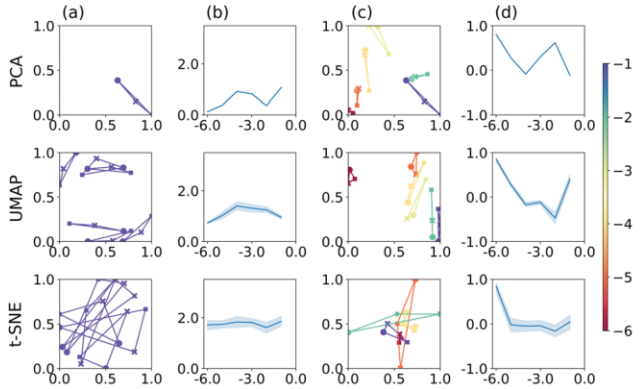



Fig. 4 PCA, UMAP, and t-SNE applied to the feature extractor output of VGG-16. Plots are arranged as described in Fig. 2.

large triangles), but separation between electrode radii is decreased (triplet overlap). In any case, hyperparameter optimisation clearly has an effect on the result of the classification.

We now compare the performance of PCA, UMAP and t-SNE when applied to the CVs, presented as image data. The results are summarised in Fig. 3. As before, we focus on a specific electrode radius in column (a), namely $r = 1 \times 10^{-1}$ cm, for illustration. For PCA (top), the corresponding triangle is well-defined, relatively large and produces the largest $P$ score within this series of electrode radii, as shown in column (b). Notably, the corresponding triangle is markedly irregular, with E and EC mechanisms relatively close and ECE further away, in line with a visual inspection of the actual CVs, *cf.* ESI† Section S4. The $P$ score appears to be larger than for PCA applied to the raw data, Fig. 2. Furthermore, the separation of point triplets, based on the $S$ score and optimised hyperparameters, columns (c)–(d), is not particularly successful with $S$ values close to 0 throughout, except for $r = 1 \times 10^{-3}$ cm.

For UMAP and t-SNE, the stochasticity of the outcome is clearly apparent in column (a), centre and bottom, respectively. Both algorithms produce good separation of the three mechanisms for this electrode radius, but the orientation and to some extent size of the triangle ($P$ score) varies, as noted above. Across all electrode radii, their performance is however rather consistent (close to $P = 2$) and better than for PCA on images. Following hyperparameter optimisation on triplet separation, the latter appears to work best for the smallest electrode radius used here; otherwise the $S$ scores remain close to 0, indicating some overlap between the triplet clusters, columns (c)–(d). Under these conditions, UMAP still appears to be better than t-SNE, with regards to separating the different mechanisms, as for t-SNE the E and EC mechanisms largely seem close together, except for $r = 1 \times 10^{-2}$ cm (light green triangle in column (c), corresponding $P$ scores not shown).

Finally, we compare PCA, t-SNE and UMAP applied to the feature extractor output of VGG-16. The results are shown in Fig. 4, following the same format as in Fig. 2 and 3. To get an impression of how the feature extractor "sees" the CV image data, we refer the reader to Section S4 in the ESI,† which show examples of how the individual images are represented in the filter output of the feature extractor (before flattening into a 1D output vector). Bright areas are strongly represented and, by implication, have a larger effect on the classification result. These appear to be, by and large, curved regions of the CV or inflection points, rather than the redox-inactive regions at low potentials, for example, and suggests that VGG-16 indeed identifies features that are related to the electrochemical process, rather than the overall appearance of the image.

Application of PCA, UMAP and t-SNE to the feature extractor output leads to qualitatively similar results, compared to using images directly, both for the separation of individual triplets, column (a), and across the range of electrode radii, column (b). Specifically, PCA appears to perform marginally better, UMAP somewhat worse and t-SNE comparably well and still best comparing the three methods. In terms of the $S$ scores, PCA produces reasonably well separated point triplets, column (c), and, over all electrode radii, a better separation than when applied to image data directly, column (d). The $S$ score reaches values close to 1 for some $r$-values, even though there does not appear to be an overall trend. Similarly, for UMAP, triplet separation is satisfactory and $S$ scores are on average larger than when applied to image data directly. Finally, t-SNE does not separate the value triplets well with some overlap remaining and $S$ scores close to 0, for all but one radius ($r = 1 \times 10^{-1}$ cm). In other words, t-SNE still reproduces the neighbourhood relation according to mechanism better than electrode radius, leading to good separation within a point triplet, but relatively poor differentiation between them.

Based on these detailed comparisons, the question arises which methodology and data input format results in the best overall performance, in terms of the respective optimisation target (mechanism *vs.* electrode radius). Hence, we show the average $P$- and $S$ scores for dimensionality reduction applied to numerical input data (blue), images (orange) and feature extractor output (green), in Fig. 5. See Fig. S10 in the ESI,† for further information.

In terms of the $P$ score, a fairly consistent picture emerges, namely that all three dimensionality reduction techniques work equally well on the different input formats. PCA and t-SNE appear to work slightly better on numerical input data, while UMAP produced the best result when directly applied to images. Importantly, however, both UMAP and t-SNE clearly outperform PCA in this metric and while PCA has the advantage of being

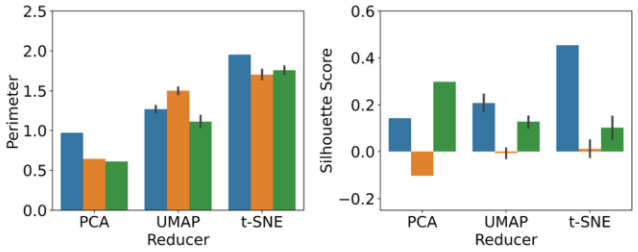



Fig. 5 Comparison of the overall classification results for PCA, UMAP and t-SNE, based on average *P*- and *S* scores, applied to: numerical data (blue), image data (orange); and feature extractor output (green).

deterministic, not requiring any hyperparameter optimisation and being computationally inexpensive, once optimised, both UMAP and t-SNE feature improved separation performance. Using the feature extractor output did not offer any advantage in this case, but it is nevertheless notable that at least comparable results have been obtained, given that the feature extractor had been trained on unrelated image data and the electrochemically relevant features highlighted in the filter outputs (see ESI†).

With regards to the *S* score, the picture is rather less clear cut. The best performing methodology here was t-SNE applied to raw numerical data, followed by PCA on features and UMAP on numerical data. Thus, feature extraction appeared to have a significant benefit in this context, even though further analysis may be required to investigate this effect in more detail.

Finally, we also investigated the effect of small to moderate amounts of noise on the classification, see Section S6 in the ESI.† Intuitively, one might expect the differentiation of CV shapes to become more difficult, as there is more likely going to be more overlap between the CV traces, *cf.* Fig. S11 (ESI†). This expectation is indeed borne out with regards to the separation of the three mechanisms, *i.e.* the average *P* score performance, Fig. S12 (left column) (ESI†), where the overall sequence t-SNE > UMAP > PCA is maintained, but *P* values decrease with increasing amounts of noise (without re-optimising the hyperparameters). For the optimisation towards highest *S* scores, Fig. S12 (right column) (ESI†) the picture is however more complex. Applied to raw data, the performance of UMAP and PCA remains more or less unchanged as the amount of noise is increased, but interestingly t-SNE does significantly worse when even small amounts of noise are added. When applied to image data, the separation performance slightly increases with increasing noise levels with t-SNE and UMAP performing marginally better than PCA in all cases. When applied to features, t-SNE becomes the best performing method in the presence of noise, followed by PCA and then UMAP. Identifying the origin of some of these effects requires further study and will form part of our future work.

Overall, among the methodologies investigated here and considering both *P*- and *S* score performance, in the absence of noise the best performing one appears to be t-SNE applied to raw numerical data. In the presence of noise, t-SNE on features is preferable, followed by UMAP applied to raw data. PCA shows satisfactory performance under all the conditions studied and compared to the other two does not require hyperparameter optimisation. It is however outperformed by some of the other methodologies, highlighting the necessity to carefully consider the dimensionality reduction technique as well as the data input format for a given classification task.

TA designed the study and supervised the work. CW performed the data analysis and co-developed the analysis pipeline. AF produced the electrochemical simulations. AV performed initial tests of approach. All authors contributed to the writing the manuscript.

## Conflicts of interest

There are no conflicts to declare.

## Notes and references

1 C. N. Elgrishi, K. J. Rountree, B. D. McCarthy, E. S. Rountree, T. T. Eisenhart and J. L. Dempsey, *J. Chem. Educ.*, 2018, **95**, 197–206.
2 P. Stonehart and P. N. Ross, *Catal. Rev.*, 1975, **12**, 1–35.
3 P. N. Ross and J. Lipkoswki, *"Electrocatalysis", in Frontiers of Electrochemistry*, Wiley-VCH, 1998.
4 P. Kissinger and W. R. Heineman, *Laboratory Techniques in Electroanalytical Chemistry*, Marcel Dekker, Inc., New York, 1996.
5 R. G. Compton and C. E. Banks, *Understanding Voltammetry*, Imperial College Press, 2010.
6 E. J. F. Dickinson, H. Ekstrom and E. Fontes, *Electrochem. Comm.*, 2014, **40**, 71–74.
7 D. Godfrey, J. H. Bannock, O. Kuzmina, T. Welton and T. Albrecht, *Green Chem.*, 2016, **18**, 1930–1937.
8 M. Sylvain, F. Lehoux, S. Morency, F. Faucher, E. Bharucha, D. M. Tremblay, F. Raymond, D. Sarrazin, S. Moineau, M. Allard, J. Corbeil, Y. Messaddeq and B. Gosselin, *IEEE Trans. Biomed. Circ. Syst.*, 2018, **12**, 1289–1300.
9 O. J. Wahab, M. Kang, E. Daviddi, M. Walker and P. R. Unwin, *ACS Catal.*, 2022, **12**, 6578–6588.
10 N. Markovic and P. N. Ross, *Langmuir*, 1993, **9**, 580–590.
11 Y. Garsany, O. A. Baturina, K. E. Swider-Lyons and S. S. Kocha, *Anal. Chem.*, 2010, **82**, 6321–6328.
12 W. S. R. Teixeira, M. K. L. Silva, D. Grasseschi, C. A. Senna, A. Guimarães de Oliveira, J. Gruber, I. Cesarino and M. Oliveira Salles, *J. Electrochem. Soc.*, 2022, **169**, 047526.
13 D. Ortiz-Aguayo, K. De Wael and M. del Valle, *J. Electrochem. Soc.*, 2021, **169**, 115770.
14 S. Acharya, D. Das, T. N. Chatterjee, S. Mukherjee, R. Banerjee Roy, B. Tudu and R. Bandyopadhyay, *IEEE Sens. J.*, 2021, **21**, 20589–20595.
15 J. M. Díaz-Cruz, R. Tauler, B. S. Grabarić and M. E. E. Casassas, *J. Electroanal. Chem.*, 1995, **393**, 7–16.
16 G. Hinton and S. Roweis, *Advances in Neural Information Processing Systems*, 2002, vol. 15.
17 L. Van Der Maaten and G. Hinton, *J. Mach. Learn. Res.*, 2008, **9**, 2579–2605.
18 L. McInnes, J. Healy and J. Melville, arXiv, 2020, preprint, arXiv:1802.03426v3 [stat.ML], DOI: 10.48550/arXiv:1802.03426v3.
19 T. Sainburg, L. McInnes and T. Q. Gentner, *Neural Comput.*, 2020, **33**, 2881–2907.
20 T. Albrecht, G. Slabaugh, E. Alonso and S. M. R. Al-Arif, *Nanotechnology*, 2017, **28**, 423001.
21 G. F. Kennedy, J. Zhang and A. M. Bond, *Anal. Chem.*, 2019, **91**, 12220–12227.
22 L. Gundry, G. Kennedy, A. M. Bond and J. Zhang, *Faraday Discuss.*, 2022, **233**, 44–57.
23 J. X. Zhang, B. Yordanov, A. Gaunt, M. X. Wang, P. Dai, Y.-J. Chen, K. Zhang, J. Z. Fang, N. Dalchau, J. Li, A. Phillips and D. Y. Zhang, *Nat. Commun.*, 2021, **12**, 4387.
24 K. Simonyan and A. Zisserman, 3rd Int. Conf. Learn. Represent. ICLR 2015-Conf. Track Proc., DOI: 10.48550/arxiv.1409.1556.
25 A. Vladyka and T. Albrecht, *Mach. Learn. Sci. Technol.*, 2020, **1**, 035013.
26 A. Krizhevsky, I. Sutskever and G. E. Hinton, NIPS'12: Proc. 25th Int. Conf. on Neural Information Processing Systems 2012, 1, 1097–105.
27 Y. Wei, Y. Sun and X. Tang, *J. Phys. Chem.*, 1989, **93**, 4878–4881.
28 D. M. Mohilner, R. N. Adams and W. J. Argersinger, *J. Am. Chem. Soc.*, 1962, **84**, 3618–3622.

# Unsupervised Classification of Voltammetric Data beyond Principal Component Analysis

Christopher Weaver,[a] Adrian Fortuin,[a,b] Anton Vladyka[a] and Tim Albrecht*[a]

[a] School of Chemistry, University of Birmingham, Edgbaston Campus, Birmingham B15 2TT, United Kingdom

[b] Faculty of Mechanical Engineering, Helmut Schmidt University, 22043 Hamburg, Germany

t.albrecht@bham.ac.uk

## Supporting Information

## 1) Simulation of electrochemical data

All input cyclic voltammograms were simulated in DigiSim® v3.0. Temperature, scan rate, uncompensated resistance and double layer capacitance were kept constant at 298.15 K, 50 mV $s^{-1}$, 10 Ω, and 20 µF $cm^{-2}$ respectively, for all reactions. Electrode radii were varied from $10^{-6}$ cm to $10^{-1}$ cm in increments of one order of magnitude. Kinetic data was based on the electrochemical oxidation of aniline to polyaniline and is summarized in Table 1.[1,2]

*Table 1: Thermodynamic and kinetic parameters for reaction modelling*

| Variable / Unit | Reaction 1 | Reaction 2 | Reaction 3 |
|---|---|---|---|
| Diffusivity / $cm^2$ $s^{-1}$ | $9.2 \times 10^{-6}$ | $9.2 \times 10^{-6}$ | $4.6 \times 10^{-6}$ |
| Redox Potential / V | 1.294 | | 0.654 |
| Electron transfer rate / cm $s^{-1}$ | $3 \times 10^{-2}$ | - | 10 |
| Transfer coefficient | 0.65 | - | 0.50 |
| Initial concentration / mol $L^{-1}$ | 0.05 | 0.00 | 0.00 |
| Reaction rate constant / $s^{-1}$ | - | $1 \times 10^{4}$ | - |
| Reaction equilibrium constant / - | - | 250 | - |
| Mechanism | A + e = B | B + A = C | C + 2e = P |

## 2) Data image processing

CVs were first plotted on current vs potential axes. The current values for each plot were normalised such that values were scaled between -1and 1. This made up a 600-dimensional dataset of data values. Each plot was then converted to an RGB image of size 1080 x 1080 x 3 which were normalised by dividing by the maximum pixel value, 255. These images were then both flattened into a raw 3499200-dimensional dataset, and put through a feature extractor followed by flattening into a features 557568-dimensional dataset. The feature extraction process utilised a topless VGG-16 CNN to compress the raw traces into feature traces.

Once the datasets were constructed they were passed through the three dimensionality reduction algorithms: PCA, t-SNE, and UMAP. The hyperparameters chosen for each technique are summarised in Table 2 and Table 3. All parameters not specified were left as their default values.

This analysis process was implemented in Python v3.8.11. The VGG-16 network, architecture and weights, were provided by Tensorflow v2.3.0. PCA and t-SNE algorithms were provided by scikit-learn v0.24.2. Lastly, UMAP was provided by umap-learn v0.5.1.

# 3) Hyperparameter Optimisation

## 3.1) t-SNE Parameters

To find the optimum parameters for both silhouette score and perimeter score for each of the three datasets, a 2D grid search was performed on the two main t-SNE hyperparameters. These are the perplexity, and learning rate parameters. Once completed, the search produced a 2D matrix for both silhouette and perimeter scores on raw data, images data, and features data. The elements of each matrix were removed if the corresponding KL-divergence was greater than 0.6 then the argmax of each matrix was calculated to find the best parameters.

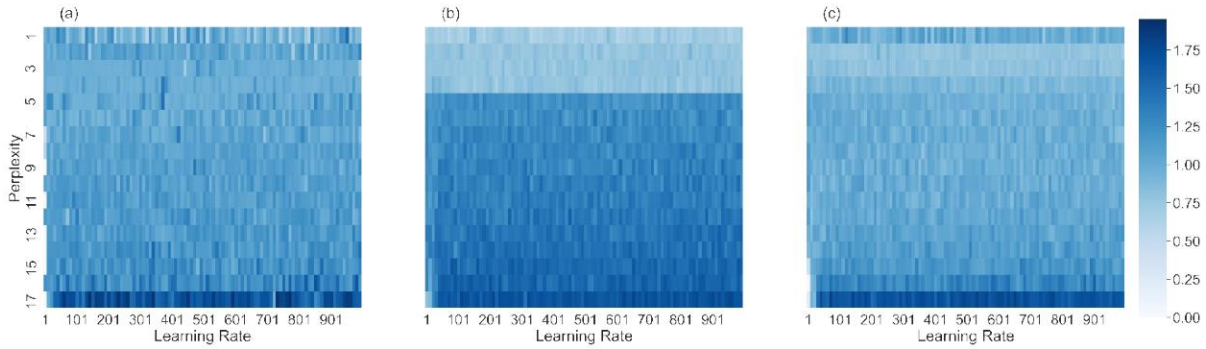



*FIG S1 Heatmaps showing the average perimeter score for each combination of learning rate and perplexity. Results are shown for the raw data dataset (a), images dataset (b), and the features dataset(c).*

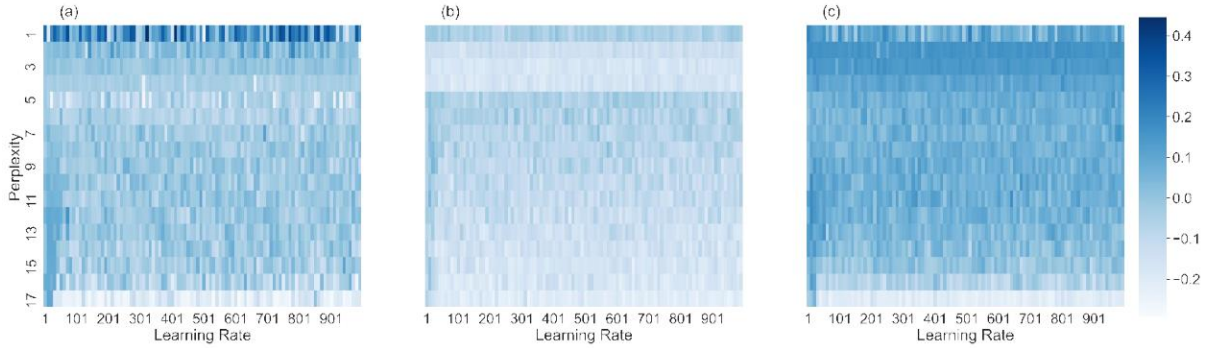



*FIG S2 Heatmaps showing the average silhouette score for each combination of learning rate and perplexity. Results are shown for the raw data dataset (a), images dataset (b), and the features dataset(c).*

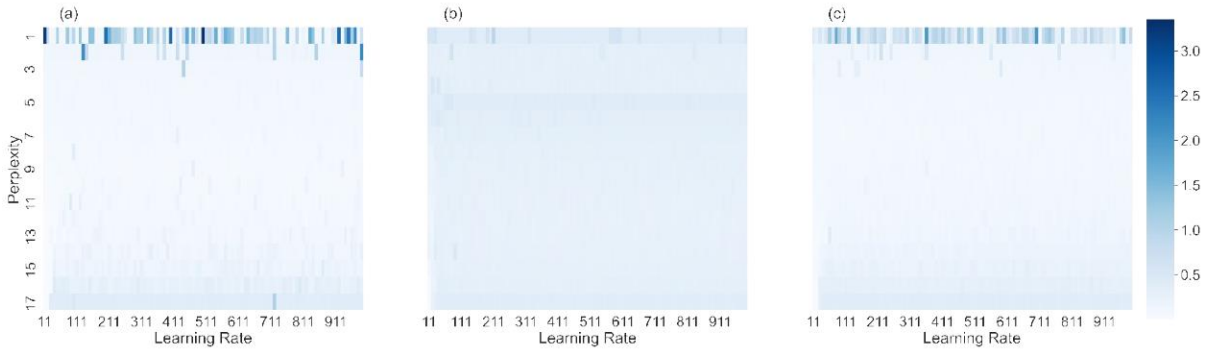



*FIG S3 Heatmaps showing the average KL-Divergence loss after training for each combination of learning rate and perplexity. Results are shown for the raw data dataset (a), images dataset (b), and the features dataset(c).*

*Table 2 Hyperparameters chosen for t-SNE for maximising both perimeter and silhouette scores alongside their KL-Divergence losses after training.*

| Data | **Perplexity** | **Learning Rate** | **KL-Divergence** |
|---|---|---|---|
| **Perimeter Score** | 17 | 771 | 0.38 |
| **Silhouette Score** | 1 | 411 | 0.38 |

| Images | **Perplexity** | **Learning Rate** | **KL-Divergence** |
|---|---|---|---|
| **Perimeter Score** | 17 | 211 | 0.38 |
| **Silhouette Score** | 9 | 21 | 0.38 |

| Features | **Perplexity** | **Learning Rate** | **KL-Divergence** |
|---|---|---|---|
| **Perimeter Score** | 17 | 71 | 0.38 |
| **Silhouette Score** | 1 | 511 | 0.38 |

(NB: Each t-SNE run utilised PCA initialisation provided by scikit-learn and was limited to 5000 iterations).

## 3.2) UMAP Parameters

To find the optimum parameters for both silhouette score and perimeter score, a 3D grid search was performed on the three main UMAP hyperparameters. These are the neighbourhood size (n_neighbors), minimum distance (min_dist), and learning rate parameters. Once completed, the search produced a 3D matrix for both silhouette and perimeter scores on raw data, images data, and features data. During the search, some of the runs produced a warning that the graph produced by spectral embedding was disconnected. These corresponding runs were removed. Then the argmax of each matrix was calculated to find the best parameters.

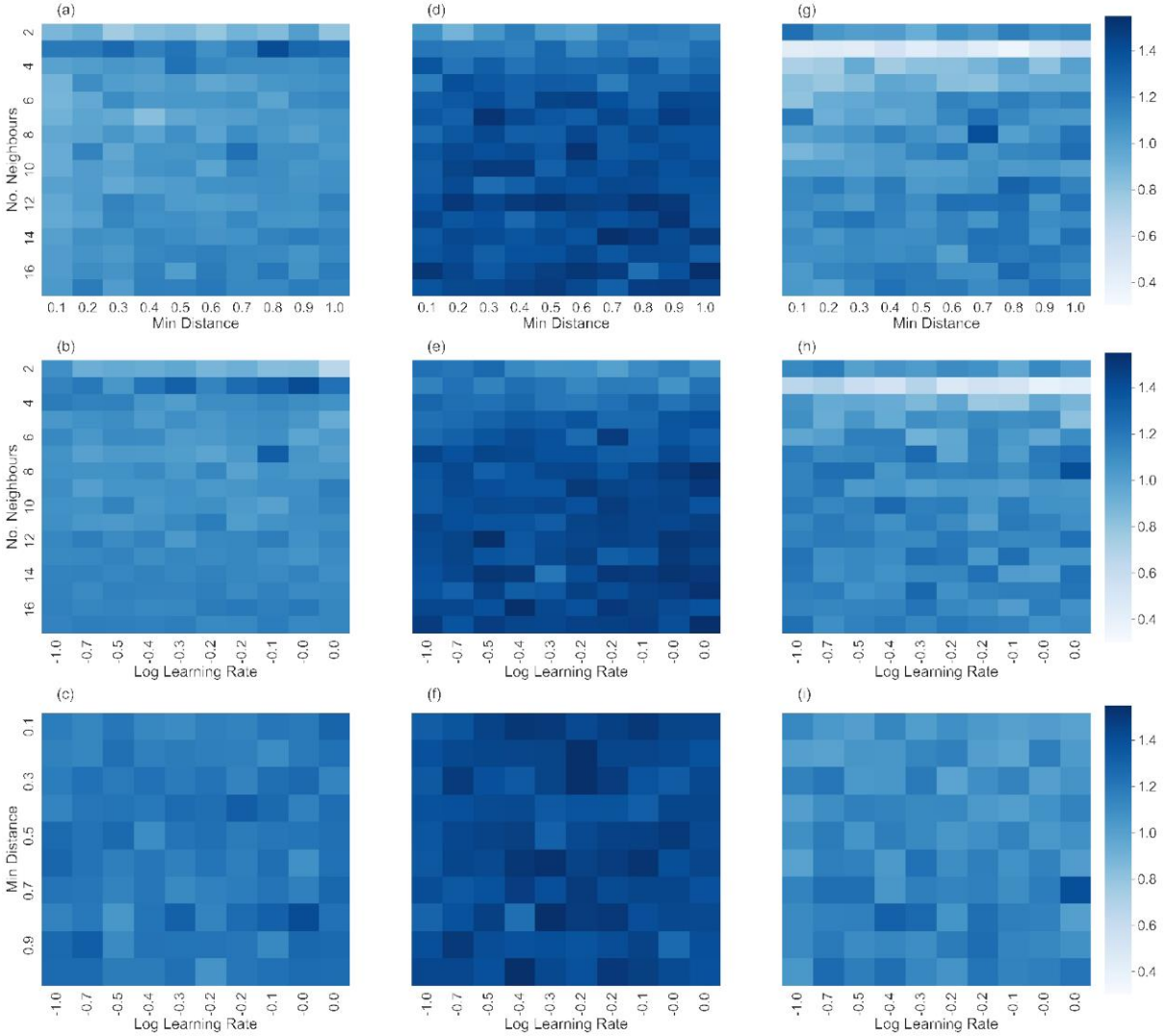



*FIG S4 Three slices of the 3D grid search matrix for perimeter score centred around the maximum score. Slices are shown for both the raw data dataset (a-c), the images dataset (d-f) and the features dataset(g-i).*

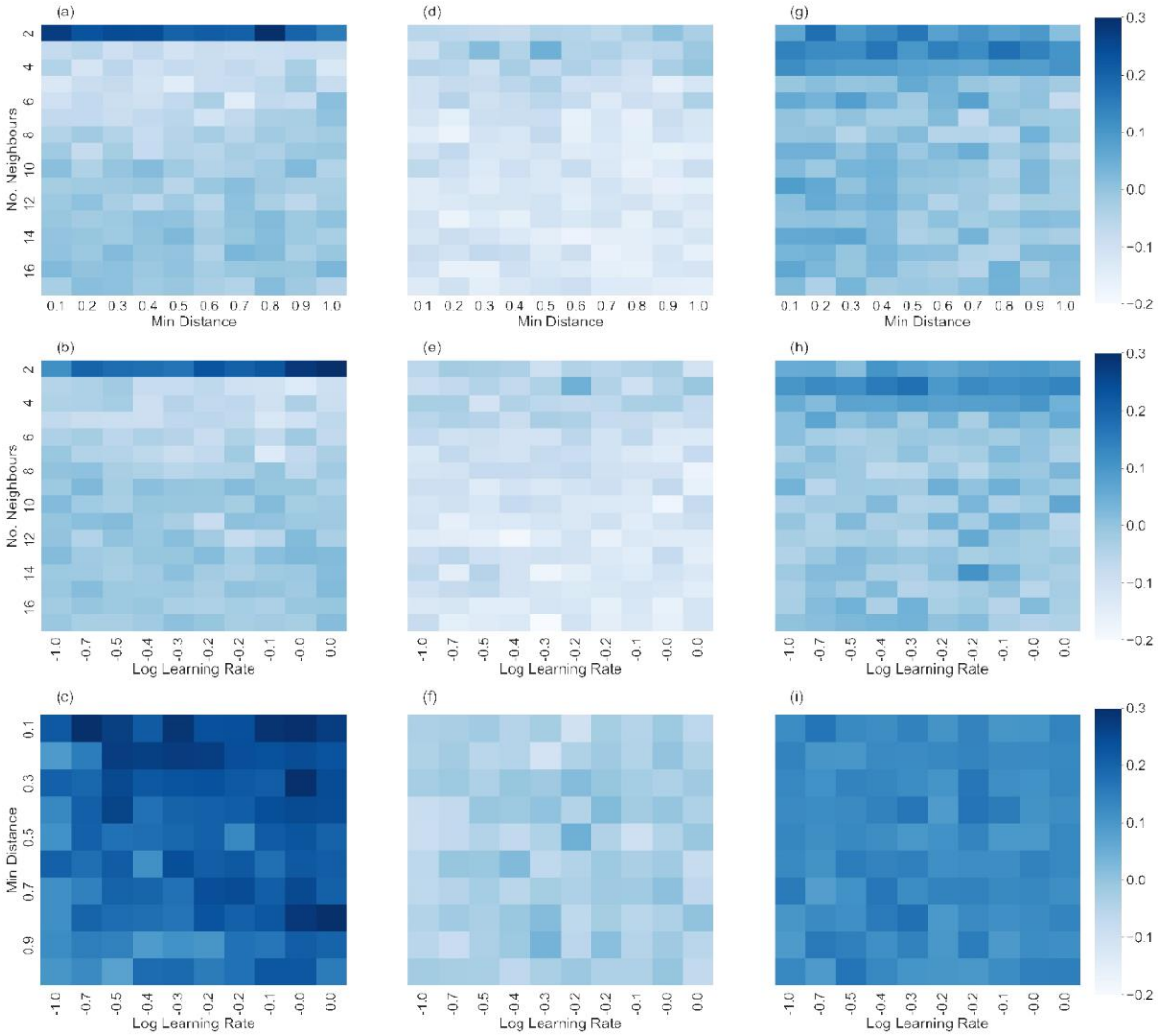



*FIG S5 Three slices of the 3D grid search matrix for silhouette score centred around the maximum score. Slices are shown for both the raw data dataset (a-c), the images dataset (d-f), and the features dataset(g-i).*

*Table 3 Hyperparameters chosen for UMAP for maximising both perimeter and silhouette scores.*

| Data | **No. Neighbours** | **Min Distance** | **Learning Rate** |
|---|---|---|---|
| **Perimeter Score** | 3 | 0.8 | 0.9 |
| **Silhouette Score** | 2 | 0.8 | 1.0 |

| Images | **No. Neighbours** | **Min Distance** | **Learning Rate** |
|---|---|---|---|
| **Perimeter Score** | 16 | 1.0 | 0.4 |
| **Silhouette Score** | 3 | 0.5 | 0.6 |

| Features | **No. Neighbours** | **Min Distance** | **Learning Rate** |
|---|---|---|---|
| **Perimeter Score** | 8 | 0.7 | 1.0 |
| **Silhouette Score** | 3 | 0.8 | 0.5 |

## 4) CVs and filter outputs

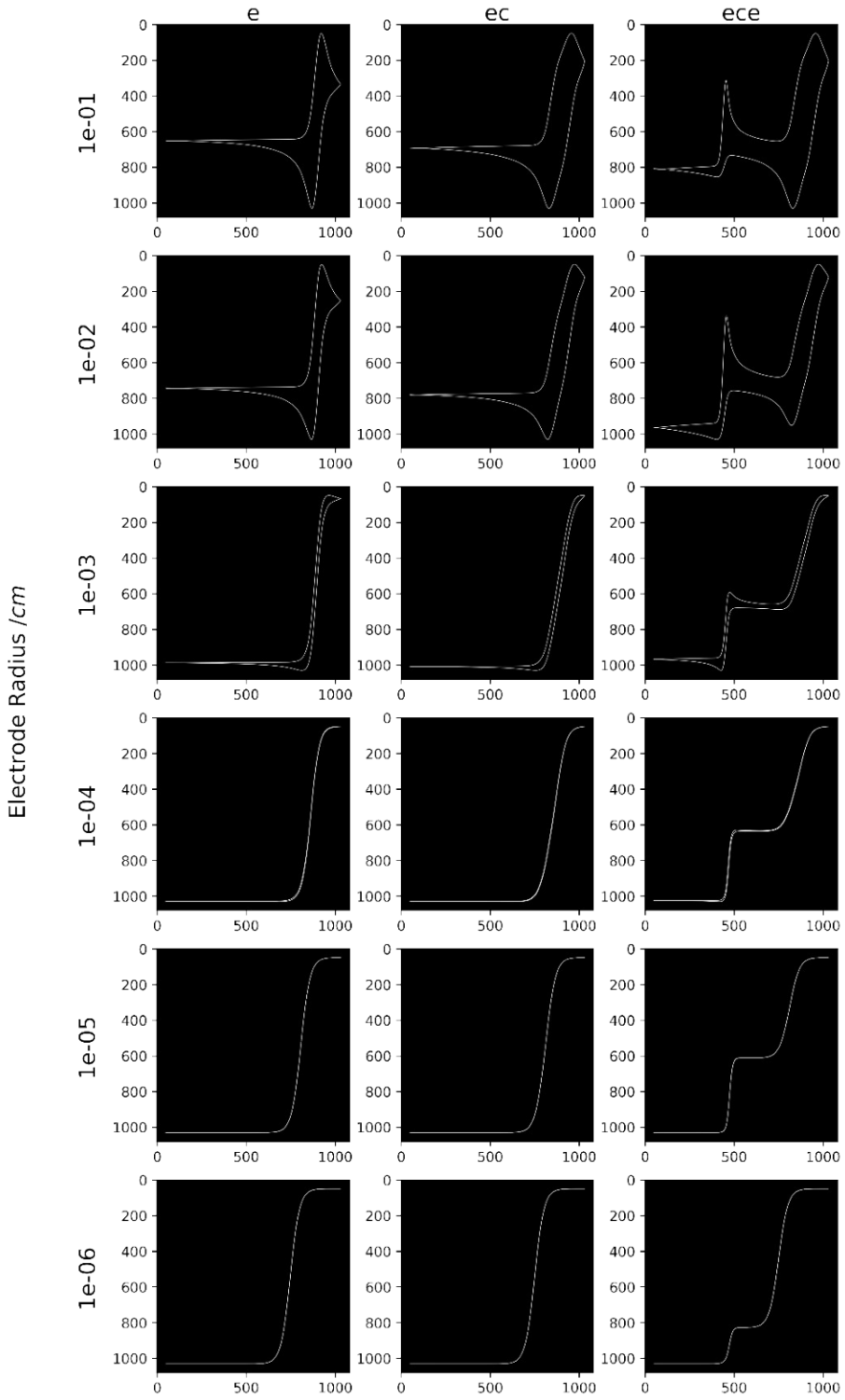



*FIG S6 Raw images of all CVs generated for the 3 mechanisms at 6 different electrode radii. Each image corresponds to a 1080 x 1080 x 3 RGB matrix.*

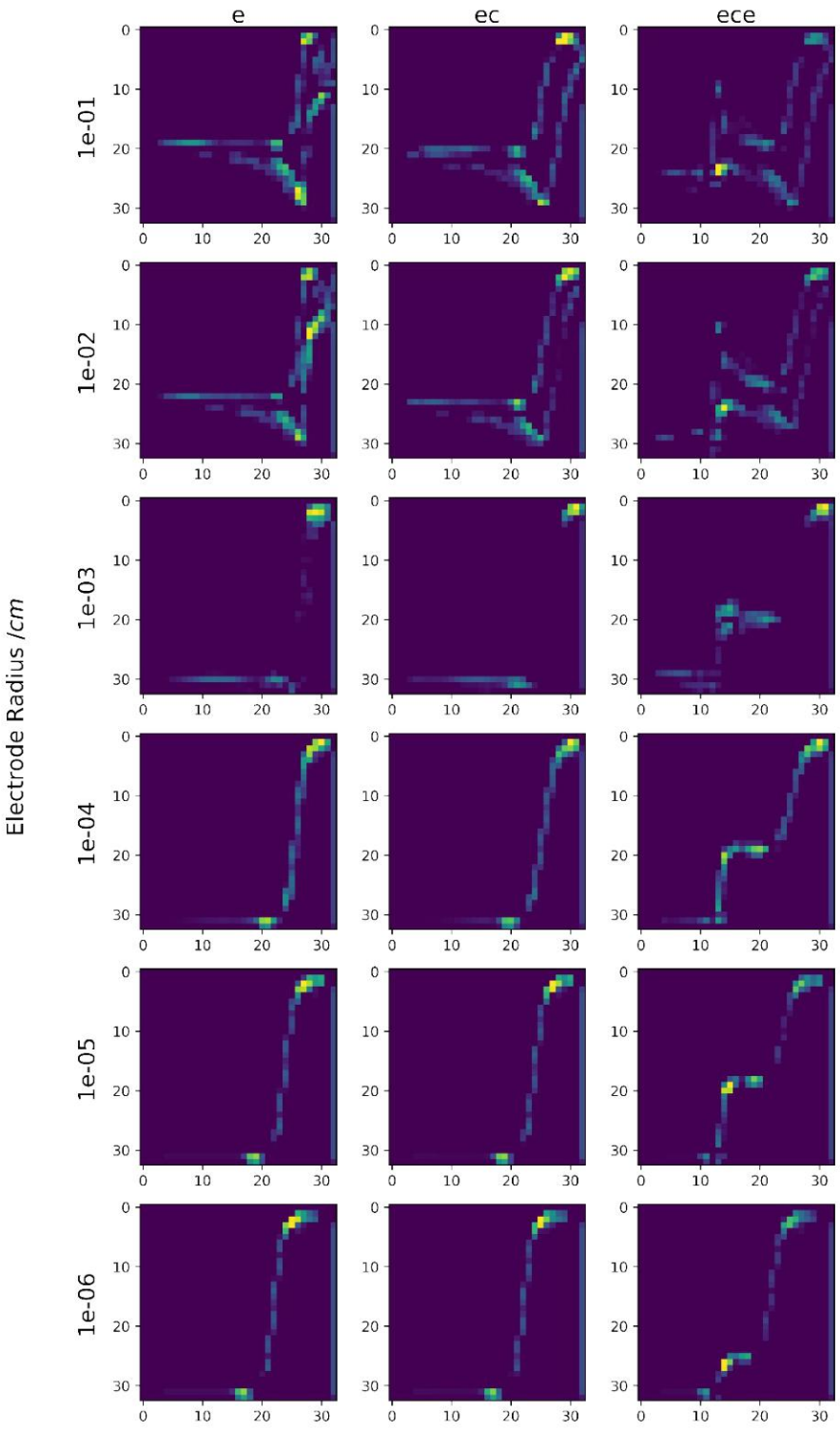
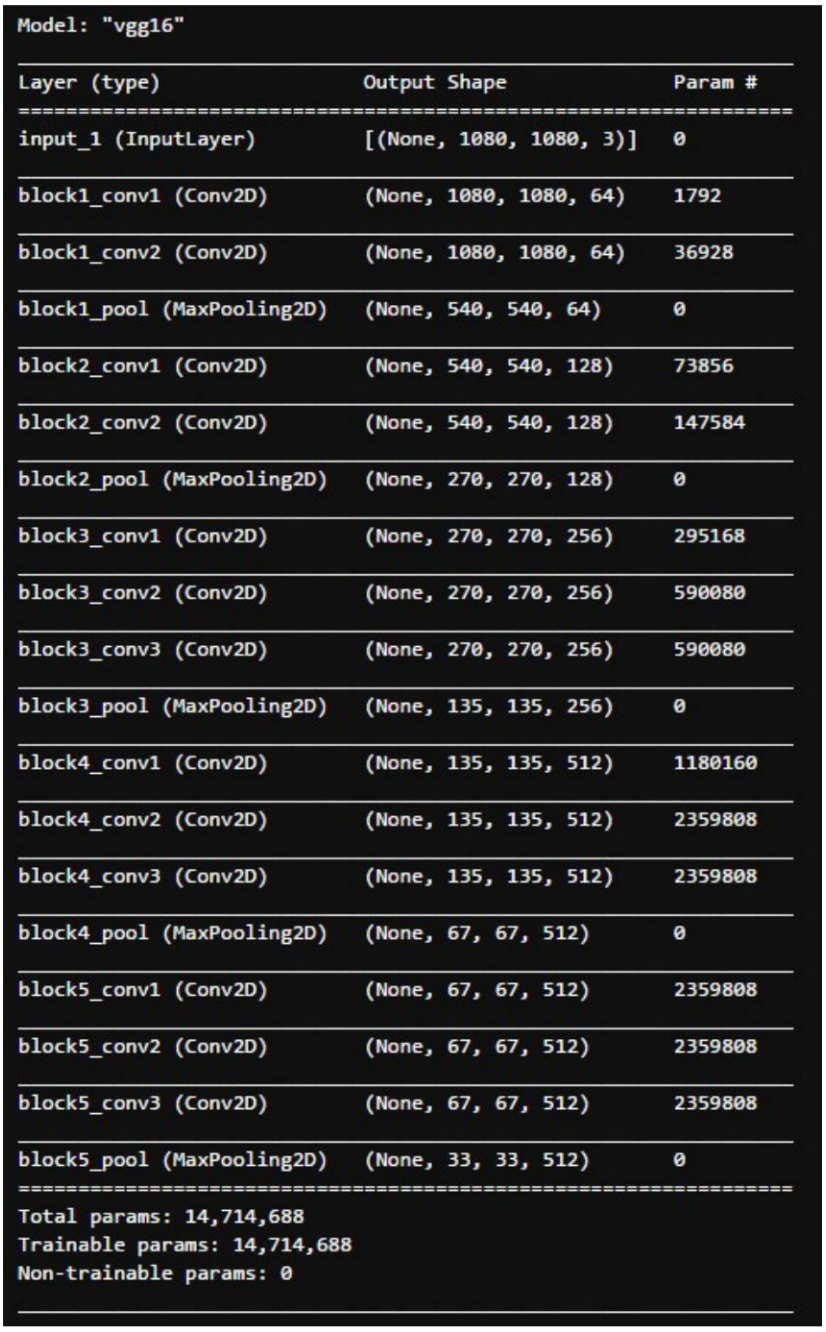



*FIG S7 Example slice of the 33 x 33 x 512 matrices outputted by the convolutional head of the VGG-16 neural network for each CV.*



```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1080, 1080, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 1080, 1080, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 1080, 1080, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 540, 540, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 540, 540, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 540, 540, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 270, 270, 128)     0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 270, 270, 256)     295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 270, 270, 256)     590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 270, 270, 256)     590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 135, 135, 256)     0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 135, 135, 512)     1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 135, 135, 512)     2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 135, 135, 512)     2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 67, 67, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 67, 67, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 67, 67, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 67, 67, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 33, 33, 512)       0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

*FIG S8 Architecture of the VGG-16 convolutional head used for generating the features dataset.*

5) Additional Results

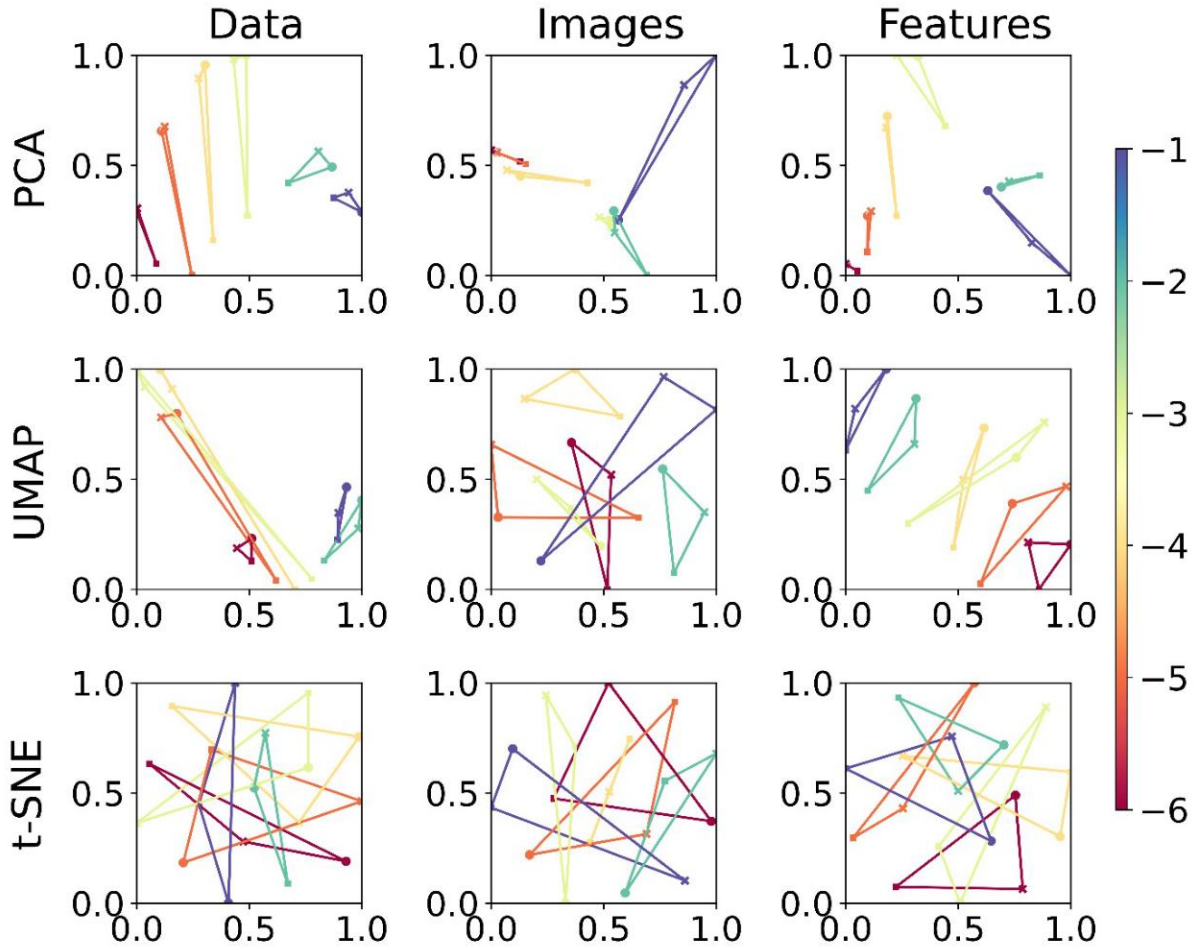



*FIG S9* *Scatter plots showing one example repeat over all electrode radii for each combination of dataset and dimensionality reducer when parameters optimised of maximum perimeter score were utilised.*

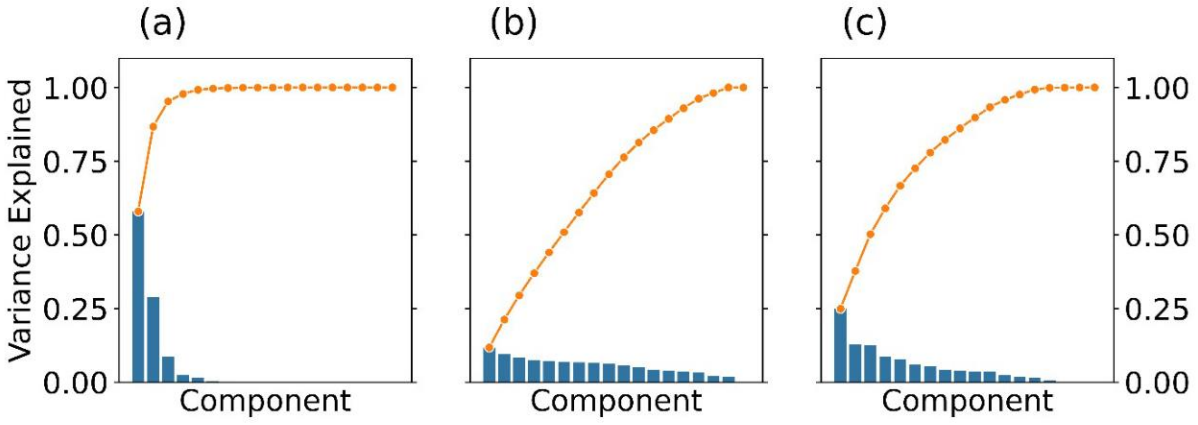



*FIG S10* *Scree plots showing the explained variance for each principal component (blue) and their cumulative quantities (orange) when applied to the raw data (a), images (b), and features (c) datasets.*

## 6) The effect of noise on the classification performance

In order to investigate the robustness of the classification in the presence of electric noise that would typically be observed in a CV experiment, a noise component of constant magnitude was added to simulated data (independent of potential, normally distributed). Its standard deviation $\sigma$ was taken relative to the scaled maximum current range from -1 to +1, as described in the main text. The figure below shows the corresponding image representation (1000 by 1000 pixels) for the three mechanisms and six electrode radii, for s = 2.5%.

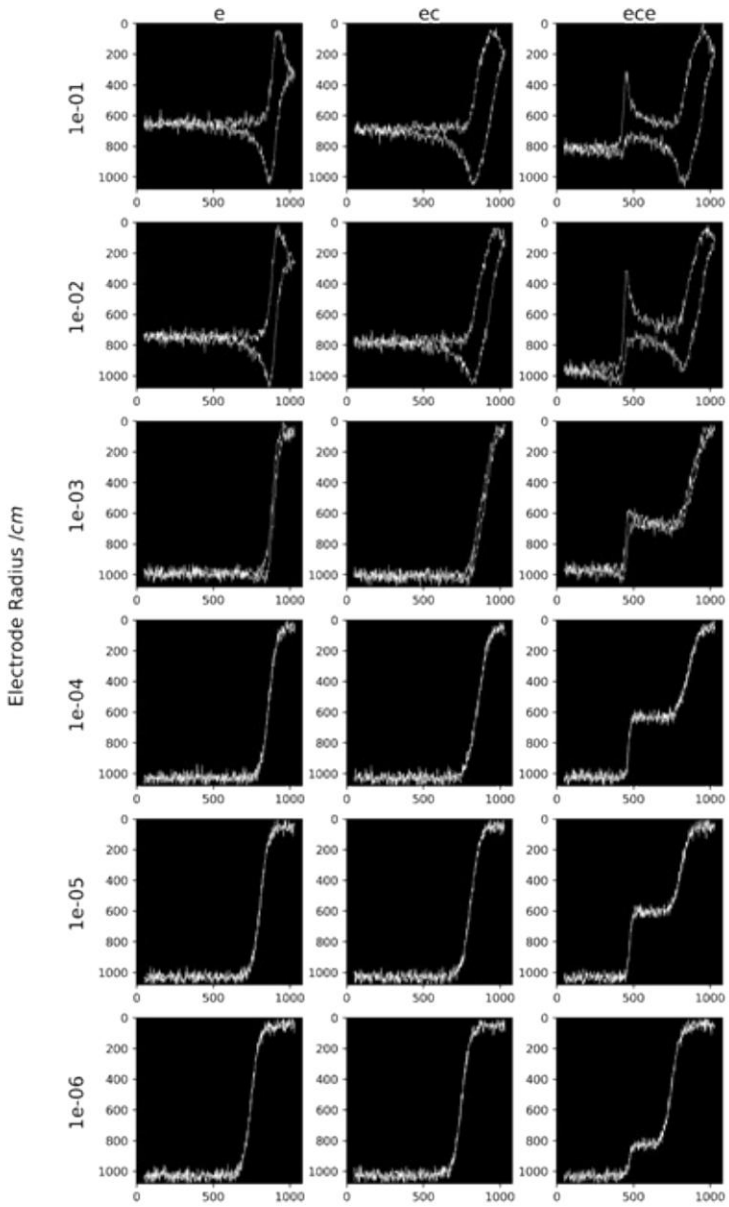



*FIG S11 Simulated CV data in the presence of electric noise of constant magnitude, $\sigma$ = 2.5%. Another set was generated for $\sigma$ = 1.25% (not shown).*

Generally, the presence of noise would be expected to increase overlap between similar CVs and hence render the differentiation more difficult (either between different mechanisms or electrode

radii). Indeed, for the perimeter score *P* (using accordingly optimised hyperparameters), this simple expectation appears to hold true, left column in fig. S12 below: the sequence of overall performance remains the same (t-SNE > UMAP > PCA), even though the numerical values seem to decrease with an increasing amount of noise present.

As for the noise-free data, the picture is however more complex for the *S* score, i.e. when the choice of hyperparameters is adapted. For PCA applied to the raw numerical data, the performance remains rather constant, application to images leads to the poorest performance of the three, while the performance when applied to features decreases somewhat, albeit not dramatically.

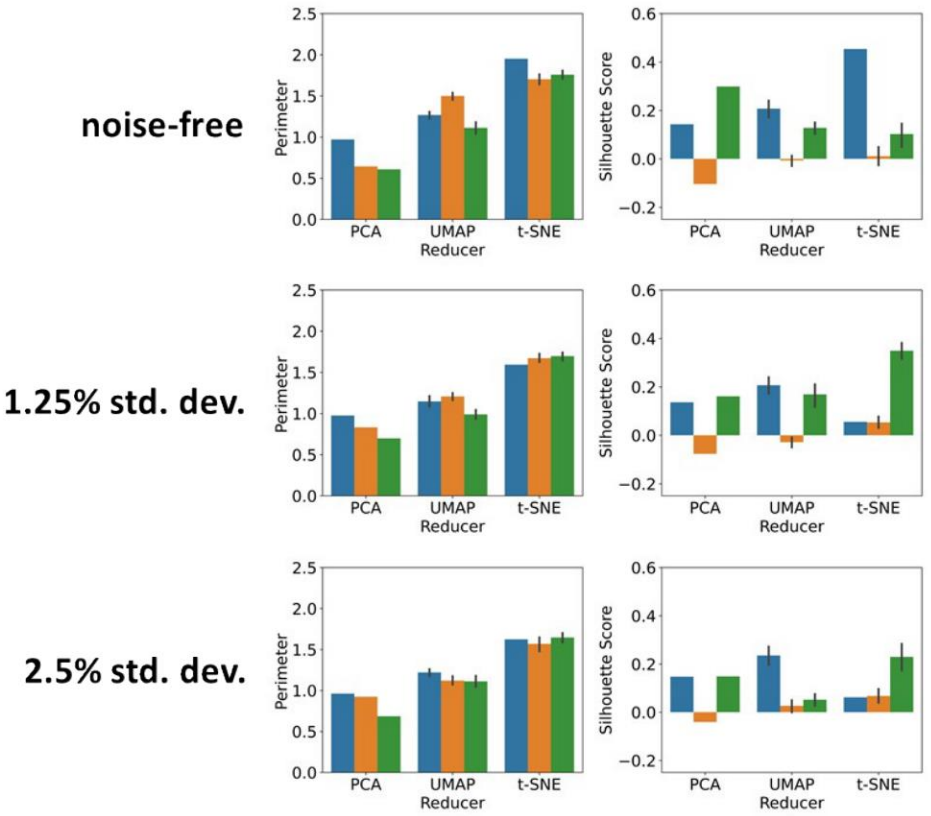



*FIG S12 Comparison of the overall classification results for PCA, UMAP and t-SNE in the presence of small to moderate amounts of electric noise, based on average P- and S scores, applied to: numerical data (blue), image data (orange); and feature extractor output (green).*

For UMAP, we observe a similar picture, even at the highest noise level, its performance when applied to features drops significantly. However, the most drastic effect is observed for t-SNE. Initially performing strongest when applied to numerical data, the *S*-score drops dramatically even for a relatively small amount of noise. Identifying the reasons for this effect requires further study. On the other hand, its application to features leads to a rather robust performance and for the highest noise level, t-SNE on features is best performing, jointly with UMAP on data.

## References

[1] Wei, Y., Sun, Y. & Tang, X. Autoacceleration and kinetics of electrochemical polymerization of aniline. J. Phys. Chem. 1989, 93, 4878–4881.
[2] Mohilner, D. M., Adams, R. N. & Argersinger, W. J. Investigation of the kinetics and mechanism of the anodic oxidation of aniline in aqueous sulfuric acid solution at a platinum electrode. J. Am. Chem. Soc. 1962, 84, 3618–3622.

## Appendix 4 Script Code

All data analysis, as mentioned in the main body of this thesis, was developed using the Python programming language. A repository of the analysis scripts written as part of this work is located at https://github.com/Chrisdw889/Thesis-Code.