Ion Channel Analysis with Deep Learning

'Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Samuel Ball

November 2023

Abstract

Ion channels govern many physiological processes in the body; and dysfunction results in several life changing conditions such as cystic fibrosis. It is thought that around 18% of drugs in the pharmaceutical market currently target ion channels, with many more substances such as sucrose and pesticides also affecting ion channel function. The process of recording and analysing ion channel data to is notoriously difficult and time consuming. Although some automatic analysis methods exist, they typically require significant supervision or parameter initialisation to run. This thesis aims to develop and test better models for such analysis by leveraging deep neural networks to both generate synthetic data and analyse patch-clamp signals.

In terms of data synthesis, the issue for deep learning is they are typically dependent on large volumes of data that include fully labelled "ground truth". We therefore used two methods; (i) Markovian simulation, (ii) a novel deep generative adversarial network (GAN) method. The GAN were trained on previously recorded data taken from a number of different sources including ENaC, TRPV4 and calcium activated potassium channels (BK) and successfully generated fully labelled data. T-SNE and UMAP analysis showed that this data is characteristically similar to the relevant source channel and also different to other channels generated; therefore, showing that the model was encapsulating the underlying biological mechanisms to some extent.

For data analysis we aimed to both improve on existing single channel analysis tool accuracy and, for the first time, use deep learning methods to recover continuous kinetic state. We developed a number of models with different architectures and found those based on convolutional neural networks (CNN) performed better than recurrent neural networks and considerably better than the original and published DeepChannel model. We then compared several different CNN based models including those adapted from visual science such as ResNet and U-Net, and found that in many cases ground truth Markovian state could be recovered with F1 accuracy of over 0.9. There was no one single "best" model since we found different models would often perform better than others in different analysis scenarios.

For model development and evaluation, "Markovian synthetic" data was used since it included a definitive ground truth. However, in my final results chapter I tested models on BK ion channels recorded from stably expressing HEK293 cells using the patch-clamp technique. The new models' performance at detection of open/close state were compared against expert human labelling and show that they could successfully detect the effects of changes of concentration in Ca2+ ions, Penitrem A and a drug previously untested on BK channels; Vernakalant.

In conclusion, this thesis explores the application of deep learning methods to the field of patch clamp ion channel electrophysiology. A new method for synthesis of ion channel data is developed, as well as two families of new models for ion channel idealisation; one more theoretical, outputting the Markovian configuration of the protein at any time, and one more practical, detecting common physiological markers when the channel experiences a change in conditions.

Acknowledgments

Firstly, I would like to thank Prof. Richard Barrett-Jolley, who conceived and supervised this project; without who I would undoubtably have been lost in the world of electrophysiology, and who guided me through many months remotely in the lab through the COVID19 pandemic. His wisdom and advice were invaluable to this project, and I have been proud to have been part of the "nerdy maths club" he founded.

I'd also like to thank the Biotechnology and Biological Sciences Research Council for funding this project; without whom none of this work would be possible.

Thank you to all the "girls in the lab": Dr Caroline Staunton, Dr Rhiannon Morgan, Dr Fiona O'Brian, Jekaterina Kumiscia and Dr Samantha Jones, who were an endless source of wisdom for my biological queries, and will never let me forget the correct pronunciation of "autoclave", or my cell culture endurance. Also thanks to Dr Robert Heaton and the work running group for the weekly parkruns.

Thanks to Dr Sean Brennan, Dr Numan Celik, Dr Lina Kadir and Dr Nathanael O'Neill who provided supplementary data and work to this work; as well as assistance in data recording and analysis.

I would like to thank my mother, Helen, who has supported me throughout my entire time at university and who without which, I would not have lasted four days, let alone four years on this work.

Lastly, I would like to dedicate this work to my late grandma, Jean "Nasa" Unwin who sadly passed away during this project; her determination and warmth have always reminded me to keep going with a smile on my face.

Contents

List of Figures

List of Tables

Abbreviations

AI; -; Artificial Intelligence

ANN - Artificial Neural Networks

ANOVA - Analysis of Variance

ATP - Adenosine Triphosphate

nAChR - Nicotinic Acetylcholine Receptor BK - "Big" Potassium Channel

BKU - Bauer-Kienker Uncoupled form

CDF - Cumulative Distribution Function

CNN - Convolutional Neural Network

CSV - Comma Separated Value

CT - computerised tomography

DMSO - Dimethylsulfoxide

DTW - Dynamic Time Warping

EC50 - Half maximal effective concentration

ECG - Electrocardiogram

EDA - Electrodermal Activity

EEG - Electroencephalogram

EGTA - ethylene glycol-bis(ß-aminoethyl ether)-N,N,N′,N′-tetraacetic acid

EMG - Electromyography

EOG - Electrooculography

FBS - Fetal Bovine Serum

FRET - Förster Resonance Energy Transfer

GAN - Generative Adversarial Network

GPU - Graphics Processing Unit

GRU - Gated Recurrent Unit

GSR - Galvanic Skin Response

HEK - Human Embryonic Kidney

HEPES - 4-(2-hydroxyethyl)-1-piperazineethanesulfonic acid

HMM - Hidden Markov Model

IC50 - Half maximal inhibitory concentration

IEEE - Institute of Electrical and Electronics Engineers

IOP - Inside out patch

J-SMURF - Jump Segmentation Multiresolution Filter

LQT - Long QT

LSTM - Long Short-Term Memory

MDL;-; Minimum Description Length

MIR - Manifest Interconductance Rank form

MLP - Multi Layer Perceptron

MMD - Maximum Mean Discrepancy

SKM - Segmental K-Means

MNIST - Modified National Institute of Standards and Technology

GPU; GraphicsGraphicsnAChR - Nicotinic Acetylcholine Receptor

NLP - Natural Language Processing

OOP - Outside out patch

PCA - Principle Component Analysis

PPG - Photoplethysmogram

ReLU - Rectified Linear Unit

RNN - Recurrent Neural Networks

SKM - Segmental K-Means

SVM - Support Vector Machine

TANE - Timeseries Amplitude Normalisation via Estimation

TRP - Transient Receptor Potential channels

t-SNE - t-distributed Stochastic Neighbour Embedding

UMAP - Uniform Manifold Approximation Projection

1 General Introduction

Ion channels are transmembrane proteins that are fundamentally important for many physiological processes in the body such as action potentials; controlling these ion channels is a common target for pharmacologists, as dysfunction of these channels is responsible for a number of health conditions. Measuring the effects of pharmacological agents on ion channels can be done via patch clamp electrophysiology, a process where the current across a membrane is measured in response to different electrical potentials or concentration of substances such as a drug or ions. One major obstacle for ion channel electrophysiologists is analysis of the output of this process; ideally the data should resemble a piecewise constant function (or "square" wave), with edges denoting openings and closings of a single channel; however the sensitivity of the experiment leads to the signal being notoriously noisy and difficult to analyse automatically. Whilst significant efforts have been made to analyse ion channel data algorithmically, many of these methods have constraints on the types of noise present (such as relying on a lack of long-term baseline drift), or a large number of parameters a researcher has to set before idealisation can occur. This results in ion channel analysis still being a significantly tedious task, with researchers often having to relabel files by hand to correct mistakes.

Deep learning is a type of artificial intelligence (AI) that has shown promise in other areas of data analysis. In image analysis, deep learning (specifically convolutional models) have shown to have state-of-the-art performance in detecting objects and segmenting them into relevant sections, similar to the process occurring in ion channel signal analysis. This study looks at the ways in which deep learning can be used in the field of ion channel physiology to approach data driven problems, and build models to better understand how these channels operate.

1.1 Ion Channels

1.1.1 Discovery

It is difficult to pinpoint exactly where the discovery of ion channels starts; Hodgkin and Huxley are commonly cited as the "grandfathers" of the ion channels due to their work on action potentials in squid axon cells (Hodgkin & Huxley, 1952); however significant work had been done previously studying the action potential of squid axons (Hodgkin & Huxley, 1939), which in turn built on previous work on the structure of nerve fibres in similar animals (Young, 1936). Hodgkin and Huxley continued making inroads into the analysis of ion channel function. Hodgkin and Huxley found that by changing the potential difference across a cell membrane, they could induce a current that decayed over time. Furthermore, by changing the temperature, the rate at which the conductance changes would differ as well. This work won the pair the Nobel prize in Physiology or Medicine in 1963.

Work continued on membrane conductance in this form for a decade, until another Nobel prize winner, Bernard Katz, questioned if these changes in current came from single "gates" opening and closing (Katz & Miledi, 1972). Due to limitations in recording techniques at the time, the current signal was too noisy to resolve individual channels.

In 1978, individual channels were eventually recorded using the revolutionary "patch clamp" technique (Sakmann & Neher, 1984). By applying slight suction to a pipette, a seal was established with the cell membrane that had a resistance across the membrane of 10-100 giga-ohms; this method for achieving this resistance was a ground-breaking discovery that allowed far higher resolution measurements with a far larger signal to noise ratio, as well as the isolation of single ion channel recordings. Sakmann and Neher won the Nobel prize for their work in 1991.

1.1.2 Ion Channel Physiology

1.1.2.1 Characteristics

Primarily, ion channels are responsible for selectively allowing ions to traverse the cell membrane. This separation of ions across the membrane creates an electrical potential across the cell (often called a membrane potential), which if occurring at a rapid rate can create bursts of changes of potential energy known as action potentials.

The opening and closing (or "gating") of these channels can be controlled with a number of different mechanisms, depending on the specific channel. Voltage gated channels gate in response to a change in potential difference across the membrane, with perhaps the most well-known being the voltage-gated sodium channels responsible for the action potential in the human body; in this case a "chain reaction" is created, where a voltage activated channel will open, causing a brief spike in its own potential, causing other nearby ion channels to behave in a similar manner.

Ligand-gated ion channels respond to specific molecules binding to the channel to control opening and closing. One example of a ligand-gated channel is the nicotinic acetylcholine receptor (nAChR) channel (Dani, 2015); which is typically activated by acetylcholine. When opened, the movement of ions creates a potential difference within the cell that can cause the activation of voltage-gated channels as above. As ligand-gated channels' behaviour is managed by external molecules, they are of particular interest for pharmacologists; for example nAChR channels also activate in the presence of nicotine.

Voltage and ligands are two mechanisms by which ion channels gate, and are generally researched the most due to their proximity to electrophysiology and pharmacology respectively; however that is not to say that they are the exclusive means by which ion channels operate; for example light-gated channels in algae (Harz & Hegemann, 1991) are gated via light stimulus instead of voltage or ligands.

1.1.2.2 Physiological Role

Since the first work in ion channels was concerned with the nervous system in squid axons, work on ion channels has typically been focused on the action potential and associated physiological processes with it. Neher and Sakmann's work (Neher & Sakmann, 1976) on action potentials in excitable muscle fibres opened further research into ion channels' function affecting other physiological systems.

Both the nervous and muscular systems rely on the membrane potential being excited by some external stimulus, either to fire neurons or muscles to achieve their function. However, it should be noted that the resting membrane potential controlled by ion channel function is also responsible for several other physiological processes in the body; from circadian rhythms (Belle, Diekman, Forger, & Piggins, 2009; Noguchi, Wang, Pan, & Welsh, 2012), hearing (Ashmore, 2008), wound healing (Reid & Zhao, 2014) and pigmentation (Bellono, Kammel, Zimmerman, & Oancea, 2013).

In addition to resting membrane potential, ion channels are responsible for managing intracellular concentration of ions (Hammond, 2015), water transport (Martinez, 1987) and calcium signalling (Bootman et al., 2001)

1.1.3 Molecular Mechanism

Most of what we know about how ion channels work comes from studying their behaviour; however there have been significant efforts to the physical mechanisms behind ion channel function. Hille was first to theorise that in sodium channels, sodium ions freely move through the pore, with ions that are larger being blocked due to their size (Hille, 1971). It was not until MacKinnon used X-ray crystallography to analyse the structure of the potassium channel KcsA was this confirmed (Doyle et al., 1998), with more information about the structure of the proteins that make up the ion channel being revealed. Later, Cryo-EM methods (see Chapter 1.2.1.2) gave additional ways to study ion channels.

We now know that ion channels are made up of a number of subunit proteins typically surrounding an aqueous pore of the lipid bilayer. The number of subunits within an ion channel is what normally dictates ion channel families; for example the voltage gated potassium channel family are made up of 4 similar α subunits for primary function, with optional additional auxiliary β subunits dictating localisation (González et al., 2012).

Ions across the cell membrane will move along an electrochemical gradient; this is a balance of a concentration gradient and an electrostatic one; in a process similar to osmosis the concentration gradient will try and equalise the concentration of ions on each side of the cell membrane; however if a potential is applied across the gradient, ions will be drawn across the membrane to balance the charge; in this case the concentration of ions on either side of the cell membrane is not necessarily equal. This forms a new equilibrium between the concentration and electrostatic equilibria where the forces balancing the two equilibria are equal. The relationship between electrostatic equilibria and ionic concentration is central to ion channel function; and can be described using the Nernst equation (Veech, Kashiwaya, & King, 1995):

1.1.4 Ion Channel Dysfunction

As ion channels are responsible for many physiological functions in the human body, the study of their dysfunction is important to remedy many health conditions and study the effects of chemicals and toxins that may cause dysfunction when administered. Controlling ion channel function via pharmacological means can also be used to treat conditions not currently thought to be directly caused by ion channel dysfunction, as well as for recreational means in commonly found drugs.

Channelopathies are genetic conditions caused by mutations in genes that cause dysfunction of ion channels. Due to the fundamental nature of ion channels, channelopathies are a wide ranging family of conditions that effect multiple different systems in the body (Kim, 2014). A table of commonly known channelopathies along with the ion channel dysfunction responsible for them can be seen in Table 1.1. Note that this

is not a complete list of channelopathies, and some nuance exists within the conditions (for example epilepsy is known to present across a wide number of ion channels' dysfunction).

Table 1.1:Channelopathies along with their responsible channel.
Many health conditions are caused by dysfunction of ion channels; understanding how these ion channels work is a key step into understanding the conditions they cause.

Condition	**Responsible Ion Channel(s)**	**Source**
Bartter's Syndrome	Chloride Channels	(Rodríguez-Soriano, 1998; Simon et al., 1997)
Congential Hyperinsulinism	ATP Sensitive Potassium Channels	(Pinney et al., 2008)
Cystic Fibrosis	Chloride Channels	(Quinton, 1990; Welsh, 1990)
Episodic Ataxia	Voltage Gated Potassium Channels	(Browne et al., 1994)
Fibromyalgia	Sodium Channels	(Vargas-Alarcon et al., 2012)
Long-QT Syndrome	Voltage Gated Potassium Channels	(Ackerman, 1998; Roden, 2008)
Short-QT Syndrome	Voltage Gated Potassium Channels	(Brugada, Hong, Cordeiro, & Dumaine, 2005; Gaita et al., 2004)
Timothy Syndrome	Calcium Channels	(Barrett & Tsien, 2008)

Tinnitus	KCNQ Potassium Channels	(Henton & Tzounopoulos, 2021)
Seizure	Voltaged Gated Potassium Channels	(Armijo, Shushtarian, Valdizan, Cuadrado, & Adin, 2005; Buono et al., 2004)

Perhaps due to the relative focus on the action potential compared to other physiological mechanisms, there is a larger body of work studying cardiac conditions compared to other conditions in the table above (such as tinnitus). These channelopathies are typically long-term, life altering conditions; in the case of Long QT (Figure 1.1) syndrome symptoms such as fainting or seizures can present, with cardiac arrhythmias caused by dysfunction of the action potential in the ion channels potentially leading to death. While some channelopathies form long term permanent conditions, there are conditions where people can lead ordinary lives until certain stimulus is presented; people with malignant hyperthermia (a channelopathy caused by dysfunction in the RYR1 calcium channel) experience no symptoms until anaesthetics are applied (Schuster, Roewer,

Schneiderbanger, & Johannsen, 2014).

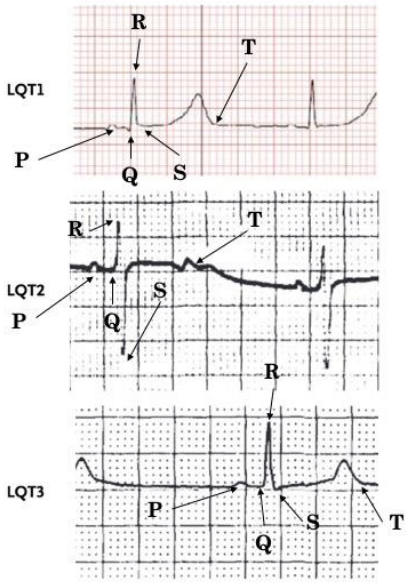


Figure 1.1: Three examples of Long QT syndrome. (Roden, 2008). *Ion channels form a critical part of healthy cardiac function and their dysfunction can lead to conditions such as Long QT syndrome. The distance between the "sharp" peak and "smooth" peak has been shown to be caused by ion channel function; if this distance is too long, a patient can suffer heart palpitations or seizures.*

Figure 1.2: The Buthus Tamulus scoripion and Iberiotoxin.
The Buthus Tamulus scorpion (A) uses Iberiotoxn (B) as a venom to inhibit calcium activated potassium channels to induce hypertension and tachycardia to its prey. Iberiotoxin and its synthetic analogues are commonly used in physiology to test for inhibition of ion channel activity.

1.1.4.1 Toxins and Pesticides

Ion channel dysfunction is not always unintentional; many toxins and pesticides function via causing ion channel dysfunction. A well-known example, Iberiotoxin, is a toxin present in *Buthus tamulus* scorpion (Figure 1.2) venom that inhibits calcium activated potassium channels (Galvez et al., 1990). By blocking these channels, cardiac activity is disrupted and typically victims suffer from symptoms such as hypertension and tachycardia, with potentially lethal consequences (Bawaskar & Bawaskar, 1992). It is thought that toxins targeting ion channels work in one of two methods; either by blocking the pore itself (Hille, 1975; Olivera, 1997) or by altering the gating mechanism (Cahalan, 1975; Koppenhöfer & Schmidt, 1968). A non-exhaustive table of toxins along with the channel they target can be found in Table 1.2, adapted from (Kalia et al., 2015).

Table 1.2: A number of toxins along with the ion channels they target, and how they bind to the ion channel to affect function.
Many toxins target ion channels to function as they play a key role in the body.

Toxin	**Target**	**Binding Site**	**Reference**
ω-conotoxin	Cav Channels	Pore	(Nielsen, Schroeder, & Lewis, 2000)
Iberiotoxin	BK Channels	Pore	(Candia, Garcia, & Latorre, 1992; Galvez et al., 1990)
Saxitoxin	Nav, Kv Channels	Pore	(Hille, 1975; Terlau et al., 1991)

Tetrodotoxin	Nav Channels	Pore	(Hille, 1975; Terlau et al., 1991)
Kurtoxin	Cav Channels	Voltage Mechanism	(Sidach & Mintz, 2002)

In fact, due to the biodiversity of naturally occurring toxins, they are a significant area of study for therapeutic drugs via creation of analogues due to the potency and stability of venoms (Lewis & Garcia, 2003). Perhaps the most compelling example is Ziconotide; a non-opioid pain relief drug developed in part in response to the ongoing opioid crisis. This drug is derived from the conotoxin venom of the marine mollusc: *Conus geographus*, (Figure 1.3) and blocks the voltage gated Cav2.2 channel, a known channel responsible for the transmission of pain (Miljanich, 2004; Safavi-Hemami, Brogan, & Olivera, 2019).

Ion channels are not only of particular interest in human patients; an increasingly pressing problem of food security leads investigation into the development of more effective pesticides with lower toxicity seen in humans. Neonicotinoids are a family of insecticides make up 25% of total insecticide sales globally (Ffrench-Constant, Williamson, Davies, & Bass, 2016), and work by targeting nAChR channels in a similar way as nicotine does. With supposedly low levels of toxicity in humans, these show promise as a way to achieve food security without passing toxic effects up the food chain; however there is significant controversy as to the effectors of neonicotinoids on worker bee

population leading to some restriction on their usage globally (Gill, Ramos-Rodriguez, & Raine, 2012).

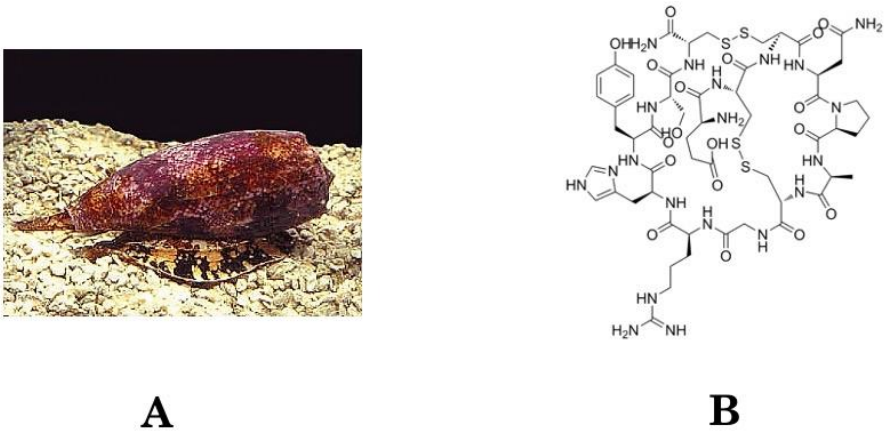


Figure 1.3: The Conus Geograhpus and Conotoxin.
The venom of the Conus geographus (A) conotoxin and it's molecular diagram (B), has been used to synthesise a non-opioid pain relief drug that blocks voltage gated calcium channels responsible for the transmission of pain in the nervous system.

1.1.4.2 Drug Targets

Due to the widespread function of ion channels, and the conditions caused by ion channel dysfunction, ion channels are a common target for drug development; with an estimated $12 billion industry built around drugs targeting ion channels (Cox, 2015).

Due to the history of ion channel research, particular interest is given to cardiac channels and drugs associated with treating associated conditions; the World Health Organisation's essential medicine list includes Amlodipine and Verapamil (both voltage gated calcium ion channel inhibitors); and Amiodarone (a voltage gated potassium ion channel blocker) as cost effective treatments for their relative conditions; this shows the relevance of ion channel targeting pharmaceuticals on the drug landscape as a whole.

Similar to how toxins function, drugs targeting ion channels typically work either by blocking the channel pore structurally, or effecting the gating process mechanistically. For example, the anesthetic Lidocaine functions by blocking the channel pore and flow of ions directly in NaV channels.

Pharmacological agents targeting ion channels are not just limited to the cardiac domain; Table 1.3 shows a selection of drugs as well as their targets, showing proposed treatments for a number of conditions. It is of particular note that these conditions are not limited to those caused by ion channel dysfunction; for example although there is only burgeoning evidence for the effect of ion channels in the role of multiple sclerosis; there is already progress in developing therapeutic treatments via ion channel targeting.

Table 1.3: Pharmaceutical drugs that target ion channels, the channel they target and their method of affecting the channel.
Many pharmaceutical agents target ion channels as a means to function; for example many anaesthetics target NaV channels responsible for pain transmission.

Drug (and use)	**Target**	**Binding Site**	**Reference**
Vernakalant Hydrochloride (Atrial Fibrillation)	Cardiac Kv Channels	Pore	(Burashnikov, Pourrier, Gibson, Lynch, & Antzelevitch, 2012; Fedida et al., 2005; Naccarelli et al., 2008; Seyler, Li, Schweizer, Katus, & Thomas, 2014)
Lidocaine (Anesthetic)	NaV Channels	Pore	(Bean, Cohen, & Tsien, 1983)
Zolpidem (Ambien) (Insomnia)	GABA channels	Positive Modulator	(Perrais & Ropert, 1999)

Nateglinide (Diabetes)	ATP dependent K channels	Negative Modulator	(McLeod, 2004)
Alprazolam (Xanax) (Anxiety)	GABA channels	Positive Modulator	(Fride, Skolnick, & Arora, 1990; Massah, Gharaghani, Lordejani, & Asakere, 2016)

Outside of prescribed drugs, recreational drugs such as nicotine, cannabis and ketamine all target ion channels (Dani, 2015; Watkins, 2019; Yamakura, Chavez-Noriega, & Harris, 2000). In addition, some food chemicals such as sucrose are known to affect ion channel function (Murakami & Kijima, 2000; White, 1995).

Ion channels affect our health in a wide variety of ways, from cardiovascular function to hearing; and can be affected through a number of different avenues (prescribed/recreational drugs, toxins and food chemicals). Since the impact of ion channels is so large, and our contact with substances that affect them is so common, study into how these substances effect the function of ion channels is important.

1.2 How are ion channels studied?

There are several ways ion channels are currently studied, typically with emphasis on characterising the structure or the function of particular ion channels. A brief summary of some of the most popular techniques is given in Table 1.4.

Table 1.4: Methods of studying ion channels.
There are numerous methods by which we can study ion channels by, each with their own goals, advantages and disadvantages. For example structural analysis via crystallography/microscopy takes a static image of the ion

channel, whereas patch clamp electrophysiology gives a functional overview of how the ion channel reacts over time.

Method	**Structural/Functional**	**Output**
X-Ray Crystallography	Structural	High resolution structural image of the channel at a single point in time.
Cryo-electro microscopy	Structural	Lower resolution structural image of the channel at a single point in time, but requires less pure of a sample to start
FRET	Functional	Real time openings and closings of an individual channel, however has a high sensitivity to background noise
Patch Clamp Recording	Functional	Real time current reading across the cell membrane corresponding to openings and closings of a channel; different configurations lead to different inferences about the cell along with different signal to noise ratios.

1.2.1 Structural study

Structural study involves methods that aim to resolve the 3D molecular configuration of an ion channel. This provides detailed information about the organization of proteins and protein domains within the ion channel. However, a significant limitation is that this work is typically based on ion channels in a static state. Understanding the structure of an ion channel can provide valuable insights into binding sites, pore selectivity, and more. This information can be utilized to design further functional analyses and experiments.

1.2.1.1 X-Ray Crystallography

In X-ray crystallography, high frequency X-rays are fired at a crystalline sample of a protein, that then diffract onto sensors that are then processed to infer information about the structure of the protein within.

X-Ray crystallography has been used (among other applications) to gain insight into the structure of transient receptor potential channels (TRP) (Singh, McGoldrick, Saotome, & Sobolevsky, 2018), which are a family of ion channels that work as sensors to a number of different stimulus including temperature, voltage, as well as some molecules such as menthol and capsaicin. X-ray crystallography produces a high resolution structural image of the ion channel, allowing researchers to understand how the ion channel operates conforms in response to external stimuli.

1.2.1.2 Cryo-electro microscopy

Another approach is cryo-electro microscopy; instead of using photons (light) as with a traditional microscope, electron guns are used to bombard a sample with electrons that then reflect off the sample and onto a fluorescent screen to produce an image. By freezing the sample at low temperatures, the structure of the sample is not degraded as it is exposed to the electron beam; this allows for far higher quality images with a lower sample size than other imaging methods (Milne et al., 2013).

Cryo-electro microscopy has also been used to analyse the structure of the TRPV2 channel (Zubcevic et al., 2016) to further understand the mechanisms as to how they work; such as the selectivity filter and pore size; as cryo-electro microscopy and X-ray crystallography have different approaches, they come with advantages and disadvantages over one another that typically dictates which one is used; however with recent advances in both methods, this gap is becoming smaller and smaller.

Traditionally, X-ray crystallography has a higher resolution than cryo-electro microscopy, however this is offset by the fact that larger protein structures are usually harder to crystallise. X-ray crystallography normally requires a higher yield of protein to work effectively, although better isolation methods are making this less of a problem.

It's important to note that cryo-electron microscopy is used to study the structure and physical characteristics of ion channels. While it can be utilized to explain and theorize drug actions, it is not as commonly used to measure their effects compared to other methods, such as patch-clamp electrophysiology. This is primarily because it is less effective in measuring dynamic temporal changes in protein conformations.

1.2.2 Functional Study

1.2.2.1 FRET

Förster resonance energy transfer (FRET) is a florescence spectroscopy imaging technique that is used in analysis to measure the opening and closing of ion channels by deriving the distance between molecules (Martinac, 2017). This is achieved by measuring the energy transfer between fluorescent donor molecules and acceptor molecules and measuring the amount of energy transferred. By measuring this over time, we can see real-time opening and closing of a protein by seeing the measured difference jump between values.

FRET, like all experimental techniques, has its own significant drawbacks; it is notable for having low signal to noise ratio and is extremely sensitive to changes in recording conditions such as temperature or pH (Leavesley & Rich, 2016; Wang et al., 2014).

1.2.2.2 Patch Clamp Electrophysiology

Patch clamp electrophysiology (Neher & Sakmann, 1976; Sakmann & Neher, 1984) is the "traditional" method for recording and analysing the behaviour of ion channels over time. It works by creating an electrical circuit through a bath solution and a small pipette (also filled with a solution) that is lowered onto a cell (Figure 1.4). As the pipette touches the cell, suction is applied to form a seal around the cell membrane (and an ion channel), with the resistance in the circuit rapidly increasing as the current must cross the cell membrane to enter the pipette. After a giga-ohm seal is achieved, the current can be recorded over time at different voltage levels to measure the ion channel activity over time; as the channel opens and closes the resistance across the cell membrane changes resulting in relatively large jumps and falls in the measured current compared to the noise.

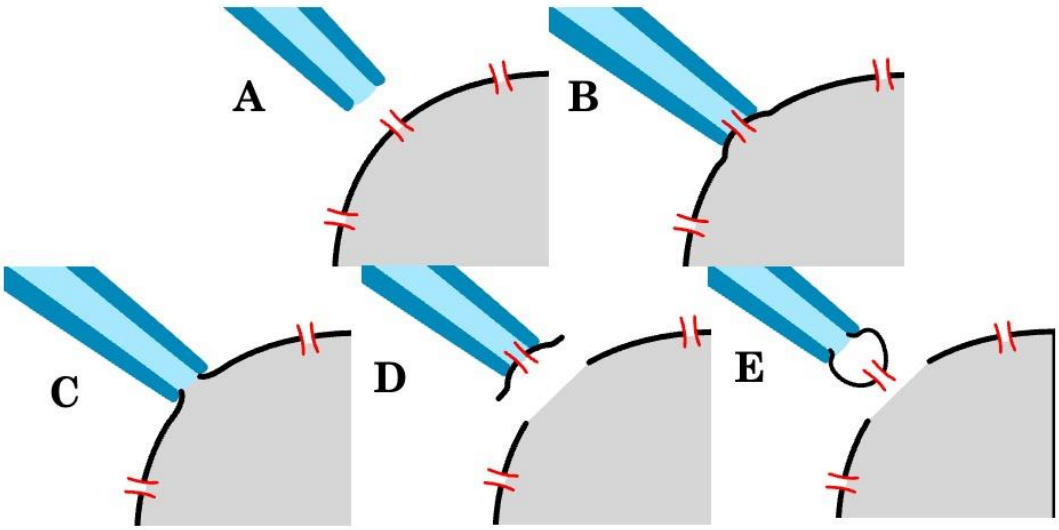


Figure 1.4: Different stages to the patch clamp process.
From top left to bottom right: A shows a sharp pipette being lowered onto the cell membrane. B shows the pipette making contact with the cell membrane with light suction being applied to create a gigaseal and allowing for cell attached patch recording. From B, whole cell recording (C) can be achieved by applying further suction; or an inside out configuration (D) can be reached by lifting the pipette away from the cell membrane. Finally, outside out (E)

recording can be reached by lifting the pipette from the cell membrane when in a whole cell setup. Patch clamp electrophysiology is a key technique in measuring the effects of substances on cell membrane activity.

Table 1.5: Different configurations of patch-clamp electrophysiology.
Patch-clamp electrophysiology has a number of different methods that can be used in experimental design to allow for changing the concentration either inside or outside of the cell; for example some pharmaceutical agents work extracellularly, and so the right method of patch-clamping must be chosen so this can be replicated in experimental conditions.

Type of Patch	**Number of Channels**	**Notes**
Cell Attached	Few	The original patch clamp method, fairly stable but does not allow access to the intracellular solution. Inability to change the extracellular solution in the pipette mid-experiment can lead to design problems as each cell can only have one point on a dose response curve.
Whole Cell	Many	Typically allows access to intracellular solution via pipette solution, but is technically very difficult to change mid-experiment Capturing entire cell electrical activity at one time has advantages and disadvantages.
Inside Out	Few	Easy access to intracellular solution via bath, and typically has higher signal to noise ratio than other methods. However rupture of membrane results in a much more instable configuration.
Outside Out	Few	Allows easier access to extracellular solutions via bath (rather than by pipette in cell

		attached;attached); however due to additional steps may be considered more difficult.

There are several distinct types of patch-clamp electrophysiology recording configurations, each prepared differently and thus requiring a variety of analysis techniques for data interpretation. These different approaches are summarised in Table 1.5, with several being utilised in this thesis (refer to Chapter 8). The simplest among these is the **'cell-attached patch**', which begins recording immediately upon achieving a giga-ohm seal. This method records a limited number of ion channels due to the resistance being over a small section of the cell membrane, and typically has a low signal-to-noise ratio as the patching area remains attached to the rest of the cell. Additionally, both the bath and pipette solutions represent extracellular conditions with respect to the channel due to the configuration of the solutions.

By pulling the pipette back from the cell and removing a patch of membrane, a subset of ion channels can be physically isolated from the rest of the cell membrane. This configuration is known as '**inside-out recording**' and is the first of two 'cell-free' patch techniques. As the membrane is now detached from the rest of the cell, this technique usually results in a significantly higher signal-to-noise ratio than cell-attached patch, while still recording either a single or a small number of channels. However, one key difference is that the bath solution now represents the intracellular solution, as the cell has been ruptured and exposed to the bath solution.

Alternatively, instead of lifting the pipette away from the cell, additional suction can be applied to the cell to rupture the cell membrane and open the cell's interior to the pipette solution. This is known as "**whole-cell patch clamp"** and is a common way of analysing ion channels due to the fact that it now results in measuring the current from all ion channels in the cells, rather than just the patched region. This configuration typically has a lower signal to noise ratio but allows for a more holistic approach to analysis by

considering large-scale responses to changes in conditions. One key advantage of this method is that since the pipette solution is now at one with the interior of the cell, but the bath solution is still on the outside, it is relatively simple to measure the effects of a drug on the behaviour of ion channels by changing the bath solution and examining any behavioural changes in the current.

Whole cell recording gives a holistic approach to ion channel analysis while also allowing for swapping extracellular solutions to measure the effects of different conditions on ion channel behaviour. However, since it *records all the channels in the cell*, which could be a mixed population of ion channel phenotypes and genuine single-channel analysis is not possible. Generally, the low signal to noise ratio can limit the types of analysis used downstream.

The fourth and final standard mode of patch-clamp recording is the '**outside-out patch clamp**'. This method *first requires the formation of a whole-cell recording*, followed by pulling the pipette away from the cell, which tears off a small section of the membrane. Due to these mechanical manipulations, the extracellular face of the membrane now faces the bath (similar to the whole-cell patch clamp), and the cytosolic face of the membrane now faces the inside of the patch pipette. This configuration is technically more challenging to achieve than others as it involves several steps of manipulating a delicate cell membrane. However, it offers advantages such as allowing for single or low-number channel recording with a high signal-to-noise ratio. It also provides the ability to change the bath solution to measure different drug effects under various conditions.

1.2.2.3 Automatic Patch Clamp Arrays

Due to the delicate and complex nature of the patch clamp process, substantial efforts have been made to automate the recording of ion channel activity. This has led to the development of automated patch clamp machine equipment (Figure 1.5) that can "patch" an array of cells simultaneously, significantly increasing throughput and, in more recent years, accuracy (Wood, Williams, & Waldron, 2004). For instance, the Sophion Qube 384

(Chambers, Witton, Adams, Marrington, & Kammonen, 2016) boasts a throughput of 384 patches simultaneously with a success rate of 93% for sealing. This results in large datasets for analysis (which are continuing to increase) and alleviates the burden on trained researchers to perform this labor-intensive work. Typically, these "parallel" recording devices focus on whole-cell recording. While single channel recording is often possible (personal communications), analyzing these large datasets would be impractical. Whole-cell recordings do provide a holistic view by examining an extremely large number of channels at once. This approach offers insights into the overall function of a cell's channels in response to changes in conditions. However, one disadvantage is the lack of resolution; since a high number of channels are being measured simultaneously, there is limited scope for examining the function of individual channels. On the other hand, single-channel recordings offer a more isolated view of channel activity. This can yield more information than whole cell recordings if sufficient data is collected. The main bottleneck for single channel work is the idealization stage - converting the noisy signal into a series of open and close events. Although automatic patch clamp apparatus has significantly increased the throughput for gathering single channel data, we still lack the necessary tools to harness this data effectively. Current workflows for idealizing and analysing data are either not accurate enough or require too much researcher input to be feasible at the scale high-throughput automatic patch clamp require.

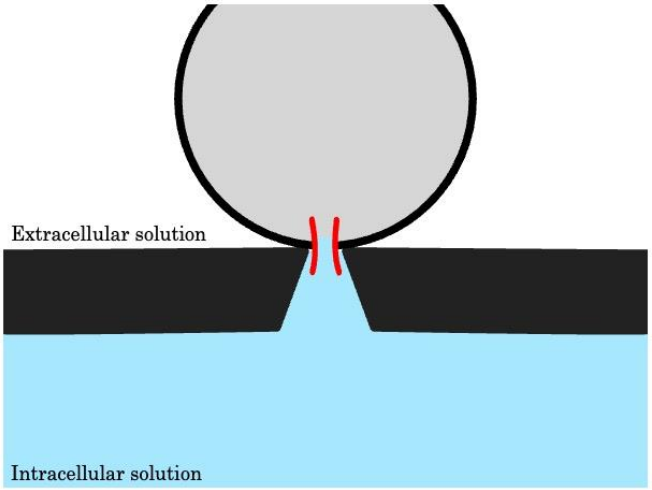


Figure 1.5: Diagram of how a array based automatic patch clamp system works.
Automatic patch clamp systems allow for a far greater throughput for recording data; cells are put on a plate with many micropores beneath, and suction is applied to the entire plate. Some cells will then form a gigaseal with these holes for either cell attached patch or whole cell recordings. Although not all of the cells seal successfully, the automatic nature of this method allows for a greater amount of data to be produced as hundreds of cells can be patched at once.

This thesis focuses solely on single channel analysis - where recordings have visible openings and closings of single channels. This allows for an exquisitely detailed examination of the function and behaviour of these individual channels.

1.2.2.4 Current Tools for Patch Clamp Signal Analysis

Automatic patch clamp arrays provide a clear route to recording large scale datasets for ion channel data; however the analysis tools are still lacking in both accuracy and usability for researchers. As more data is generated, any time recovered for researchers in the lab is lost at the analysis stage where all existing models need initialisation or supervision to operate properly.

One of the most common pieces of software for ion channel idealisation (the process of denoising a raw ion channel signal into a square-like wave of opening and closings) and analysis is the QuB (Nicolai & Sachs, 2013) software suite that utilises the segmental k-means algorithm (SKM) to perform idealisation (Qin, 2004). The segmental k-means

algorithm works by iteratively fitting a hidden Markov model onto the data that has been denoised by a clustering process. The Markov model's parameters are then used to improve the cluster process over time in a recursive manner until the desired idealisation is achieved. There are a number of drawbacks in this work due to the nature of the analysis pipeline; it is very uncommon to see ion channel data not have "baseline drift;"; that is, the current slowly starts to move away from the zero point on the axis, and in addition the noise terms seen in ion channel records are rarely Gaussian (in fact, studies have suggested a larger superset of noises called 1/f noise to be responsible (Bezrukov & Winterhalter, 2000)). Both of these problems are fundamental to the model's performance; baseline drift will cause the clustering step of the segmental k-means algorithm to not identify clearly separable clusters, and the incorrect noise in the simulation raises questions about how other artifacts typically present in ion channel recordings would affect the output.

Another algorithm using a jump segmentation multiresolution filter (J-SMURF) (Hotz et al., 2013) promises idealisation with no a-priori knowledge about the Markovian network or channel dynamics, robustness against non-Gaussian noise and baseline drift, and high accuracy on both synthetic and lab recorded data. This model works by detecting steps at different resolutions within the data to build a lower resolution description of the signal, then applies a threshold algorithm to this new signal to find the clear events. The data used in this work however is of a very high quality with long, well defined events clearly visible within the recording; there are very many ion channels that do not exhibit this behaviour and a successful model needs to resolve smaller channels with much lower dwell times. Indeed, others have commented on this drawback (Gnanasambandam et al., 2017) and have found that this model is computationally expensive.

Similarly, the minimum description length algorithm (MDL) (Gnanasambandam et al., 2017) is an algorithm very similar to J-SMURF in that it attempts to find a lower resolution description of the data, but does so in a slightly different way. It shares many of the same advantages of J-SMURF; no a priori knowledge of the underlying Markov

model is required, however proves to be more robust against shorter events with a lower signal to noise ratio. Again however, all the experiments in the MDL paper were carried out with synthetic data, with the model performing best on Gaussian noise.

Finally, DeepChannel (Celik et al., 2020) approaches the problem using deep neural networks using "semi synthetic" datasets (that is, synthetic datasets passed through recording apparatus identical to that used in a lab. The model is much more computationally expensive than the previous models, but performs far stronger than both MDL and QuB's SKM implementation in end-to-end tests (Celik et al., 2020). This research showed that deep learning was applicable to the problem of single channel analysis, however this also opens exploration to different model architectures; preprocessing methods and other problems in the field that were not covered in the previous work.

The heavy use of synthetic data for evaluation of models is not without good reason; for ion channel recordings, artificially creating the data from a known sequence of open and closed events is the only way to get an absolute ground truth. For models evaluated on lab recorded data, the labels are provided by expert analysis, which is not infallible. It is extremely common to have artifacts present in a recording that are hard to distinguish from real events, and visa-versa. Furthermore, the use of Gaussian noise in place of more complex noise terms in simulated data comes from a continuous developing understanding of the noise present in ion channel signals. Further work into improving models using non-gaussian noise is ongoing, furthering the work on J-SMURF (Pein, Bartsch, Steinem, & Munk, 2021; Pein, Eltzner, & Munk, 2021; Pein, Tecuapetla-Gomez, Schutte, Steinem, & Munk, 2018); however the signal to noise ratio in these papers are still significantly higher to what is commonly observed in a lab environment. The original DeepChannel work approached this in a particularly novel way through recording the synthetic data through an analogue system which generates far more authentic noise than digitally generated noise. This method too is not without its drawbacks; the method has an extremely low throughput of data generation since data has to be passed through the

analogue amplifier in real time. Therefore, the synthesis of realistic ion channel data with a high throughput of production and realistic signal noise is somewhat of an open problem. In Chapter 3, I will focus on the development of methods for ion channel simulation. Moving forward to Chapter 8, I will adopt a distinct approach to "ground truth". Here, the "ground truth" is considered to be the comprehensive response of a channel to a drug, as opposed to a point-by-point ground truth.

1.2.2.5 Markovian Modelling & Kinetic Analysis

Not long after the development of the Hodgkin-Huxley model of squid axons (Hodgkin & Huxley, 1952), further discoveries were made to suggest that mechanism of the ion channel's function was more complex than simply one open state and one closed state (Armstrong, 1971). It is now thought that the behaviour of single channel activity is better described as a Markovian process, with the channel moving through many different hidden states that exhibit either open or closed behaviour (as well as some other underlying physiological characteristics).

By examining the dwell times of the idealised record, we can attempt to fit multiple models to the record. This will enhance our understanding of how the channel transitions through these states over time. This level of mechanistic detail could be instrumental in drug discovery. For instance, two different drugs might inhibit a channel to the same extent, but through distinct mechanisms. These mechanisms could only be discernible by variations in the resulting Markovian state. There are numerous approaches to recover the Markovian state. If the Markov model is known a priori, the Viterbi algorithm can be employed to determine the most probable sequence of states. If the structure of the Markov model (i.e. the number of open and closed states) is known but the rates not, the Baum-Welch algorithm can be used to fit the rates to the data statistically; a different approach is to use Monte Carlo modelling (Siekmann et al., 2011) to try and fit different parameterisations to the data stochastically using an additional discrete time Markov model whose equilibrium corresponds to the rate constants in the continuous time ion channel Markov model.

All of these methods require the idealised record as an input rather than the raw recorded signal; so the time-consuming idealisation step still remains. One area for exploration is if an end-to-end pipeline can be developed for prediction of Markov state from the raw signal. Since each hidden Markov state corresponds to only one of the visible states, this forms a "super-problem" of the idealisation problem; if the state prediction is correct then so too will be the open/closed prediction. In chapter Chapter 4 I will discuss methods which allow for canonical forms, and the challenges behind using them for deep learning; and in Chapters 5, 6 and 7 I will compare methods to recover Markovian structure directly from raw ion channel data using deep-learning methods.

1.3 Deep Learning

1.3.1 Discovery

Neural networks started with its roots in neuroscience, with the McCulloch-Pitts neuron (McCulloch & Pitts, 1943) being the first mathematical model of a neuron. At the time, these were mainly thought to be used as logic gates, taking in a number of Boolean values (0 or 1) and outputting a Boolean value depending on the kind of activation function (or in this case, gate) used. More complex systems could be built up by chaining these neurons together to make complex decisions. These activation functions were chosen in a way to achieve the desired effect and neurons had no weights associated with the inputs to train; therefore these models were not "trainable" as we know deep learning models to be now.

It would not be until the 1960s until the idea had another breakthrough; optimisation of weight parameters in a neural network to achieve optimal performance was a subset of a larger problem of general parameter optimisation which was experiencing a wave of progress due to nonlinear optimisation. Multiple authors are credited with discovering backpropagation and gradient descent, typically Bryson and Kelley for the first derivation (Bryson, 1961; Kelley, 1960) and Dreyfus for a much more elegant method using fundamental calculus (Dreyfus, 1962). Since computational resources were scarce at this time, the process of backpropagation could not be done automatically and was only

possible with small neural networks. Backpropagation is discussed further in section 1.3.3.3.

Figure 1.6: Sample images from the MNIST dataset. (Yann LeCun, 1998).
The Modified National Institute of Standards and Technology database (MNIST) was used as an initial dataset for measuring neural networks' performance at classifying different handwritten digits. It now forms a benchmark dataset that many deep learning architectures are compared against.

The field again entered a period of low activity, mainly due to the lack of computational power we have today. In 1980, Fukushima published a model (called a "neocognitron") inspired by the visual nervous system which is largely attributed as the start of convolutional neural networks today. In 1989 LeCun applied these concepts to recognising handwritten digits (LeCun et al., 1989), later developing the Modified National Institute of Standards and Technology (MNIST) (Figure 1.6) database of handwritten digits (LeCun, 1998) that are still used as a benchmark for image recognition today.

At this point, neural networks were slow, and in most cases performed worse than simpler methods. The networks required to solve complex problems were simply too large for computers at the time to handle; however through the 1990s and 2000s graphics processing units (GPUs) saw significant improvements in performances and thus deep learning began to become a viable option for machine learning applications. GPUs were developed for computer vision and image processing and allow thousands of simple operations to occur in parallel by housing thousands of individual processing cores. GPUs

can significantly speed up model training, because the computations involved in training neural networks (like matrix multiplications and additions) can be effectively parallelized.

Neural networks continued to become more complex, with the invention of recurrent neurons (Rumelhart, Hinton, & Williams, 1986) allowing for more compact representations of deep networks. As neural networks expanded in size, depth, and complexity, the "vanishing gradient problem" surfaced (John & Stefan, 2001), causing new challenges. This problem stems from the application of the chain rule during backpropagation. The trainable parameters at the onset of the neural network usually had gradients represented by a product of many small numbers (between 0 and 1). This gradually diminishing product led to the inability of deep networks to learn, particularly in long time series problems. However, this issue was mitigated with the advent of long short-term memory (LSTM) cells in 1997(Hochreiter & Schmidhuber, 1997). LSTM cells are designed to retain important information and "forget" unimportant information and are now a popular form of recurrent neuronal network (RNN)_widely used in for time series such as language modelling or time-series predictions (stock market for example).) (Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman, 2017; Xiao & Zhou, 2020).).

A persistent challenge, with deep learning models, is the law of diminishing returns associated with increasing model size. This issue was temporarily circumvented by the development of convolutional neural networks (CNNs). However, as we transitioned into the 2010s, there was a significant increase in the depth and number of parameters in these models, with state-of-the-art models regularly featuring millions of trainable parameters. During this period, U-Net (Ronneberger, Fischer, & Brox, 2015) and ResNet (He, Zhang, Ren, & Sun, 2015) emerged as notable advancements. They introduced "skip connections" to bypass the diminishing returns issue, however it is important to note that these architectures simply mitigated the problem rather than solving it completely.

In the 2010s, the idea also emerged, that multiple disconnected neural networks could be used in tandem to achieve more than the sum of their parts. In 2014, generative adversarial networks (GANs) (Goodfellow et al., 2014) were developed for synthetic image

generation, starting a revolution of media synthesis from known training sets (Figure 1.7). Work on generative adversarial networks continues with advances to mix different characteristic for an output (Karras, Laine, & Aila, 2019a) and fixing a particular feature of the generated output by supplying a label in the generation process.

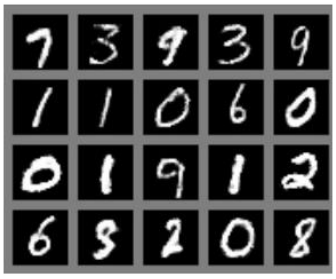


Figure 1.7: Some generated images from training a GAN on the MNIST dataset. (Goodfellow et al., 2014). *Generative adversarial networks (GANs) create new data from an existing dataset by training two models in competition with each other; a generator creates new samples and a discriminator determines if the new sample is from the generator or from the original dataset. After training, the generator's outputs are taken as the synthetic data; and if training is successful this new data should bear some similarity to the original dataset.*

1.3.2 Motivation

Deep learning has proven to be a significant tool in the field of image analysis, with many ground-breaking discoveries and landmark points in deep learning coming directly out of the ImageNet Large Scale Visual Recognition Challenge that ran from 2010 to 2017. Deep neural networks are designed to detect complex patterns in high dimensional data using large datasets, and images are a significant source of data for this application. High resolution time series signals are also a source of large, complex data that we can apply deep learning methodology to.

A particularly pertinent example of the application of deep neural networks to the field of time series analysis are the advances made in sleep stage time series analysis using deep CNN (Supratak, Dong, Wu, & Guo, 2017) applied to electroencephalogram (EEG) signals. This work is particularly striking as it aims to recover a series of unknown but

interrelated states from a signal electrical signal using the characteristic features of the input.

The original DeepChannel work already showed that a rather simple deep-learning model showed promise for tackling the single channel idealisation problem above, and so in this thesis I develop new models and rigorously test how far more sophisticated models can go.

1.3.3 Theory

1.3.3.1 Neurons and Functions

Neural networks can have a variety of structures, however it is popular to discuss the architectures in terms of "layers"; that is, collections of neurons and functions that operate on the previous layer. There are several different common types of neurons and functions used in deep learning, all with different applications and purposes; those that we use in this work are explained below.

1.3.3.1.1 Dense Layers

The simplest type of neuron takes in several inputs and weights (and typically a bias term), passes them through an activation function and outputs the result to either the next layer or an output (Figure 1.8). In mathematical terms, this can be formulated by the equation:

$$f(\mathbf{x}, \mathbf{w}) = g(\mathbf{x}^{\mathrm{T}}\mathbf{w} + b)$$

Where $\mathbf{x}$ is the vector of inputs, $\mathbf{w}$ the vector of weights, b the bias term and g the activation function.

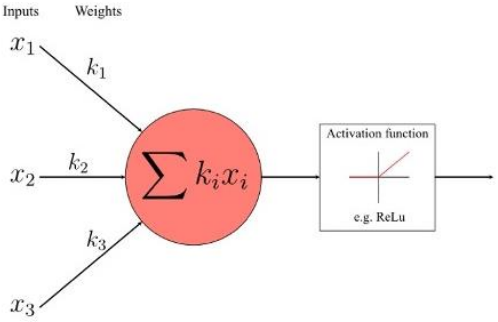


Figure 1.8: An example of a simple model neuron in an artificial neural network.
The inputs are multiplied by trainable weights, then the results are added together before being passed through an activation function to the output. A bias term is added to the neuron before the activation function to avoid a neuron from having a weight vector of o – causing the backpropagation step to fail entirely. These simple neurons become the building block of large neural networks.

Dense layers are made up of these simple neurons and fully connect to every neuron in the previous layer, and output to every neuron in the next layer. This was the method used in the first document recognition paper (LeCun et al., 1989); since the number of inputs (one for each pixel) was fairly low, the number of parameters that come from a small neural network is still manageable. For larger scale models with higher resolution data, this becomes unmanageable as the number of parameters between two layers is equal to the product of the number of neurons in each layer (Figure 1.9). For example, for 1 second of 10kHz ion channel data, and 1000 neurons in the first hidden layer, we get 10 million parameters just from the first layer; this causes the model to be extremely large and slow, and nfeasible for our use case.

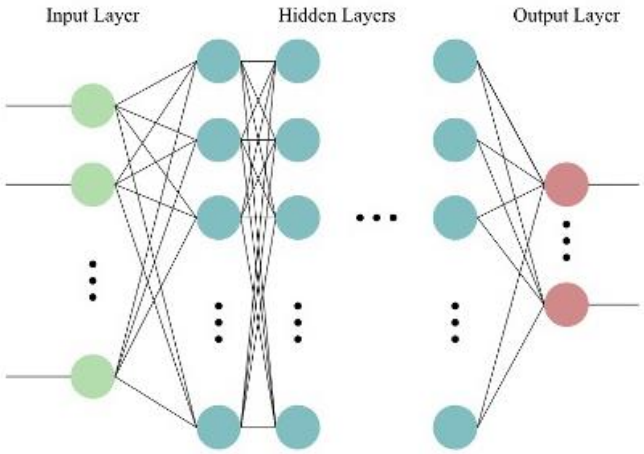


Figure 1.9: An example of a fully connected deep neural network.
Each line represents a weight, with the output to each neuron becoming an input to all the neurons on the next layer. This can become computationally expensive, as models where the number of neurons in a layer is increased, the rate at new connections are needed increases exponentially. Typically this kind of fully connected layer is used in conjunction with other neural network mechanisms (after the input dimension has been reduced significantly) to reduce the number of parameters needed.

1.3.3.1.2 Recurrent Layers

For temporal data such as time series data or signal analysis, traditionally the approach is recurrent cells. A simple recurrent cell simply takes in the output of the previous data point and uses at as an input to the current calculation, rather than being independent to the previous output. This recurrent nature encodes all previous activity within the calculation:

$$f(\mathbf{x_t}, \mathbf{w}) = g(\mathbf{x_t^T w} + \mathbf{x_{t-1} v} + b)$$

Where $\mathbf{v}$ is the recurrent weight parameter scaling the output from the previous output $\mathbf{x}_{t-1}$.

The vanishing gradient problem (explained earlier) heavily affects recurrent neural networks, and there have been two main cells designed to remedy this problem: Long LSTM cells and Gated Recurrent Unit (GRU) cells (Figure 1.10).

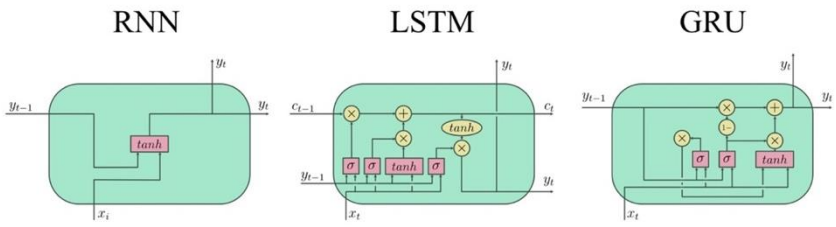


Figure 1.10: The construction of different recurrent cells.
Recurrent cells use the previous output of the last sample as an input to the next sample; this allows for networks to better understand historical time-series data. The RNN (left) only takes the last raw output in as an input, whereas the LSTM incorporates a variable "memory" output that controls how much past data effects the next output. A GRU still only takes in the output of the last cell as an input, but uses it to inform the output of the current calculation in more detail.

LSTM cells (Hochreiter & Schmidhuber, 1997) work by introducing a "forget gate" that propagates historical data within the cell up to a certain, trainable point, at which it discards the information.

The mathematical construction of LSTM cells is fairly complex; however much easier to understand from an intuitive standpoint:

$$\boldsymbol{f_t} = \text{sig}(\mathbf{x}_t^T\mathbf{w_f} + \mathbf{h_{t-1}}\mathbf{v_f} + \mathbf{b_f})$$

$$\boldsymbol{i_t} = \text{sig}(\mathbf{x}_t^T\mathbf{w_i} + \mathbf{h_{t-1}}\mathbf{v_i} + \mathbf{b_i})$$

$$\boldsymbol{o_t} = \text{sig}(\mathbf{x}_t^T\mathbf{w_o} + \mathbf{h_{t-1}}\mathbf{v_o} + \mathbf{b_o})$$

$$\boldsymbol{c_t} = \boldsymbol{f_t} \circ \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \circ \tanh(\mathbf{x}_t^T\mathbf{w_c} + \mathbf{h_{t-1}}\mathbf{v_c} + \mathbf{b_c})$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \circ \tanh(\boldsymbol{c_t})$$

Where x is the input vector; $w_f, w_i, w_o, w_c, v_f, v_i, v_o, v_c$ are weight vectors; b_f, b_i, b_o, b_c the bias vectors; sig, tanh are the sigmoidal and hyperbolic tangent activation functions respectively, and $\circ$ representing the element-wise product operator.

Intuitively, the first 3 lines of the formulation are just standard neuron activations as in the simplest case with sigmoidal activation functions. The critical part of the operation is in the final two lines; here $\boldsymbol{f_t}$ is the "forget vector" that tracks whether or not to use the information from the last input (encoded through $\boldsymbol{c_{t-1}}$) in the next output.

Gated recurrent units work in a fairly similar way, with two familiar looking neurons being combined in a way that balances past outputs with current outputs. The formulation of this is as follows:

$$\boldsymbol{f_t} = \text{sig}(\mathbf{x}_t^T\mathbf{w_f} + \mathbf{h_{t-1}}\mathbf{v_f} + \mathbf{b_f})$$

$$\boldsymbol{i_t} = \text{sig}(\mathbf{x}_t^T\mathbf{w_i} + \mathbf{h_{t-1}}\mathbf{v_i} + \mathbf{b_i})$$

$$h_t = f_t \circ \tanh(\mathbf{x_t}\mathbf{w_h} + (i_t \circ h_{t-1})\mathbf{v_h} + \mathbf{b_h}) + (1 - f_t) \circ h_{t-1}$$

Again, these equations are not that helpful in immediately understanding how gated recurrent units work; but they operate in a similar way; in this case the forget gate f_t creates a "balancing mechanism" in the final equation that forms a weighted, transformed average between the previous output and the current data.

1.3.3.1.3 Convolutional and Deconvolutional Layers

Convolutional layers were inspired by models of the visual cells in the eyes in that there is a notion of "localised focusing" at a neuron level that builds a pattern map of the input based on a trainable kernel to build an overall encoding of different features (Hubel & Wiesel, 1962).. Similarly to fully connected layers, convolutional layers increase in abstraction the deeper we get into a model; for example in image processing, typically the first few layers are detecting edges and simple shapes, whereas the final layer will be detecting eyes, noses or mouths.

Convolutional layers result in down sampling of the input data depending on the kernel size and "stride" (that is, how the kernel "scans" over the input data). This is helpful in a number of cases (such as image classification) as reducing the size of the image (and hence

number of parameters at the final stage) is highly beneficial. However, for problems where up-sampling is required (for example increasing the size of an image while filling in missing pixels intelligently), convolutional layers are not appropriate. In this case, deconvolutional layers (or transposed convolutional layers) allow for the convolutional process to occur while upscaling the image by adjusting the padding within the input data. It should be noted that the term deconvolutional layers is a misnomer; it does not apply the inverse of the convolutional function but rather the convolutional process in a slightly different way.

Convolutional and deconvolutional layers work via very similar means to achieve down-sampling and up-sampling respectively. Typically, two dimensional layers are used for image analysis, but the principles can be extended to any number of dimensions such as one-dimensional time series data or three-dimensional spatial scans.

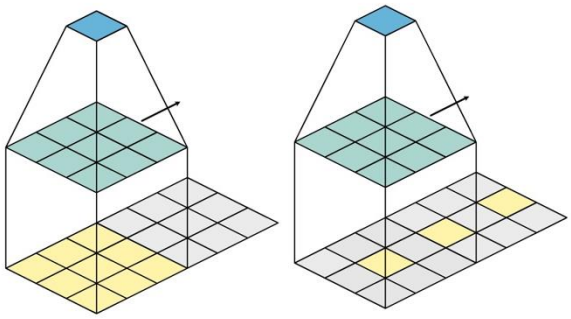


Figure 1.11: Diagram of Convolutional and deconvolutional layers.
Typically used in image analysis, convolutional (left) and deconvolutional layers (right) work by passing a series of trainable kernels (green) over an image, taking the sum of the pairwise product of the kernel and each submatrix of the input. As these kernels "scan" across the image, features such as edges are detected, and "feature maps" are built as the network increases in depth. By adding zero padding between each input, the downscaling effect of the scanning process becomes upscaling, which can be used in generation instead of detection.

At its core, the process involves using a kernel K of a chosen size with trainable parameters that passes over the input and is multiplied by a sub-matrix of the input X - the size of

this submatrix, and the order of operations dictates whether downsampling or upsampling occurs.

In down-sampling, each element $y_{i,j}$ of the output matrix Y is given as follows - for simplicity here the kernel size is (3×3) and the stride is 1. Figure 1.11 shows visualisations of the areas chosen and the mappings. they correspond to.

$$y_{i,j} = b_{i,j} + \sum_{c=1}^{3} \sum_{d=1}^{3} K_{c,d} \cdot X_{(i+c-1),(j+d-1)}$$

The dimensions of the kernel are adjusted by changing the bounds of the sums, and the indexes of the input matrix (the -1 terms in the indexing of X become $-\lfloor s/2 \rfloor$, where s is the desired kernel size). A stride can be applied in either or both directions by incrementing i and/or j in values other than one, and simply ignoring the resulting "gaps" in the output matrix. Typically, when either of the indexes of X are outside of the bounds of X (for example, when the index is negative), the value is taken as zero (this is referred to as padding).

1.3.3.2 Other Layers and Tools

1.3.3.2.1 Max Pooling Layers

In convolutional models, convolutional layers with no stride reduce the size of the output by the size of the kernel minus 1. For small images, this can work, as the size towards the end of the model gets small enough to simply apply fully connected layers without the number of parameters exploding in size. For larger images, the decrease in image size over time may not be fast enough to make a model feasible (Figure 1.12). For example, in a image that is 1000 pixels wide and high, using a 3x3 kernel will reduce the size by 2 each layer; this will not make a significant impact on the image size and thus we have a problem getting the output classification of the image. One solution is to use a “stride.”. Instead of “scanning” the kernel over the image pixel by pixel, we skip a number of pixels

equal to this parameter each time. Another approach is to insert "max pooling" layers that dramatically down-sample an image in a fast but effective way.

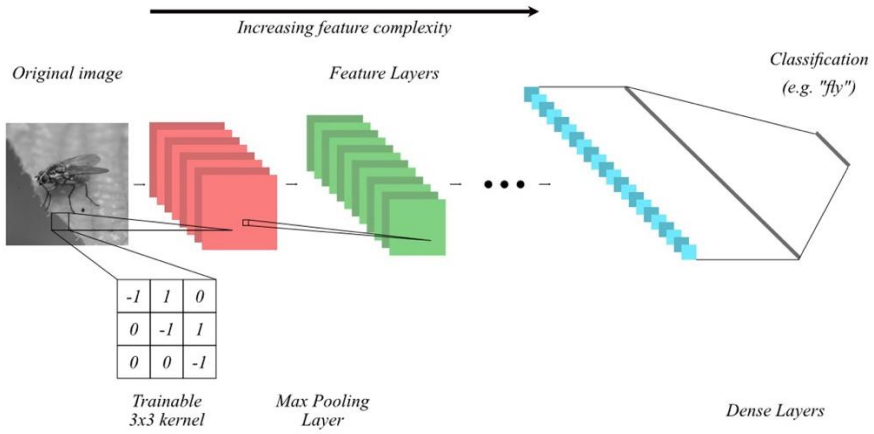


Figure 1.12: A basic diagram as to how convolutional neural networks work.
Trainable convolutional kernels (or filters) detect features in the input, such as edges, and pooling layers reduce this down further to smaller images. As the model depth increases, the complexity of features does as well, so whilst the top layer kernels might represent edge detection, the deep kernels might represent face or eye detection.

Max pooling layers achieve down sampling but are far simpler than convolutional layers. A set of $n \times n$ submatricies are taken, and for each one the maximal value of each submatrix is passed to a new, smaller submatrix (Figure 1.13). For example, when $n = 2$, the output matrix will be of size $\frac{n}{2} \times \frac{n}{2}$. This quickly and cheaply achieves down-sampling but does not recognise patterns as strongly as convolutional layers.

Minumum and average pooling layers exist; all of these serve similar purposes using slightly different manipulations.

1.3.3.2.2 Dropout Layers

Dropout layers are a popular regularisation technique first introduced by Hinton et al (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) designed to stop a model overfitting. They work by randomly disabling a *selected proportion of the* neurons. Dropout is only applied during training; during testing and deployment, the dropout is

turned off so that the entire network is now involved. The idea is that individual neurons will not get locked onto specific input features.

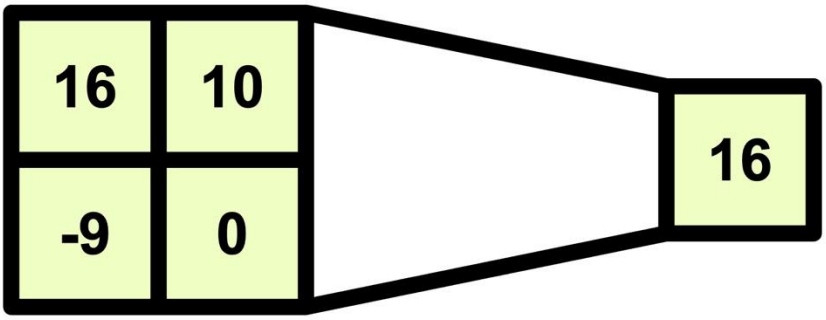


Figure 1.13: Example of a Max Pooling layer in action.
Pooling layers are a non-parameterised layer mainly used to reduce the size of an image so later layers won't need as many parameters themselves. Typically used is a 2x2 max pooling layer, taking in each set of 2x2 submatricies of the input matrix and outputs a new matrix of the maximum value in each 2x2 submatrix.

1.3.3.2.3 Batch Normalisation

Batch normalisation is another regularisation technique. In theory, a layer's parameters are adjusted based on the assumption that the previous layer's parameters are static; however this is clearly not the case; since those parameters are trainable as well. A batch normalisation layer scales the output of a layer to have a mean of 0 and standard deviation of 1 for each batch to adjust the input and force this assumption to be true. It is said that normalising each layers' input improves the efficiency of training, and also reduces overfitting in analogous way to Dropout (above).

1.3.3.3 Back Propagation and Gradient Descent

When training deep learning models, a loss function L is provided for example aa mean squared error for regression problems or cross-entropy for classification problems. We want to find a set of weights in the network that minimises this loss function. To find this

minimum, we look at the derivative of the loss function with respect to all the weights in the model. The complexity of the problem comes from the deep interconnectedness of different neurons and activation functions; calculating these derivatives is non-trivial and relies heavily on the chain rule in multivariate calculus.

For example, take the very simple example with 2 neurons.

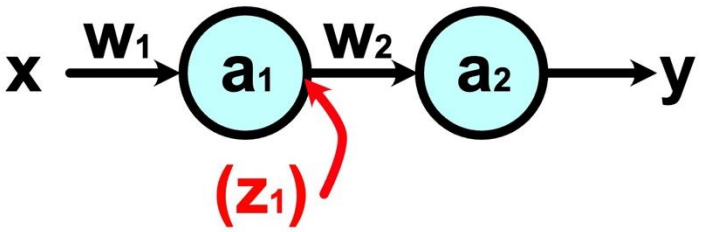


Figure 1.14: Simple backpropagation example.
In this case, blue circles are simple neurons, with weights w and activation functions a. For this example and the next, we always say that z_i is the output of the activation function a_i. Backpropagation is a method that collectively changes the weights w to minimise a loss function (such as mean squared error) by calculating the derivative of the loss function with respect to each weight, and using a gradient descent algorithm to adjust the weights towards the minimum.

A loss function L measures the difference between our output and ground truth labels, and we let J be the average loss over a batch (the number of samples the network sees before adjusting weights, the size of which is chosen by the user). Then we get:

$$J = \frac{\sum L(\hat{y}, y)}{n}$$

Therefore we want to find the gradient of J with respect to each of the weights. For Figure 1.14 as an example, with w_2 this is fairly straightforward; we know the activation function a_2 is differentiable (we choose it in model design), so by considering the dependencies of each function we get:

$$\hat{y} = a_2(w_2 z_1)$$

$$\frac{dJ}{dw_2} = \frac{dJ}{d\hat{y}}\frac{d\hat{y}}{dw_2}$$

For w_1 this is slightly more complex since we have several activation functions to go through:

$$\frac{dJ}{dw_1} = \frac{dJ}{d\hat{y}}\frac{d\hat{y}}{dz_1}\frac{dz_1}{dw_1}$$

Of course, this is the simplest case. For example, in a fully connected case (Figure 1.15):

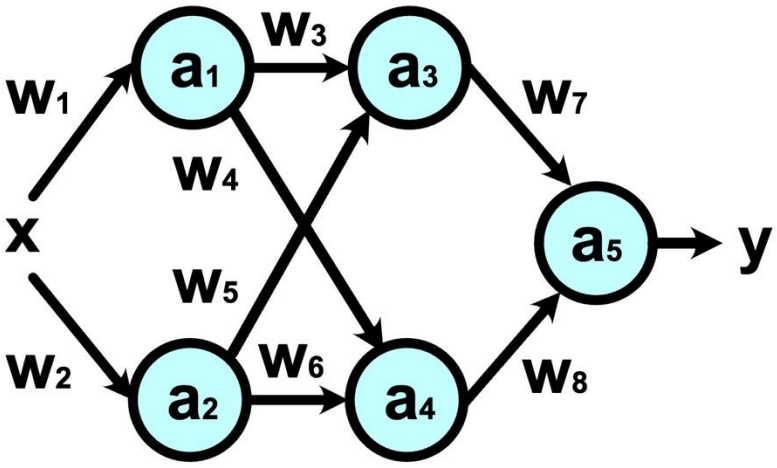


Figure 1.15: : More complex backpropagation case.
In larger neural networks - the derivatives become multivariate and harder to compute by hand. Therefore, high processing power is needed to compute the millions of gradients in large models. This is usually achieved by leveraging graphics processing units (GPUs) as their underlying processing architecture is more equipped for tasks of this kind.

We get:

$$\frac{dJ}{dw_1} = \frac{dJ}{d\hat{y}}\left(\frac{d\hat{y}}{dz_4}\frac{dz_4}{dz_1}\frac{dz_1}{dw_1} + \frac{d\hat{y}}{dz_3}\frac{dz_3}{dz_1}\frac{dz_1}{dw_1}\right)$$

This forms the basis for gradient descent; there are many gradient descent algorithms but in the simple case we take the weights and change them using the gradients calculated above with amount according to some weight α (the "learning rate;"); we do this since J is

not static during training; it changes for each new batch (and therefore so do all the derivatives calculated). So for w_2 above, we change the new value to:

$$w_2 - \alpha \frac{dJ}{dw_2}$$

Then we repeat the process until we reach the end of our dataset. Then we can continue training from the start again (one loop is known as an "epoch;"); or stop training when desired.

The problem of exploding or vanishing gradients appears from this process; in the recurrent case (Figure 1.16); if we let $\dot{z}_2$ (not to be confused with the derivative of z_2 with respect to time) be the previous output of the neuron, we get (since $\dot{z}_2$ is dependent on w_3 also):

$$\hat{y} = a_2(w_2 z_1 + w_3 \dot{z}_2)$$

$$\frac{dJ}{dw_3} = \frac{dJ}{d\hat{y}} \frac{d\hat{y}}{da_2} (\dot{z}_2 + w_3 \frac{d\dot{z}_2}{dw_3})$$

However, $\dot{z}_2$ is still a function of w_3 and its previous value $\ddot{z}_2$- this creates an infinite product in the calculation of the derivative. If the weight is small, this will tend towards 0; and if this weight is large, it will tend towards infinity as time goes on.

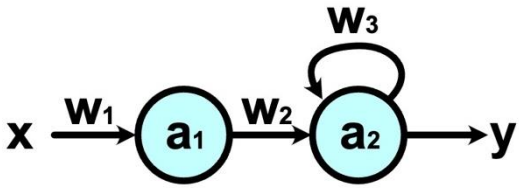


Figure 1.16: Backpropagation in a recurrent neural network.
Recurrent neural networks are a common type of neuron used for time series tasks, using the previous output as an input for the next calculation. For the backpropagation algorithm, this creates a unique problem; due to the recursive nature of the neuron at a_2, if we calculate the derivatives of the loss function with respect to w_3 we get an infinite series. This can cause a training problem known as exploding or vanishing gradients; where large or small gradients cause irregular training (if the gradient is large) or training to stop (if the gradient is small). This is the direct reason for the use of LSTM or GRU cells, to truncate this sum and avoid this problem.

By using a “forget gate” as in the Long Short-Term memory or Gated Recurrent Unit cells; we truncate this product so the gradient does not spiral out of control as the number of historical outputs increases.

1.3.3.4 Model types and Architectures

There is a wealth of different deep learning model types currently used for a number of different problems such as text synthesis, sound classification or regression; and a similarly large number of model architectures for each of these problems. One of the most common problems for deep learning models is the problem of classification; these models usually take in a number of inputs and output a single number representing a class that has been encoded from the training set. MNIST is an example of this problem where a set of images are all given numbers depending on the handwritten digit represented in the image, and these labels are used to train a digit recognition model. For simple problems like the MNIST dataset, a simple, shallow convolutional neural network will suffice to achieve high accuracy in obtaining the correct labels for each image, however for more difficult tasks with larger images or more complex problems, a larger model is typically used.

As models get larger, it has been found that adding additional layers give diminishing returns on accuracy (He et al., 2015). Therefore, two important models; ResNet and UNet were developed for slightly different purposes that aim to solve this problem in similar ways.

For image classification, ResNet (Figure 1.17) works by introducing so-called “skip connections” that pass outputs from early in the network to deeper layers directly, skipping intermediate layers; this provides a “short circuit” for the original image to inform the deeper layers, avoiding over abstraction deep in the network; ResNet is discussed further in Section 4.

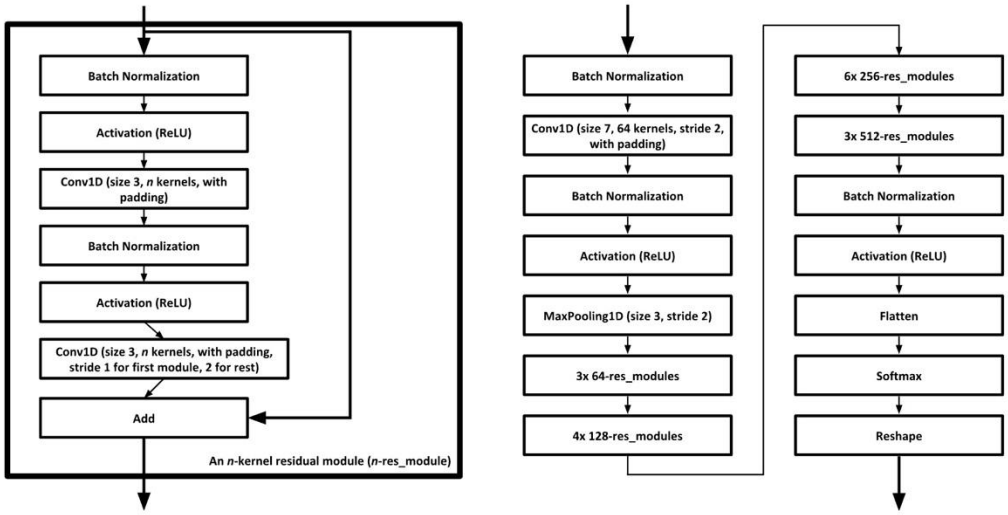


Figure 1.17: Modified ResNet Model Diagram for time series analysis.
The ResNet neural network architecture allows for deeper, larger models without getting diminishing returns by using "skip connections" that pass early outputs directly on to later inputs. ResNet has been shown to achieve best-in-class results for tasks such as image classification, due to the depth of the model achieving high levels of abstraction of complex images. Here, the residual module diagram (left) makes up the building blocks of the larger neural network model diagram (right). "3x" means that residual module is repeated 3 times.

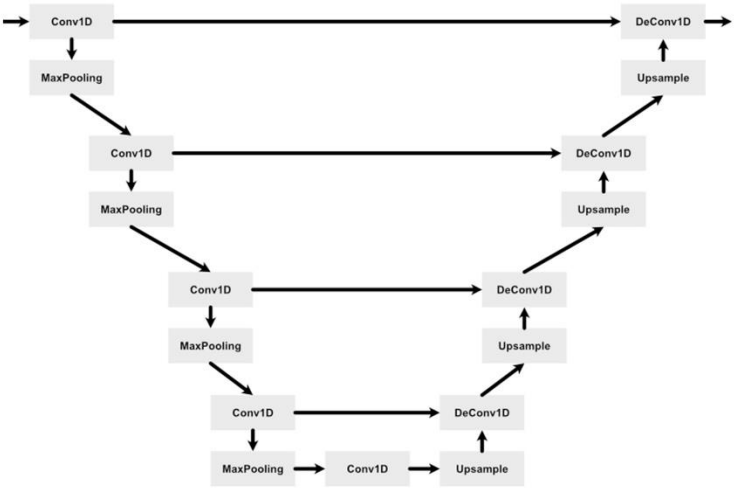


Figure 1.18: Diagram of the UNet architecture, adapted for time-series work.
The U-Net model is typically used for image segmentation, and works similarly to the UNet model by using skip connections to pass information directly further down the model than just the next layer. The model first uses a series of convolutional layers to downsample the image into a large number of small kernels (similar to an auto-encoder); then uses deconvolutional layers and upsampling process to build a segmentation of the original image from this representation.

Another approach for this is UNet (Figure 1.18), which is typically used for image segmentation where the output needs to have the same output as the input. In this case, the input is down sampled significantly to many small convolutional kernels. This representation is then up sampled using deconvolutional layers, with each up-sampling step also taking the output of the equivalently sized down-sampling layer as an input; this pattern of down then up-sampling with skip connections naturally generates a "U" shape in model diagrams, giving the model its name. Similarly to ResNet, we will implement UNet in later chapters and more discussion can be seen in Chapter 4 onwards.

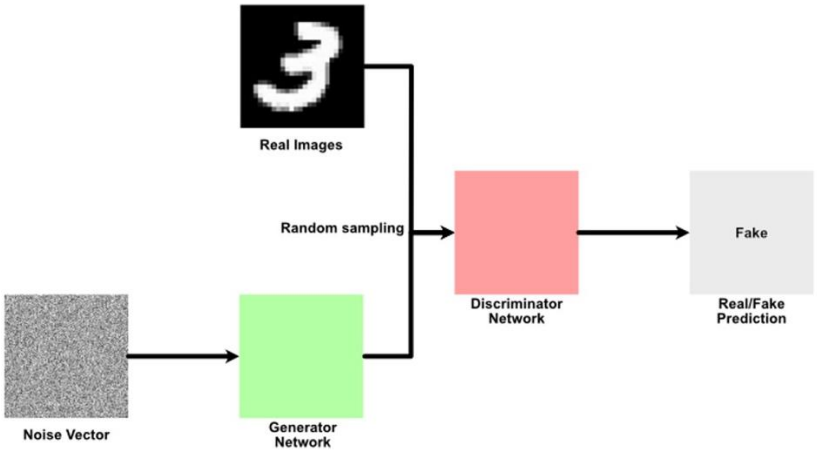


Figure 1.19: Diagram of how a GAN works.
A latent noise vector is passed into a generator network (green) that attempts to create and image similar to those in the training set, without seeing them. Then, either this output or a randomly sampled image from the training dataset is input into the discriminator network which must attempt to tell if the image came from the generator or the real data. The generator and discriminator have coupled losses such that the better the discriminator gets, the more the generator adjusts its weights.

Another problem deep learning can approach is the problem of data synthesis. Goodfellow noticed that by carefully constructing the loss functions between two neural networks, we can construct a zero sum game between them and train them together (Goodfellow et al., 2014). In GANs (Figure 1.19), there are two models; a generator and a discriminator. The generator takes in a latent vector of noise, with the goal of up sampling this noise through deconvolutional layers into a realistic sample of data from the dataset, without seeing the dataset it is trying to generate from. The discriminator model randomly takes in a sample

of data either from the generator or the genuine dataset and attempts to discern if the input is from the real data or the generator. The models weights are then both changed based on the discriminator's output; the more the generator "fools" the discriminator, the more the discriminator changes its weights. Likewise, as the discriminator gets better at recognising the fake data, the generator adjusts its weights more. This means that the models should train in parallel, with eventually the generator making data visually similar to the data from the dataset. GANs are the primary focus of section 3 in this thesis and will be discussed further there.

One problem that can commonly occur in GANs is when the generator model continuously generates the same sample data due to the discriminator labelling it as genuine. This is called "modal collapse". A similar problem is encountered in unbalanced data where a predictive model continuously predicts one class if that class is primarily present in the training set. This is typically approached by selection of training data in a way that balances the classes, or by using an intelligent loss function that avoids this problem, although this is not always possible.

1.4 Systematic Review of Existing Deep Learning Approaches to Electrophysiological Data

Before beginning development of novel approaches to single channel idealisation, I conducted a systematic review of past work on deep learning approaches to electrophysiological data which is presented below.

1.4.1 Method

Our initial search was carried out on Web of Science, looking for papers whose title or abstract contained all of the following:

- "deep learning", "artificial intelligence", "neural networks" or "machine learning"
- A word starting in "electrophyi-" or "physio-"
- "time series" or "signal"
- No instance of "CT", "MRI", "fMRI", "CAT", words starting with "video", "image" or "audio".

This query was constructed to be as comprehensive as possible while excluding any papers focusing on image, video or audio analysis from sources such as x-rays, or computerised tomography (CT) scans. We also excluded audio papers as otherwise this produced a large amount of acoustic papers. Even with these filters in place, 1305 papers were given, and further filtering was required (Figure 1.20).

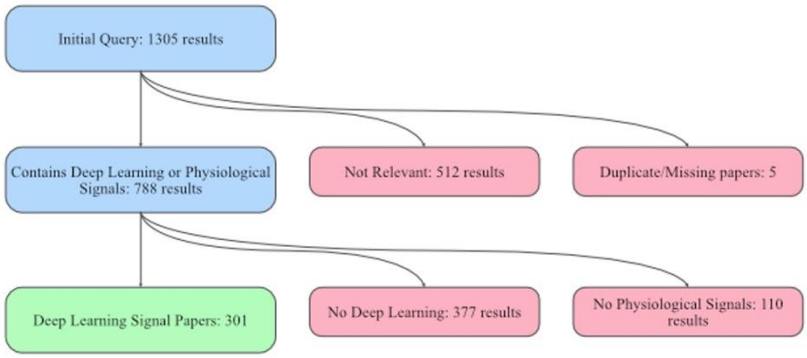


Figure 1.20: Diagram showing the filtering protocol for papers in the systematic review
The initial query returned 1305 papers; which was first reduced into papers that contained deep learning or physiological signals by inspecting the abstract. Some of these papers only contained either deep learning or physiological signals, so then the remaining papers were manually sorted by reading the entire paper and seeing if the deep learning methods were applied to the physiological signals directly.

We then manually discarded any papers without an abstract or title that had a specific reference to deep learning or a deep learning model, and papers without reference to a specific physiological signal. Most of these papers were non-deep learning methods applied to signals, and some of these papers did not analyse signals at all. This left us with 294 papers, which were then manually read and tagged with the year, type of signals analysed, and types of deep learning models used. For signals, papers where multiple signals were used in analysis were tagged, however any comparisons to other signals were omitted. Some signals were described under different names, or under an umbrella term, such as Galvanic Skin Response (GSR) and Electrodermal Activity (EDA). In these cases, the terms were logged under their umbrella term. Measuring the types of deep learning model

needed a similar compromise. All convolutional neural networks (CNN) and recurrent neural network (RNN) models were tagged as CNN or RNN respectively, along with any other structures they had. Papers that did not explain the details of the neural network or described it simply as an Artificial Neural Network (ANN) or Multi-Layer Perceptron (MLP) were logged as ANNs. Miscellaneous models were also logged as ANNs if they could not be described otherwise. RNNs were also sorted into miscellaneous RNNs, LSTMs and GRU models.

These then imported into an Excel file, which was then imported into a Python3 Jupyter notebook for analysis using the pandas and matplotlib libraries; this Excel file can be found online at https://github.com/stmball/PhysiologicalSignalsDeepLearning. We counted the number of papers by year, by signal and by model, and also the relationship between year and signal, and year and type of model. We looked at the breakdown of types of RNNs used, and finally the types of signals used in different deep learning models.

1.4.2 Results

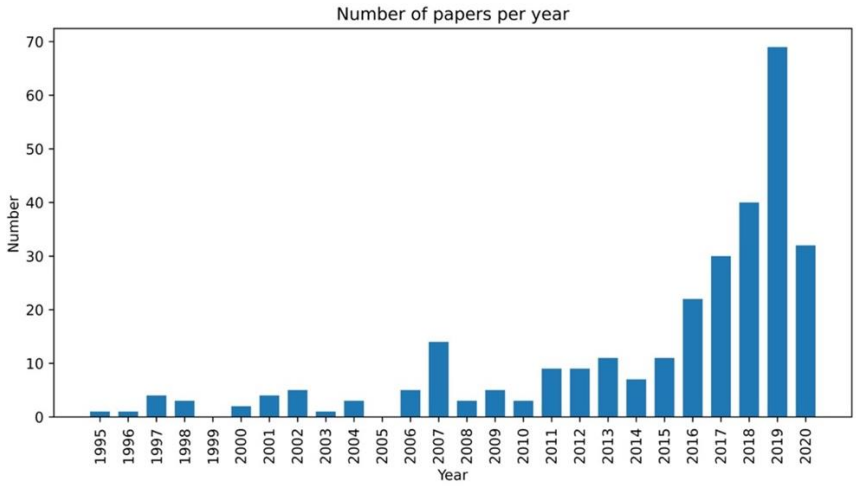


Figure 1.21: The number of papers on electrophysical deep learning analysis by year.
After conducting our search, we plot the number of papers containing signal analysis using deep learning by year. Here we qualitatively see an increasing trend in the use of deep learning methods for analysing physiological signals.

Grouping the papers by year published showed that deep learning is gaining significant popularity recently, with 2019 showing the largest amount of deep learning physiological papers to date (Figure 1.21); this is perhaps unsurprising as the underlying deep learning technologies continue to improve.

Next, we grouped the papers by the tagged signals; some papers studied more than one signal, in this case both signals were included in the counting. By discarding signals that were only counted once, we can see that the most popular signals are EEG, ECG, photoplethysmogram (PPG), EDA and electromyography (Figure 1.22).

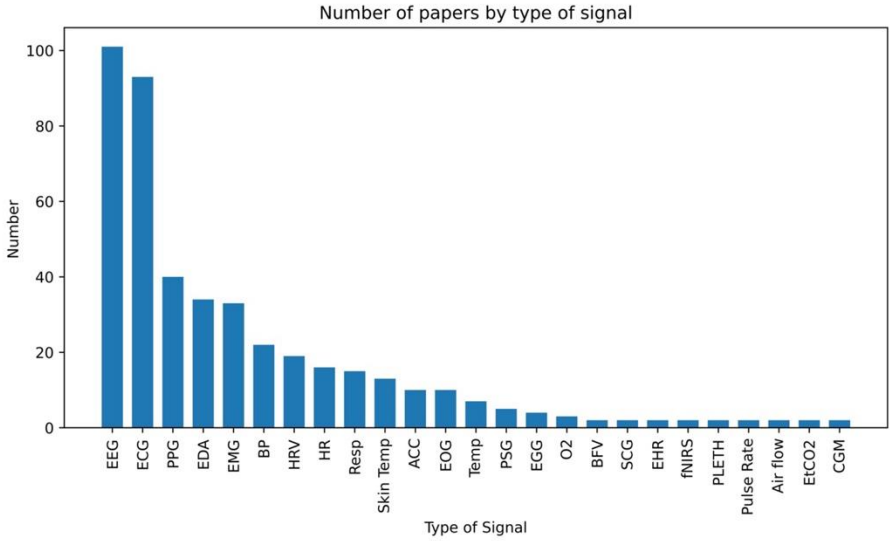


Figure 1.22: The number of papers on electrophysical deep learning analysis by year.
We also counted what signals were used in each paper. Here we qualitatively see an increasing trend in the use of deep learning methods for analysing physiological signals.

We then counted the papers by the types of models they used. We found that the most popular model type were ANNs (MLPs, miscellaneous ANNs and where details were omitted), then CNNs and RNNs (Figure 1.23).

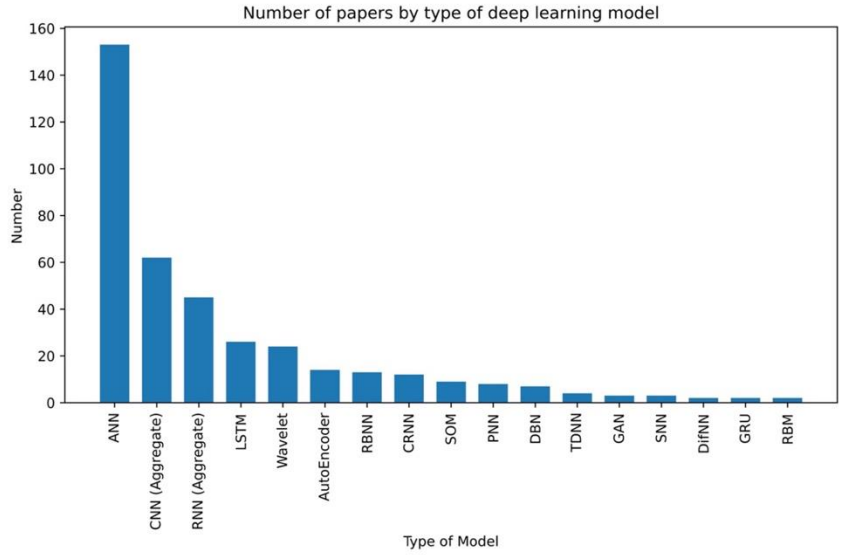


Figure 1.23: The number of papers by type of neural network used.
Similar models were aggregated into one category, and papers could be given more than one category (for example, a CNN based GAN). We see that the majority of papers are generic artificial neural networks (ANNs), with CNNs second, and RNNs third – this is surprising as conventional wisdom would dictate that recurrent neural models are more apt for time series data. Unsurprisingly, long short term memory (LSTM) models are far more popular than gated recurrent unit (GRU) models, which follows trends in more theoretical deep learning papers.

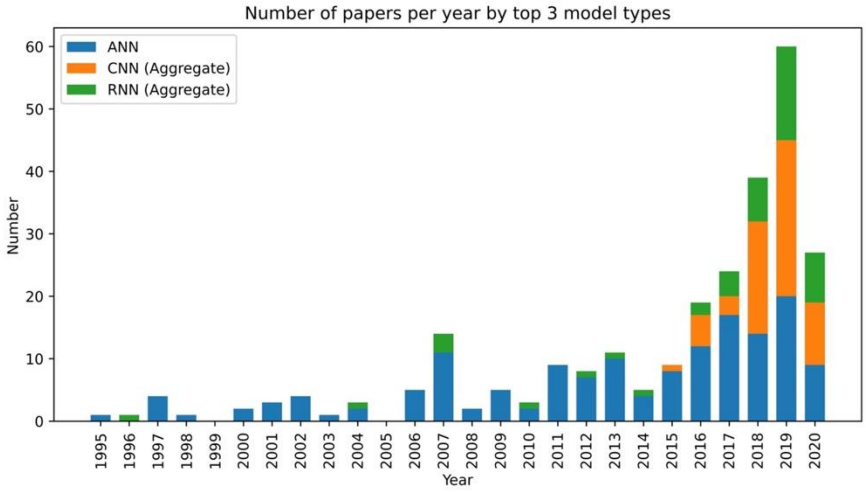


Figure 1.24: Number of papers by year for the top 3 model types (ANN, CNN, RNN).
Here we see that convolutional neural networks are a recent addition to the physiological signal analysis space, but now make up a significant proportion of papers. This again follows the trends in the wider neural network community, where convolutional neural networks gained popularity in the 2010s for their ability to recognise complex characteristics of data.

We then grouped the papers by models and plotted them against the year they were published (Figure 1.24). From here we can see that CNN and RNN papers have grown in popularity over recent years. We also plotted the same data but for the top five most popular signals (EEG, ECG, PPG, EDA and EMG) (Figure 1.25). We found that the usage of EDA and PPG data in deep learning physiological analysis is fairly new, with proportionally far more papers for EEG, ECG and EMG data from 1996 to 2010.

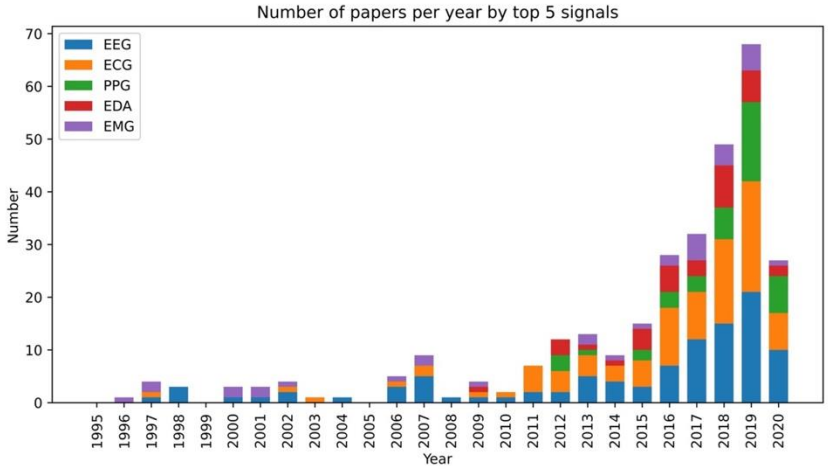


Figure 1.25: Number of papers by year for the top 5 signals (EEG, ECG, PPG, EDA and EMG).
We also split the count by year by what signals were being analysed by deep neural networks. We see little change in the proportionality of the types of models examined by the deep learning models – together EEG and ECG papers make up around 60% of papers in any given year.

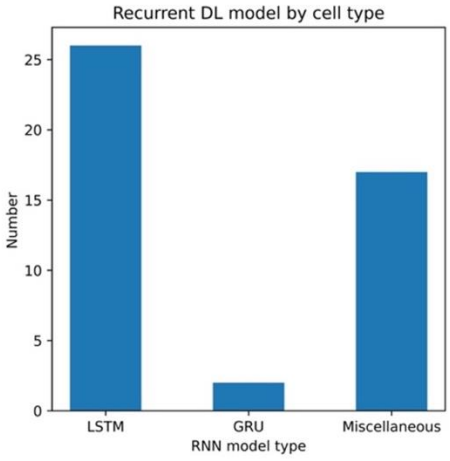


Figure 1.26: Breakdown of RNN models by type of cells used

We find that LSTM cells are by far the most popular type of recurrent cell used in signal analysis – here we have grouped all other types of RNN together, as implementations ranged from the simple recurrent cell as in Figure 1.10 to more bespoke implementations.

Looking at the types of recurrent neural networks used, the most popular type of cell used is the LSTM cell, with other miscellaneous RNN cells coming second followed by GRU cells (Figure 1.26).

We looked at if there was any visible correlation between the type of model and type of signal analysed (Figure 1.27). There was no significant correlation between the type of signal and type of neural network used ($p > 0.05$, $n = 301$, 2-Way ANOVA).

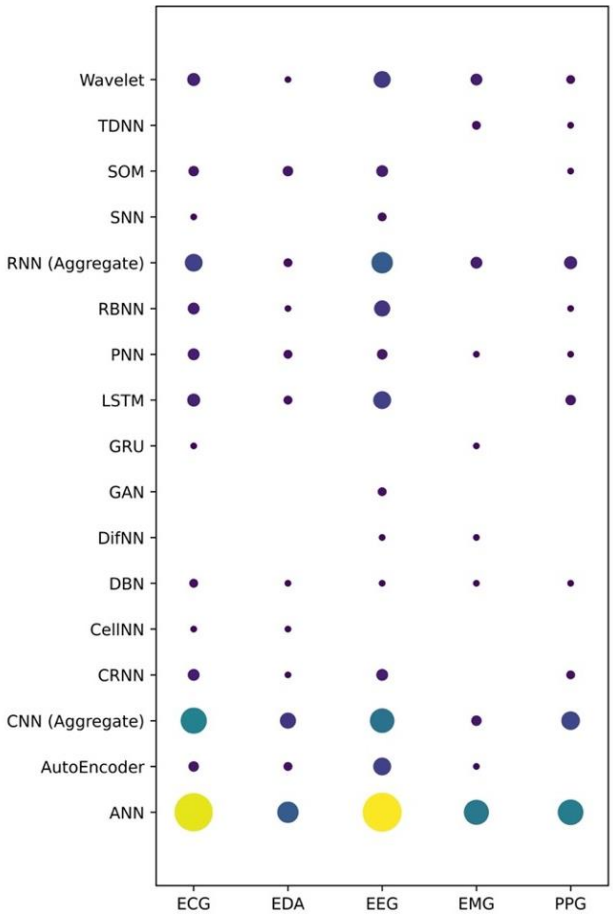


Figure 1.27: Papers represented by the type of signal they analyse, and the type of model used
To see if there was any relation between the type of model used and the data analysed, we counted how many papers used each combination of a model type and a signal. A larger and more yellow circle means more papers were analysed – we see no correlation between the type of model used for the signal ($p > 0.05$, $n = 301$, two way ANOVA).

The papers were sorted into the country of where the institution was based; in cases where no single institution could be found, the primary author was used instead. The results showed that authors from the People's Republic of China have published the most papers, followed by the United States of America (Figure 1.28).

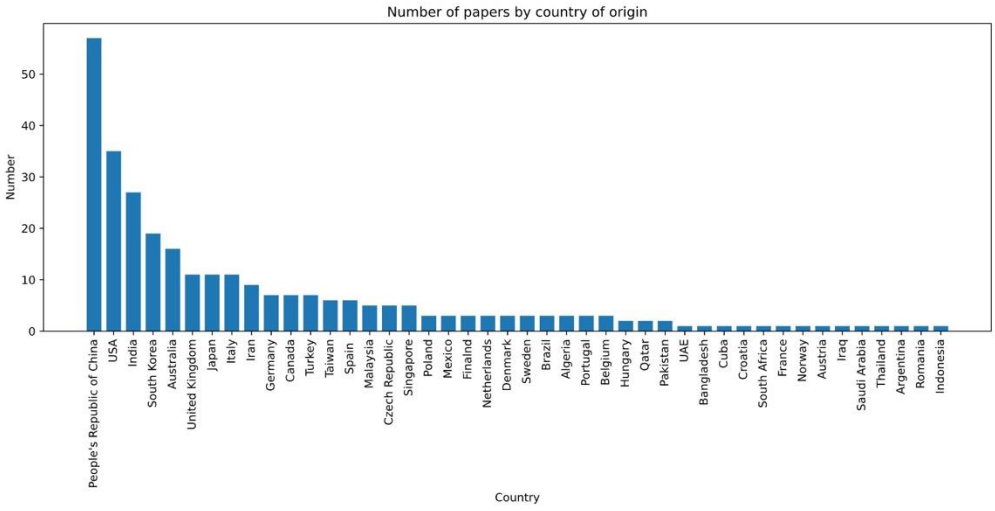


Figure 1.28: Papers by country of origin of publishing institution or first author
We plot the number of papers from each country of origin. Here the first author's base institution was used as the origin country – for papers from industry, the country where the company is based was used as the country of origin. We see that the People's Republic of China has the most papers, followed by the United States of America

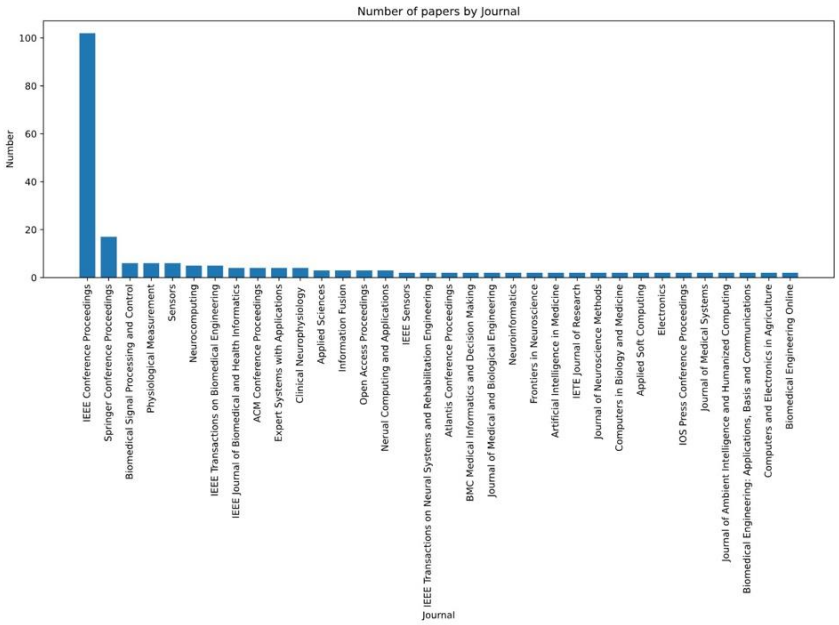


Figure 1.29: The number of papers published in different journal providers.
Published conference proceedings were aggregated by the institution responsible for publication. We see that IEEE conference proceedings makes up the vast majority of papers – this may be the result of different publishing traditions between fields; for example conference papers are much more common in engineering whereas journal papers are more standard for physiologists.

Finally, we sorted the papers by the journal, or institution they were published in; for example, conference proceedings published by Institute of Electrical and Electronics Engineers (IEEE) were logged under "IEEE conference proceedings". Since the number of unique journals published to was large, singleton entries have been omitted (Figure 1.29). We see that conference proceedings are by far the most popular entry, primarily from IEEE. This indicates that it is mainly engineering fields investigating the use of deep learning to analyse physiological signals, rather than more traditional biological ones.

1.4.3 Discussion

It is perhaps unsurprising that deep learning has gained such popularity in recent years when it is also gaining popularity in other areas of research and industry. We have seen a year on year increase in the number of papers published for deep learning from 2014-2019, however this trend may stop in 2020 if the rate of publication continues at the rate it is. Similarly, we found that the popular signals for deep learning physiological analysis were similar to those found in other reviews of the field (Faust, Hagiwara, Hong, Lih, &

Acharya, 2018; Rim, Sung, Min, & Hong, 2020), however electrooculography (EOG) signals were relatively underrepresented in our work.

The upward trends of EDA and PPG signals over time can be perhaps explained by the equipment used for recording these signals; both EDA and PPG sensors are relatively non-invasive whereas EEG and ECG recording is usually much more involved, with many sensors needed to be placed on different parts of the body as opposed to just one or two that can be coupled with a portable recording device.

The growth of CNN and RNN models tracks the development of CNN and RNN technologies in general; in recent years CNNs have gained significant attention in the field of computer vision but only recently have they been applied to time series signal analysis. It was only from 2012 when CNNs started to win the ImageNet Large Scale Visual Recognition Competitions, only 3 years before the first CNN paper for physiological signal analysis was published according to our findings.

LSTMs continue to be the most popular choice of RNN cell, potentially because they have the most trainable parameters compared to GRU or regular RNN cells. It remains to see if LSTMs and RNNs in general have a place in physiological signal analysis, as part of hybrid CNN RNN architectures, or if CNNs or other models will take over.

Finally, there is no obvious relationship between the types of signal analysed and the types of model used. It is unknown as to if there signals have any inherent characteristics that would make one type of model preferable to another; in many cases it's the purpose the model is being used for that drives the model choice; for example GANs are typically used for synthetic data generation rather than classification tasks, and AutoEncoders used for dimensionality reduction, although this is not always the case.

1.4.4 Conclusion

We have reviewed the literature for physiological signal analysis using deep learning techniques. We found that the most popular signals were EMG, EDA, EMG, EEG and

PPG, with PPG and EDA signals increasing in popularity especially in recent years. CNNs and RNNs have grown in popularity over time, but there is no "one size fits all" type of model that is being used in most papers. This is perhaps unsurprising since CNNs have only gained significant attention from around 2012. Finally, there seems to be no correlation between the types of signals being analysed and the types of models being used.

1.5 Hypothesis and Objectives

Given the above; as new deep learning tools and computing resources have become readily available it is important to investigate how far deep learning models can advance ion channel research. It is therefore hypothesised that new, convolutional deep learning-based architectures can both idealise patch-clamp data more accurately, and recover more information (such as the Markovian state) than existing models or conventional solutions. It is also thought that deep learning can go further than a simple classification problem and perhaps be used to build more accurate simulation methods to the simple Markovian models with noise that exist currently.

Therefore, the aims of this work are as follows:

a) Develop new ways of ion channel record generation either by improving traditional methods and by using generative adversarial networks. [Chapter 3]
b) Perform the first direct recovery of ion channel Markovian state with deep learning models [Chapter 5, 6, 7]]
c) Adapt the Markovian state model architecture to analyse multichannel data and improve performance over the existing DeepChannel model. [Chapter 8]
d) Compare performance of current ion channel analysis techniques (QuB) with deep learning models [Chapter 8]
e) Quantify performance of our best deep learning models on 3 different real biological datasets, compare with existing methods. [Chapter 8]

2 Methods

The principle aim of this work was to build new deep learning models for classification and generation of ion channel records. To be able to achieve these aims, we need two approaches; biological work within the lab to obtain real, lab recorded data for model training or evaluation; and computational work for developing these models.

2.1 Computational methods

2.1.1 Data Simulation Pipeline and Datasets

For our use case, we require enormous, labelled data sets for model training and evaluation. Hand labelling recorded data would not be appropriate as it would involve a subjective idealisation process and be prohibitively time consuming. For example the benchmark dataset used for evaluating image recognition models, ImageNet (Deng et al., 2009) is a labelled dataset of (currently) 14 million images. Hand labelling these images (accounting for 5 seconds per image) would take 810 days, or over two years. In addition, the underlying Markovian processes would need additional analysis and may not be accurate depending on the quality of the idealisation.

In similar work, a simulation approach has been used to generate ion channel records for analysis, but it has typically been done so with a static Markov model and simple Gaussian noise added. This is not representative of real ion channel data; it is more common for $1/f^{\alpha}$ noise (with $\alpha \geq 1$) to be present in ion channel records (Siwy & Fuliński, 2002), as well as some lower frequency noise for baseline drift over a longer term (Figures 2.1, 2.2).

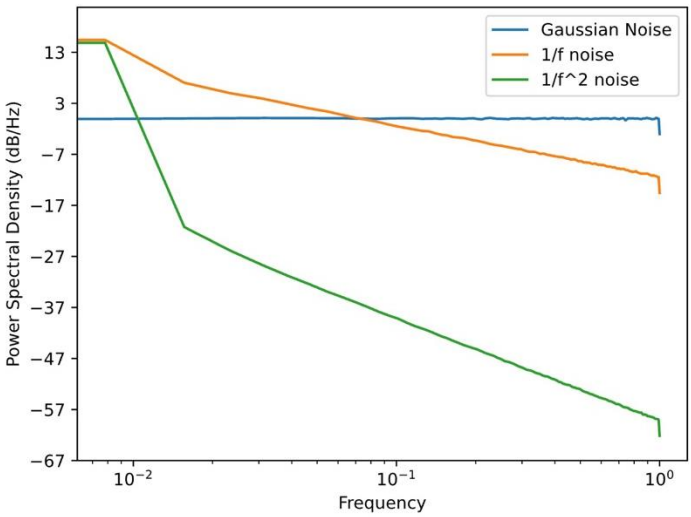


Figure 2.1: Power spectral density graphs for simulations of normal Gaussian noise, 1/f noise and 1/f^2 noise. *In related work, Gaussian noise is often added to a square wave signal to simulate the noise seen when recording ion channel activity in the lab; however this is not what is normally observed. Typically the type of noise seen in an ion channel recording is a type of noise called 1/f^n noise, where n determines the gradient of the slope in the power spectral density graph of the noise. 1/f noise (also known as "pink" noise) is typically seen in nature; and as n increases, the noise frequency profile skews towards the lower frequencies. In this work we extensively use 1/f^n noise with randomised n to create realistic noise to add to a simulated signal.*

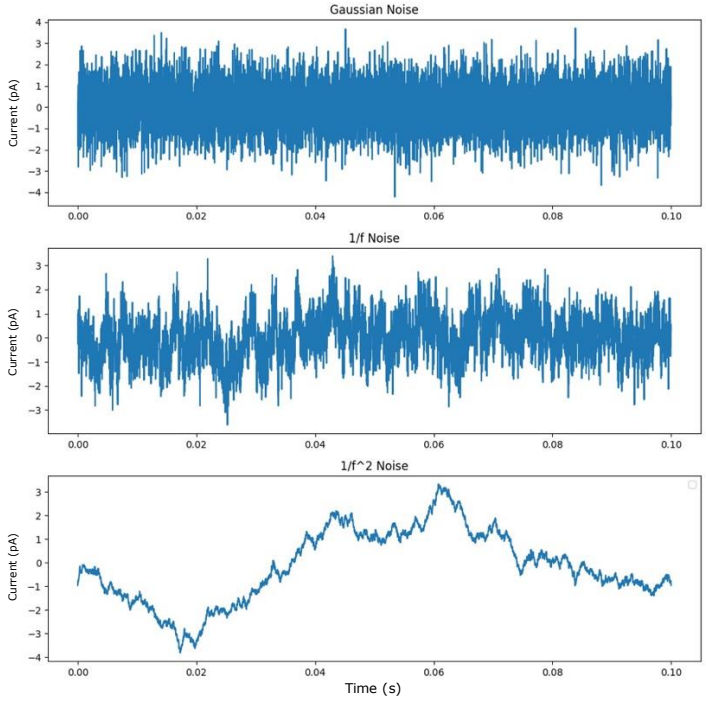


Figure 2.2: Examples of simulations of different types of 1/f^n noise with the n parameter changed
Gaussian noise, or 1/f^n noise, can be seen in the top axis, and is typically added to square wave signals in related work to simulate ion channel noise; however the 1/f noise in the central axis is seen much more commonly in ion channel noise analysis. As n increases (for example, n = 2 in the bottom axis), the frequency profile skews more towards the lower frequencies, a slower variation in the signal over a longer period of time. By combining these 1/f noise terms, we can create more realistic synthetic ion channel data than seen before through mathematical simulations.

Therefore, we developed a new Python package, `DeepMICA` (https://github.com/stmball/DeepMICA), that allows for the simulation of Markovian ion channel records, allowing for perturbation of the Markovian matrix, and a more nuanced noise system that gives users the ability to layer different types of noise on top of each other to give a more authentic output record (Figure 2.3).

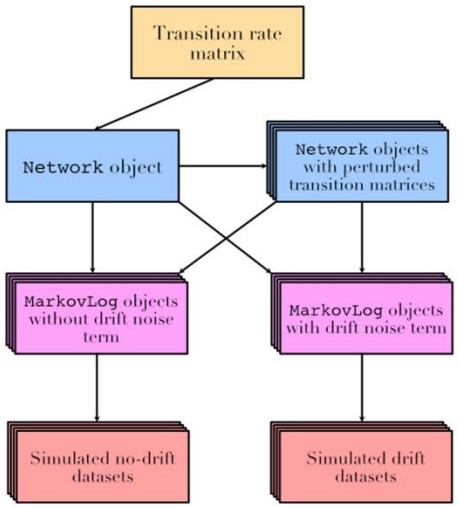


Figure 2.3: DeepMICA Workflow for generating new datasets.
We start with a transition rate matrix (either manually derived or taken from other work), and use it to create a Network object. For static datasets, this Network object is unchanged, but for the datasets that test for changes in transition rate matrix, we create a number of new Network objects with perturbed transition rate matricies known as the "perturbed" datasets. From this, MarkovLog objects are created which are responsible for the simulation of the datasets, with some with drift and some without for both "Static" and "Perturbed" Networks. These are then used to generate the simulated signals used for deep learning training.

The general structure of the package is as follows: first, a `Network` object is initialised which manages the Markovian model, allowing for the generation of canonical forms (*explained in Chapter 3*), and perturbation of the transition rate matrix using a symmetric triangular distribution for scaling with a user defined magnitude to simulate drug action affecting changes in channel kinetics. This allows measurement of the effects of small changes in the Markovian model on the performance on a deep learning model; if the deep learning model is overfitting onto the initial Markovian model, then changing the Markovian matrix in this way will drastically harm performance. Conversely, a

successfully generalising model will be robust to these small changes in Markovian transition rate matrix.

The `MarkovLog` object is responsible for taking these Markovian models and generating a recording by sampling the dwell times from the Markov model to get a "discrete record" of states and dwell times, and interpolating this record and adding a series of noise terms to simulate the characteristically noisy nature of ion channel data for a "continuous record" as expected from patch clamp recordings. The discrete record can also be used in a Viterbi analysis of the data, which is explained in more detail in Chapter 5. An example output of the kind of data obtained from this methodology can be seen in Figure 2.4.

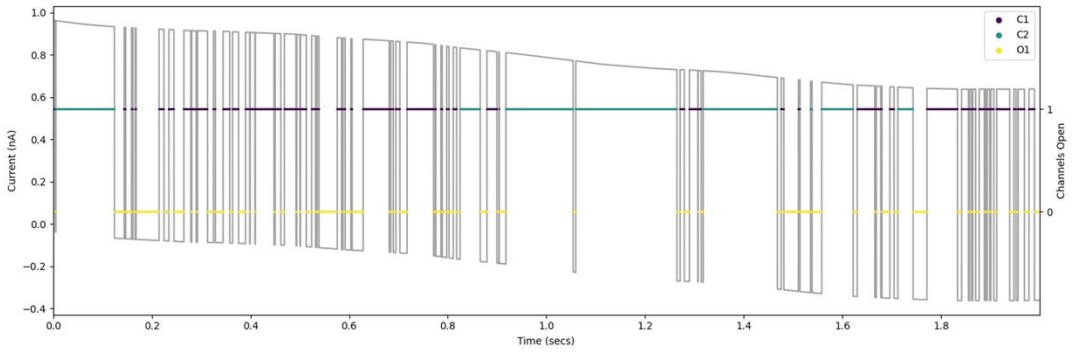


Figure 2.4: Example of 2 seconds of "simple, slow" 1/f noise layered on top of a simulated Markovian signal with no additional noise layers
Here we use a simple 3 state hidden, continuous time Markovian model for the square wave simulation, and apply 1/f^n noise with a very high n (randomly sampled from a uniform distribution between 3.5 and 4.5). This results in a very low frequency, but still random noise term that reflects the baseline drift visible in ion channel recordings from external conditions. Most other work (QuB, MDL, JSMURF) doesn't implement baseline drift at all, and the previous DeepChannel work implements baseline drift as a sinusoidal change over time.

Although multiple noise terms are possible with our `DeepMICA` package; the two used in all the data generation are the Simple and Scaled $1/f^{\alpha}$ noise mechanisms. These were chosen for the following reasons:

a) Simple $1/f^{\alpha}$ noise with a controllable amplitude and α parameter. This simply generates $1/f^{\alpha}$ noise and adds it onto the input signal with no consideration of the number of channels open. In our case, we use this with a high value of α

(typically between 3.5 and 4.5) to simulate baseline drift. An example of only this baseline drift applied to the simulated Markovian signal can be seen in Figure 2.4.

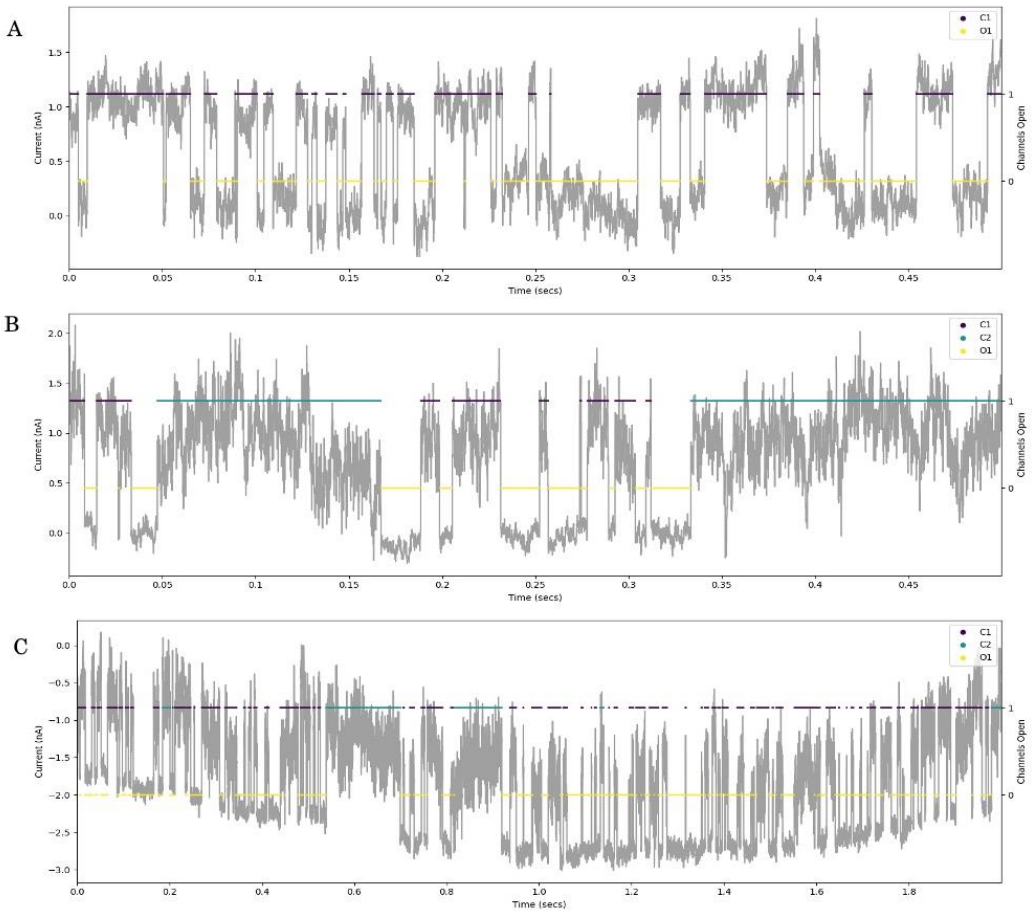


Figure 2.5: Examples of different approaches of simulating the high frequency noise in ion channel recordings. *Plot A is the "simple" 1/f noise seen above with the α parameter set to 1, and signal to noise ratio of 4. Plot B shows the "scaled" 1/f noise with α set to 1, base signal to noise ratio of 10, but the scaling parameter set to 4. This means that for every channel open, the scale of the noise increases by a factor of 4 – this is a more realistic approach as it is seen in practice that noise increases as channels open. Plot C shows both the scaled noise with a low α parameter aggregated with the simple 1/f noise with a high α. Parameters for data used in training were tuned empirically to match the ranges of what was seen in existing recorded data. In each case, the legend shows the underlying Markovian state the simulation is in for each timepoint.*

b) "Scaled" $1/f^{\alpha}$ noise with controllable amplitude, scaling and α parameter. This generates $1/f^{\alpha}$ noise but scales the amplitude with the number of channels open according to the scaling parameter. It is clear from analyses of ion channel records that the signal is significantly noisier when a channel is open, so called "open channel noise" (Heinemann & Sigworth, 1988; Sigworth, 1985). By using this scale parameter, we can adjust the amplitude of this noise on a per-channel

opening/closing basis to reflect this behaviour. We typically use this noise term for the background noise that is usually, incorrectly represented by Gaussian noise in other work; by setting α randomly between 0.8 to 1.4 we cover a range of conditions to test for model robustness versus different noise levels. An example of this noise applied to a channel, and its difference to the "simple" noise can be seen in Figure 2.5, along with an example of scaled noise along with the simple, high α noise above.

The inclusion of 1/f noise is an improvement on Gaussian simulations seen in previous work; however may be as much as a result of common filtering practice as the nature of the data itself. Therefore, in addition to this data, a digital 8-pole Bessel was added at the end of the generation pipeline to reflect this process.

We also developed, and included in the `DeepMICA` library; a novel toolbox for preparing this data for deep learning, as well as a suite of tools for evaluating model performance on both the Markovian prediction (state unreduced) and open/closed configuration (state reduced). To achieve this, we rely on a number of external packages, namely `numpy` (Harris et al., 2020), `pandas` (McKinney, 2010), `matplotlib` (Hunter, 2007), `scikit-learn` (Buitinck et al., 2013), `colorednoise` (https://github.com/felixpatzelt/colorednoise, (Timmer & Koenig, 1995) and `tqdm` (https://github.com/tqdm/tqdm)

For training and evaluating deep learning models data was sourced from three methods: We produced synthetic datasets using the DeepMICA library above; recorded our own data from the lab; and used previously recorded data from other work.

We synthesised different synthetic datasets for different tasks, each testing at different levels of "difficulties" for the model; for example, asking a deep learning model to recover the Markovian state of a very simple Markov model; or simply to idealise a single channel with low levels of noise and no baseline drift are both relatively easy problems for a model to solve. However, resolving 10 channels or a complex Markov model with changing rate

parameters and high noise is significantly harder. In total, we have simulated 16 different *types* of datasets, using the our new methods; 8 for Markovian state recovery (Chapters 5, 6, 7) and 8 for multichannel idealisation (Chapter 8). For Markovian state recovery we have every combination of the simple 3-state model and the 5-state model taken from literature (see below), modulating the transition rate matrices and not; and baseline drift being present and not. For the multichannel idealisation datasets were generated with and without drift for one, three, five and ten channel records.

For the Markovian dataset, the two Markov models used can be seen in Figure 2.6. The first (known henceforth as the "3 state model"), is a very simple Markov model consisting of two open states and one closed state, with one open state around 10 times more likely to be reached from the closed state, but on average has around 10 times shorter a dwell time as the "longer" open state. The second Markov model is taken directly from the literature (Davies, Purves, Barrett-Jolley, & Dart, 2010) and reflects a far more complex and realistic Markov model in practice. Figure 2.6 shows what this Markov model looks like diagrammatically.

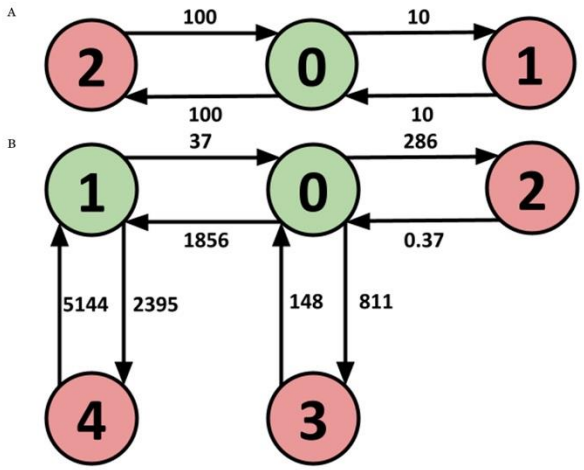


Figure 2.6: Diagrammatic representation of Markov models used throughout this work.
Green states represent open channels and red closed. Plot A shows the "simple" Markov model where we have one open channel and two closed channels; one being a "long" close and the other a "short close". Plot B shows a model taken from literature (Davies et al., 2010) used to test model performance on a real Markov model.

2.1.2 Deep Learning Training

For the deep learning portion of this work, two different setups were used throughout at different stages of development. In cases where monitoring output was important (such as checking the output of a GAN), Jupyter notebooks (https://jupyter.org/) were used with a manual TensorFlow (Abadi et al., 2015) install. As the TensorFlow package needs specific versions of GPU drivers and supplementary software installed, for easy of use on other computers the official TensorFlow Docker image was used to run training in a container to avoid version conflicts.

Training was run on a number of different machines, but all were equipped with NVIDIA TITAN RTX GPUs and Intel Xenon processors to ensure all training times were consistent.

Unless explicitly stated otherwise, all models were trained using an 80/20 train-test-split validation technique, along with a learning rate scheduler and early stopping mechanism to allow for each model to reach peak performance. If the validation data 'sets' loss did not improve within 5 epochs, the learning rate was reduced by a factor of ten, and if the loss did not improve within 20 epochs, training was stopped and evaluation began. We use the categorical cross entropy loss function for training (also known as log-likelihood).

For classification models, model evaluation was performed using a matrix of different metrics; both the micro F1 score and Cohen's Kappa score (explained below) were taken for both the Markov-state prediction (where relevant), and the simple open/closed configuration of the channel ("reduced-state"); or the level of the square wave classification. This gave multiple avenues for investigation to improve model performance, for example if the Markovian state classification was weak but the channel classification was strong, the model was doing well at idealising the recording but could not understand the underlying mechanism.

The F1 Score is a commonly applied metric for measuring the efficacy of classification models; it is calculated with the following formula:

$$F1\ score = 2\frac{Precision * Recall}{Precision + Recall}$$

Where:

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Where TP, FP, FN are the number of true positives, false positives and false negatives respectively.

In multiclass classification problems, there are two options for calculating the overall F1 score; the *micro* average calculates the F1 score across all classes separately, and the *macro* average calculates the F1 score for each label, then averages them together. We use the *micro* averaging to achieve a measure of point by point accuracy.

Another way of approaching the modal collapse problem discussed earlier is to use the Cohen's Kappa score. Formally it is defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Where p_o is the observed agreement between the model and the ground truth; i.e. the number of correct classifications divided by the total number of data points; and p_e is the expected probability of chance agreement, calculated as follows:

$$p_e = \sum_{i \in S} \frac{\#\ predictions\ of\ class\ i\ from\ model}{total\ number\ of\ points} \cdot \frac{\#\ ground\ truth\ instances\ of\ class\ i}{total\ number\ of\ points}$$

Where S is the set of all classes.

Cohen's Kappa score is much stricter score for measuring the accuracy of machine learning models, as it penalises modal collapse in multiclass classification severely. For example, in the case where a model is predicting all 0 classes in a record with nine 0

classes and one 1 class, the macro averaged F1 score gives a score of 0.47, the micro averaged F1 score gives a score of 0.9, but the Cohen's Kappa score gives a score of 0. Furthermore, if the model starts to get the prediction drastically wrong; worse than a random classifier; this score can reach negative values. A negative Cohen's Kappa indicates prediction worse than that expected by random chance.

Both of these metrics are helpful tools in measuring the efficacy of our models accounting for the need to predict underrepresented classes. The major disadvantage of these two metrics however is that as they are not differentiable; due to the nature of back-propagation needing derivatives of the loss function with respect to each parameter, such metrics are helpful for model evaluation but cannot be used as loss functions for the deep learning training process.

3 Synthetic Dataset Generation with DeepGANnel

3.1 Introduction

Mathematically modelling ion channel data and adding generated noise allows for a large amount of generated data to be made available for deep learning training, but this data is likely to overlook some unknown characteristic features of the data. A data driven approach of simulating data from existing data allows for a deep learning model to learn the characteristic features of the data and potentially allows for a more accurate simulation of ion channel data, with no prior knowledge and control of the underlying Markovian model. This chapter has been adapted from the work published in the PLOSONE journal (Ball et al., 2022).

In this chapter, we successfully leverage generative adversarial networks (GANs) to build an end-to-end pipeline for generating an unlimited amount of labelled training data from a small, annotated ion channel "seed" record. Our method utilises 2D CNNs to maintain the synchronised temporal relationship between the raw and idealised record. We demonstrate the applicability of the method with 5 different data sources and show authenticity with t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation Projection (UMAP) projection comparisons between real and synthetic data. The model would be easily extendable to other time series data requiring parallel labelling, such as labelled ECG signals or raw nanopore sequencing data.

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have proven to be important for generative tasks, such as image generation (Karras, Laine, & Aila, 2019b) or audio synthesis (Donahue, McAuley, & Puckette, 2019). GANs have been used in electrophysiology to create new unsynchronised datasets (Qin et al., 2020; Truong et al., 2019), with the former example being used to generate data for model training, but

continuous labelling of data (millisecond-by-millisecond) has remained an unsolved challenge. GANs have great potential to allow a data driven approach to synthesis, using a small amount of "real", lab recorded data to generate an unlimited amount of simulated data; the advantage of this method is that noise and artefacts present in the real training dataset also appear in the generated data.

T-distributed stochastic neighbour embedding (t-SNE) (Van der Maaten & Hinton, 2008a) and uniform manifold projection (UMAP) (McInnes, Healy, & Melville, 2018) dimensionality reduction algorithms are non-linear graph-based methods for reducing the number of dimensions of a dataset; either for computational reasons (e.g. computationally expensive algorithms might need dimensionality reduction to become feasible), or visualisation purposes (reducing a complex dataset down to two dimensions to graph and understand the structure of the data is sometimes helpful). These methods differ from the commonly used principle component analysis (PCA) dimensionality reduction method as they aim to preserve different kind of structures; PCA prioritises preserving global structure over long distances whereas t-SNE preserves local distances. UMAP aims to achieve both by implementing a further manifold transformation to an adjusted t-SNE to preserve some global structure as well.

In this work we develop a GAN model to generate realistic, simulated ion channel data from existing patch-clamp recordings from several channel phenotypes, each with a continuous parallel state label. These outputs could then be used to build more accurate ion channel analysis models. We use a further series of analysis tools to evaluate the model's performance; including T-SNE and UMAP projections to investigate if each channel's high dimensional characteristics are reflected in the simulated data.

3.2 Methods

3.2.1 Experimental Workflow Summary

Experimental ion channel data (See "Source Data", Table 3.1) was recorded using patch-clamp electrophysiology and then passed to our GAN model network (using a Jupyter

notebook) as input data. After training, the GAN network is set to write simulated data to file. These simulated data were then passed along with the original raw "seed" partner datasets to a Jupyter notebook that ran dimensionality reduction algorithms to compare the raw and GAN simulated datasets (see "T-SNE and UMAP").

3.2.2 Model design

We propose a GAN based model to generate synthetic time-series data that includes realistically similar features to real ion channel molecule currents. Figure 3.1 describes the complete pipeline of the proposed DeepGANnel model in this work. The architecture of the GAN model introduced in this work follows the regular DC-GAN methodology, but applies the convolutional neural networks that efficiently demonstrated to produce time series datasets in previous work (Delaney, Brophy, & Ward, 2019; Donahue et al., 2019). In the following sections, the utilized neural networks briefly are covered along with the processed data information and evaluation metrics criteria.

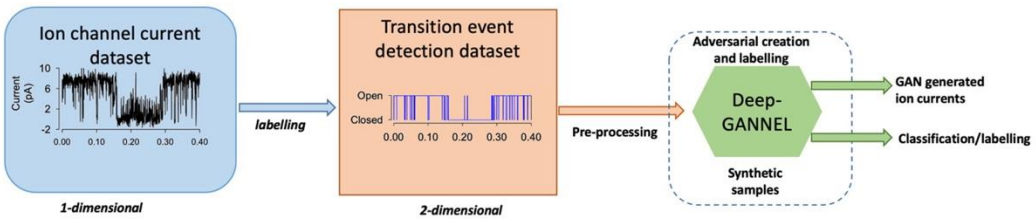


Figure 3.1: Pipelines for DeepGANnel Development.
A relatively small amount of lab recorded data is labelled manually, then preprocessed and used to train the DeepGANnel networks to generate labelled, synthetic ion channel data.

Data are recorded using standard electrophysiological techniques. These raw data are then labelled using existing software and expert supervision. Pre-processing makes copies

of these data adding a small amount of gaussian noise to the raw data and reshapes the two signals into 2D "images" (2x1280).

3.2.3 Generative Adversarial Networks (GAN)

A GAN typically consists of two neural networks competing against each other; the generator network tries to convert random noise into observations that seem as if they have been sampled from the original data; while the discriminator network aims to classify whether the sampled data comes from the original dataset or output of the generator network by predicting a class probability. The training process is employed in an adversarial manner between the two networks by updating the parameters of both models based on updates of the discriminator; the generator is looking to maximise the discriminator's uncertainty in classifying an output as real or fake, and the discriminator is looking to minimise this.

3.2.4 Design of the Networks

The DeepGANnel networks are based on the successful DC-GAN model (Radford, Metz, & Chintala, 2015), with the input and output shapes of the generator changed to images with dimensions n by m (one channel for the raw signal, one for the idealisation, n and m were typically 2 and 1280, but these numbers can easily be varied depending on context) along with changes to the subsequent hyperparameters along with pre/post processing to facilitate this change.

The architecture of the generator G consists of strided deconvolutional layers that allow the network to undergo spatial learning with its own up-sampling; batch normalisation layers allow for stabilising learning parameters by normalising inputs and Leaky Rectified Linear Unit (ReLU) activation functions for all layers, except for the tanh function that is used in the output layer. The input to this model is a matrix of latent noise that is transformed into the output signal, and the output of the model is a two-dimensional sample generated from the noise that can be sent to the input of the discriminator for training the model.

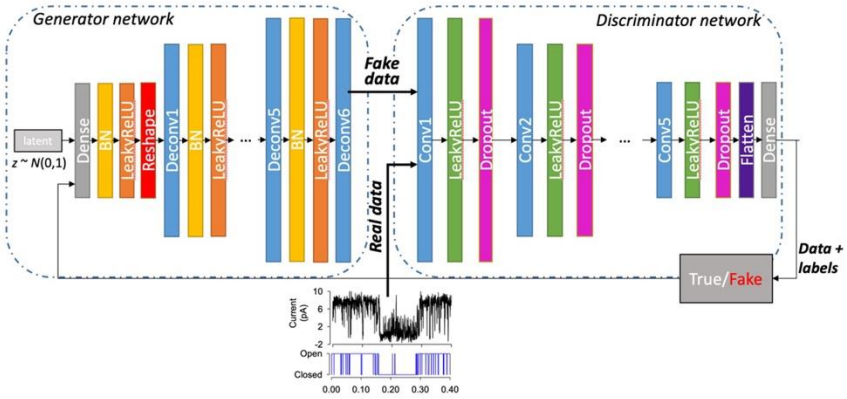


Figure 3.2: DeepGANnel network architecture
On the left is a generator network of DeepGANnel in which the input is a random noise and output is a 2D matrix by 2x1280 that is converted from 1-D time series. The generator model in this architecture uses three deconvolution layers (upsampling) to produce an matrix from seed (random noise shape by 2x1280). Several Dense layers are used to take the noise input and transform into a desired matrix by up-sampling (Deconv or transposed convolutional layer) steps size of 2x1280. At each layer, Leaky Rectified Linear Unit (LeakyReLU) is used as activation function with batch normalisation, except output layer which uses tanh activation function. The discriminator model comprises of several convolutional layers with the same activation function (LeakyReLU) at each layer to classify whether a matrix of values is real or fake by comparing the sampled data to real data. Dropout layers were also appended to all convolutional layers except input layer with the value of 0.3 to reduce overfitting. Finally, a Dense output layer is used after flattening the network structure with classified labels.

The discriminator D, is also a deep convolutional neural network. Similar to the generator, this model combines strided convolution layers to downsample the input data to obtain a binary classification of the input record (real or generated). In our discriminator architecture, the batch normalisation layers are not used, but instead a dropout regularisation technique was used at each layer. The last convolution layer in our D network is flattened and passed into a sigmoid function for classification, but otherwise similarly to the generator; Leaky ReLU activation functions are used for all other layers. Figure 3.2 shows the architecture of our DeepGANnel model in this work including both G and D networks. The Adam optimiser (Kingma & Ba, 2014) was used initially with a 0.0001 learning rate for the training process. Manual tuning of the learning rate parameters was needed during training to avoid overfitting from one of the models.

3.2.5 Data sources

For GAN simulation training, ion channel recordings were obtained from cell-attached patch clamp electrophysiology on a number of different cell types; for detailed methods please see the associated references.

Table 3.1: Sources of raw data for development and testing of the GAN network

Code	**Channel source**	**Sample size (segments)**	**Methods Reference**
Phenotype A	Canine articular chondrocyte	179	(Lewis et al., 2013)
Phenotype B	WinEDR simulated	244	*None, see text below*
Phenotype C	Tracheal chondrocyte	36	*None, see text below*
Phenotype D	Paraventricular nucleus of the Hypothalamus	499	Feetham, Nunn, Lewis, Dart, & Barrett-Jolley, 2015)
Phenotype E	Equine articular chondrocytes	286	(Mobasheri et al., 2010),

In the case of tracheal chondrocytes ("Phenotype C") data was provided by my co-author Dr Abdul Kadir (Ball et al., 2022). Data was acquired with methods similar to (Lewis et al. 2013), except that trachea were isolated, cut into small pieces, and thence treated as for articular chondrocytes. In all cases, animals were previously euthanised for unassociated reasons; no animals were killed or harmed for this study. Typically, patch

pipettes were fabricated using 1.5 mm o.d. borosilicate glass capillary tubes (Sutter Instrument, USA, supplied by INTRACEL, UK). They were pulled using a two-step electrode puller (Narishige, Tokyo, Japan) and when filled with recording solutions, had a resistance of approximately 5-10 MΩ. In each case, data was recorded using cell-attached patch clamp with an Axopatch 200/a amplifier (Axon Instruments, USA). Low-pass filtering was set to 1 kHz (Axon's built in 4-pole Butterworth filter) and data were digitized at 5 kHz with a Digidata 1200A interface or CED 1401 (CED, Cambridge, UK). Recordings were made with WinEDR (John Dempster, University of Strathclyde, UK).

In addition, in order to achieve a diverse population of ion channel records to test our GAN model, we simulated ion channel data using the simulation feature in WinEDR (John Dempster, University of Strathclyde, UK) using default rate constants ("Phenotype B"). Idealisation/annotation of raw data was performed with QuB (Nicolai & Sachs, 2013). Datasets had different record lengths between 1024 to 4096 datapoints per channel. Sample sizes given in Table 3.1.

For manifold projection data input. The raw data is the same as above (Table 3.1), and the simulated data are outputs from the GAN, these specific datasets are included in data in the associated repository.

3.2.6 Evaluation metrics

GANs are considered successful when they implicitly learn the distribution of samples of the real dataset. We assess the efficiency of the proposed DeepGANnel model to simulate single molecule data by comparing real to GAN simulated data using a number of different approaches. The standard metrics for GANs are the so-called generator loss and discriminator loss. These are calculated as the logistic binary cross entropy loss; this is calculated as:

$$L = -(y \log(p) + (1 - y) \log(1 - p))$$

Where y is the true label and p the predicted label. For the generator this is calculated once, with the true label being whether the output is fake or not, and the predicted label being whether the discriminator predicted if the output is fake. For the discriminator, this is calculated twice then totalled; once for the fake output as above, and again in the same manner for real outputs.

We also use the t-SNE and UMAP dimensionality reduction algorithms as a means to compare and contrast the different ion channels and sources in a more meaningful way. Additional measures of GAN performance (Maxiumum Mean Discrepancy and Dynamic Time Warping) were used by Dr Numan Celik in the original paper (Ball et al., 2022).

3.2.6.1 T-SNE and UMAP

To compare the raw data (Table 3.1) and the matching GAN outputs; raw data (Table 3,1) and matching GAN outputs were used as inputs for manifold projection (t-SNE and UMAP) analysis. Dimensionality reduction algorithms allow us to visualise high dimensional data in a low dimensional format, typically within a visualisation such as a scatter plot. Principal Component Analysis (PCA), for example, reduces the dimension of data by choosing a new orthogonal basis for the data based on the maximal variance of the data. For non-linear datasets, this dimensionality reduction may not accurately convey the shape or pattern of the higher dimensional data, as points that are close together in Euclidian space but far away in the context of the data may be brought together during the PCA transformation (for example, a circular or ring-shaped dataset loses its shape in PCA). The so-called "kernel trick" can solve this problem by providing a non-linear transformation to the space before the PCA algorithm is applied, however this requires that this non-linear transformation to be found algorithmically, which can be time consuming and inaccurate.

The t-SNE (Van Der Maaten & Hinton, 2008b) algorithm focuses on local similarity rather than global similarity by considering only a neighbourhood around each point in high dimensional space. The algorithm first constructs a matrix of probability distributions for

each pair of points in the higher dimensional space such that close points have a high probability and far away points have a low probability. The algorithm then constructs a second matrix of probability distributions for the lower dimensional space and minimises the Kullback-Leibler divergence between the two via gradient descent. This method preserves local similarities, but global similarities are lost between far away points.

Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) was developed to attempt to maintain the global similarities by using the same general method as t-SNE (the construction of two matrices and optimisation to fit the lower dimensional one to the higher dimensional one), but uses a topological transformation of the data and a different divergence metric to achieve stronger global similarity. Note that in both cases we trained these projections on the super dataset together (i.e., all raw and GAN records concatenated together). We used the standard Python packages t-SNE from Scikit Learn and UMAP from the umap-learn on conda-forge. Full code is in the code repository as a Jupyter notebook, but essentially raw and GAN generated datasets were broken into 512 datapoint length "windows" and input to the manifold function with dimensionality of 2 (which allows projection onto an x-y plane). To give an objective indication of whether there was a significant difference between manifold clusters between original and GAN simulated data, we tested for statistical difference between each pair with the R package ClusterSignificance (Serviss, Gådin, Eriksson, Folkersen, & Grandér, 2017), which tests the separation of the clusters by computing as score based via Euclidean distance .

Whilst helpful for visualising high dimensional data, t-SNE and UMAP are not without criticism. The iterative matrix-based approach can lead to poorer fits than PCA (Chari & Pachter, 2023). For our data however, the complex relationship between the timeseries warrants a more non-linear approach than a standard PCA.

3.3 Results

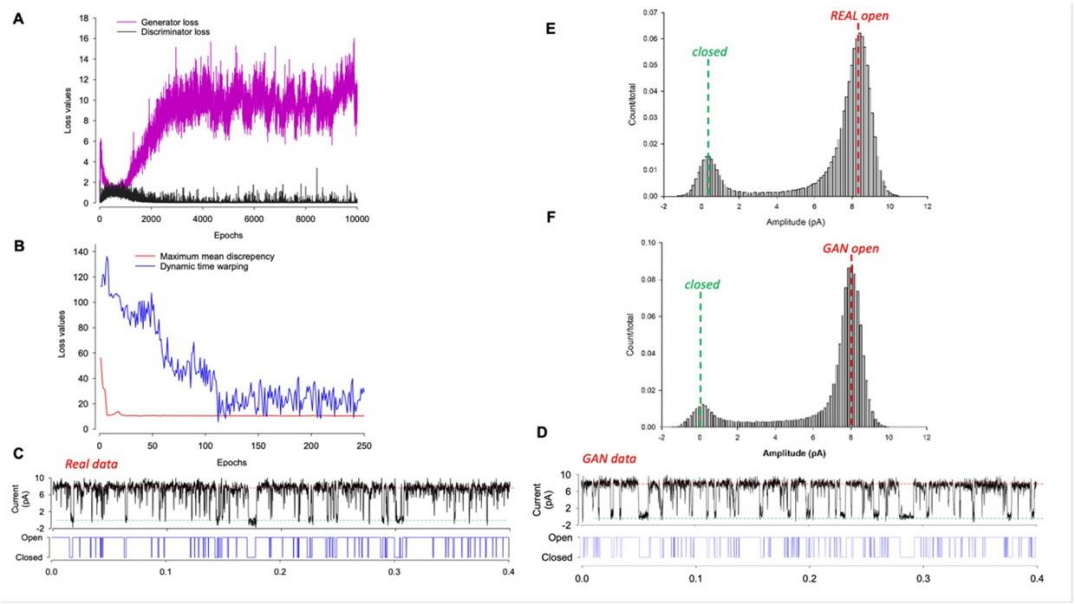


Figure 3.3: Realistic labelled single molecule data are synthesised by DeepGANnel.
(A) Discriminator and Generator losses plotted per epoch during training. (B) The maximum mean discrepancy (MMD) and dynamic time warping (DTW) evolution over the first 250 epochs as they approach equilibrium as measured by Dr Numan Celik. (C and D) Examples of real (C) and GAN synthesised (D) labelled data records. (E and F) All points amplitude histograms calculated from the real (E) and GAN synthesised (F) data.

Single (ion channel) molecule datasets were recorded using the patch-clamp technique and our standard protocols, and approximately 30 seconds was recorded under constant conditions (room temperature etc.). The resulting datasets were annotated (idealised) in QuB to produce a two-dimensional signal (dimension 1 = raw signal, dimension 2 = continuous annotation). Following robust scaling (Scikit learn) and reshaping these data were passed to the DeepGANnel model for approximately 10,000 epochs. Processing within each training epoch included augmentation with an invisibly small amount of Gaussian noise (approximately 0.1% of signal amplitude) applied to each data window. Figure 3.3A shows the characteristic evolution of discriminator and generator losses, with typical generator losses 10x or more the discriminator loss. Figure 3.3 C and D show examples of raw (real) input data and a representative strip of post train DeepGANnel

generated data. The two first analyses that are typically conducted in patch-clamp research are amplitude histograms and kinetic analysis. In addition to the objective metrics, amplitude histograms are shown in Figure 3.3E (real events) and Figure 3.3F (GAN simulated events), are also clearly similar in terms of size and shape.

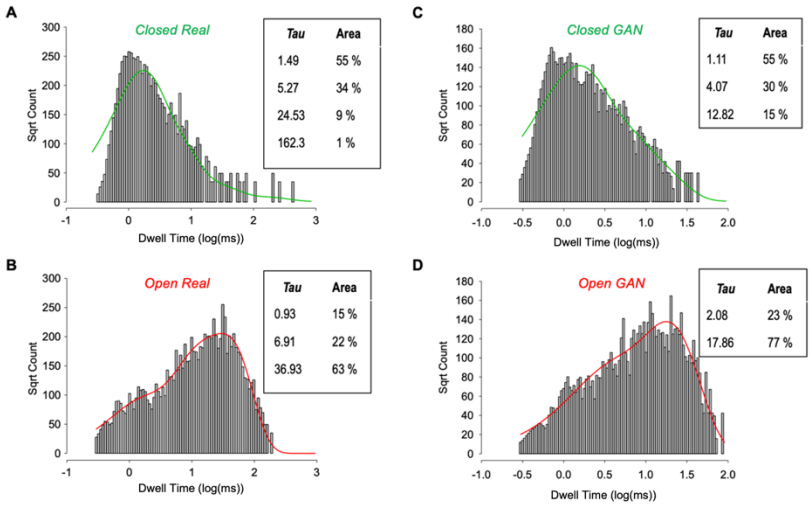


Figure 3.4: Comparative kinetic analyses of real and DeepGANnel simulated data
(A and B) Kinetic analysis provided by Dr Fiona O'Brian from the real (input) single molecule data with closed times (A) fit with 4 exponentials and open times (B) fit with 3 exponentials. The respective time constants and weights are displayed in the inset tables. (C and D) Kinetic analysis from the GAN simulated (output) single molecule data with closed times (C) fit with 3 exponentials and open times (D) fit with 2 exponentials. The respective time constants and weights are again displayed in the inset tables.

For an in-depth analysis we conducted full kinetic analysis of both raw (real molecular data) and GAN simulated data. These are shown in Figure 3.4, and there are similarities and differences between the real and GAN events. In terms of closed times, it is apparent that whilst the over-all distribution is similar between real (Figure 3.4A) and GAN (Figure 3.4C) there are differences. Kinetic anaylsis, performed by an experienced electrophysiologist and co-author on Ball et al, Dr Fiona O'Brian showed the real data included some long closed-events that are absent from the GAN simulated equivalent, perhaps due to the short window length as the generator output. In terms of open times again the over-all distribution is similar between real (Figure 3.4B) and GAN simulated

(Figure 3.4D), but there is attenuation of the very long open sojourns in the GAN simulated data.

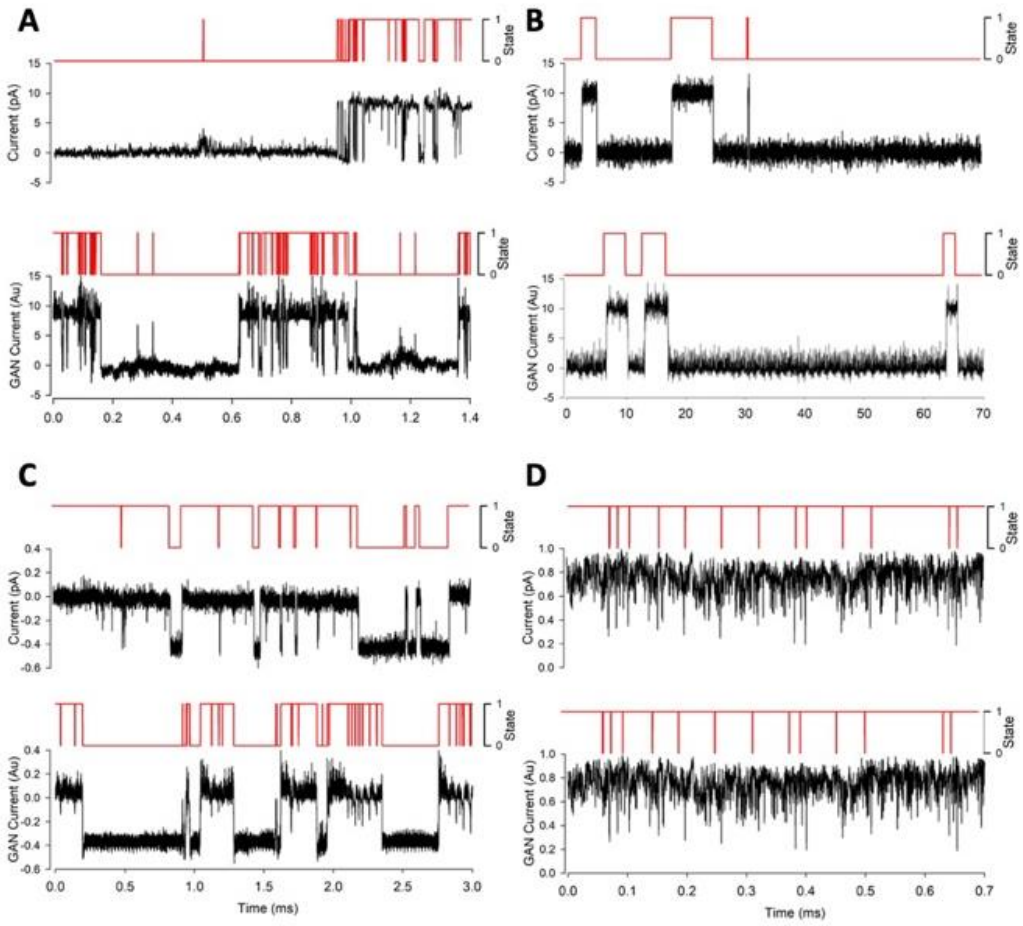


Figure 3.5: Sample data from both real dataset and DeepGANnel model output from numerous ion channels. *In each subpanel, real raw and labelled data is displayed on top with simulated data below. (A): "Channel phenotype A" canine articular chondrocyte sample data and model output. (B): "Channel phenotype B", WinEDR simulated sample data and model output. (C): "Channel phenotype C" tracheal chondrocyte sample data and model output. (D): "Channel phenotype D" PVN sample data and model output.*

To further test the ability of the GAN method to simulate a wide range of ion channel-like data we then trained DeepGANnel on 4 further ion channel phenotypes (see Table 3.1). Each of these is shown in Figure 3.5, where raw data, including the timeseries label, is shown a above the GAN simulated data. Subjectively, it appears each phenotype of channel is well represented by the matching GAN. To investigate more objectively how similar each GAN generated dataset is from its parent raw data (Table 3.1) we performed

t-SNE and UMAP dimension reduction comparisons between each of the 5-phenotypes of data (Figure 3.6). It is clear that there is a tendency for each GAN manifold cluster (t-SNE or UMAP) to align closely with its parent real data projection. Following this, we systematically analysed the cluster separation between real and GAN clusters, but first, we investigated whether this method could distinguish between each real vs real combination, and each GAN vs GAN combination. We focussed on UMAP projections. Table 3.2 shows the comparisons between UMAPs of the real data. Each real dataset is significantly different to each other, demonstrating the power of UMAP cluster analysis to objectively classify real world data. Secondly, the equivalent analysis focusses on UMAP comparisons of the 5 different GAN generated datasets is show in Table 3.3. In this case, all GAN simulated datasets are different from each other, except for the A and C datasets (different types of cartilage ion channels), which are not significantly different from each other. Finally, Table 3.4 shows that cluster-analysis of UMAP projections is unable to see a statistically significant difference between the canine articular ion channel data (Phenotype A) and its respective GAN, the tracheal ion channel data (Phenotype C) and its respective GAN or the equine cartilage data (Phenotype E) and its derived GAN data. However, the GANs produced from channel phenotypes B and D were significantly distinguishable from their parent datasets (Phenotypes B and D).

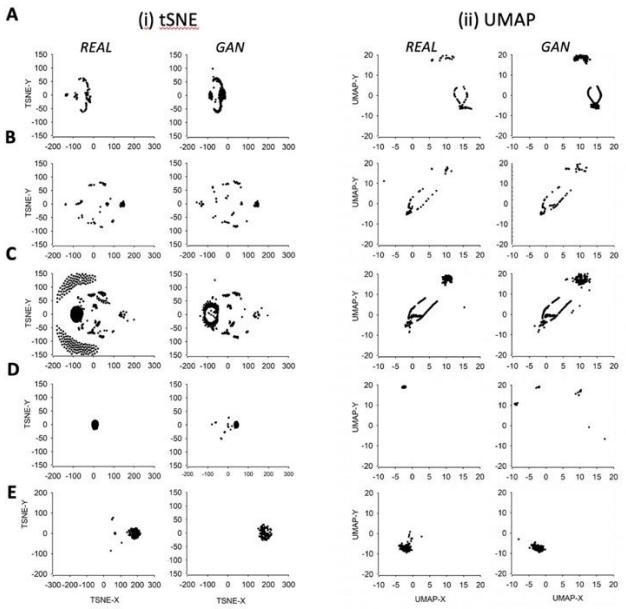


Figure 3.6: T-SNE and UMAP dimensional reduced visualisations for DeepGANnel and real data.
For each of the channel phenotypes shown earlier (A to E), (i) T-SNE and (ii) UMAP projections are shown with "GAN" data on the left and "real" source data on the right. Sets of projections A to D correspond to those channel phenotypes shown in Fig 5, (A): "Channel phenotype A" canine articular chondrocyte. (B): "Channel phenotype C" WinEDR simulated. (C): "Channel phenotype B", tracheal chondrocyte. (D): "Channel phenotype D" PVN data and (E) shows the same arrangement with the equine chondrocyte data shown in Figs 36 to 39.

Table 3.2: : Statistical separation of UMAP clusters between real datasets.
Maximum p-values calculated by the ClusterSignificance permutation package in R (Serviss et al., 2017). p-values are given after at least 1000 permutations. Phenotype codes described in the methods, Table 3.1.

	Phenotype A	**Phenotype B**	**Phenotype C**	**Phenotype D**	**Phenotype E**
Phenotype A	1	0.02	0.001	0.001	0.001
Phenotype B	-	1	0.02	0.001	0.001
Phenotype C	-	-	1	0.001	0.001
Phenotype D	-	-	-	1	0.001

Phenotype E	-	-	-	-	1

Table 3.3: Statistical separation of UMAP clusters between GAN generated datasets.
Maximum p-values calculated by the ClusterSignificance permutation package in R (Serviss et al., 2017). P-values are given after at least 1000 permutations. Phenotype codes described in the methods, Table 3.1.

	Phenotype A	**Phenotype B**	**Phenotype C**	**Phenotype D**	**Phenotype E**
Phenotype A	1	0.001	0.590	0.001	0.001
Phenotype B	-	1	0.001	0.001	0.001
Phenotype C	-	-	1	0.001	0.001
Phenotype D	-	-	-	1	0.001
Phenotype E	-	-	-	-	1

Table 3.4: Statistical separation of UMAP clusters between each real dataset and its GAN simulated equivalent.
Maximum p-values calculated by the ClusterSignificance permutation package in R (Serviss et al., 2017). p-values are given after at least 1000 permutations.

Phenotype	**Source**	**p-val**
A	Canine articular chondrocyte	0.739
B	WinEDR simulated	0.001
C	Tracheal chondrocyte	0.371
D	Paraventricular nucleus of the Hypothalamus	0.001
E	Equine articular chondrocytes	0.993

3.4 Discussion

In this work, we have generated synthetic raw single-molecule timeseries data along with continuous synchronised annotation/idealisation using a generative adversarial network (GAN) based on both real ion channel single molecule data from cultured chondrocytes. We demonstrate that the GAN generated raw ion channel data was similar to those obtained by real ion channel data. We assessed success of the GAN by three methods and in each they proved successful, but retain some limitations.

A central problem in single molecule, including "ion channel" research is that analyses of data is laborious and frequently requires a degree of expert hand crafting to complete. The first step in such analysis is idealisation of the record, or in machine learning terms, annotating or labelling. Each time point (of which there will be many million) needs to be annotated as to how many molecule pores are open at that instant. This then becomes,

effectively, a two-dimensional representation of the data. Currently, we are working with simple datasets with one type of molecule in the dataset, but in the future such analysis will extend this to more complex datasets. Clearly new analysis methods will also be necessary to get the maximum amount of information from complex single molecule data. A number of tools have been developed to address these issues (Colquhoun, Hatton, & Hawkes, 2003; Gnanasambandam et al., 2017; Juette et al., 2016; Nicolai & Sachs, 2013) including our own Deep-Channel deep learning model (Celik et al., 2020). For further development of similar or enhanced tools there is a lack of available training data. There are two clear choices for such data; (i) real biological data that has been annotated in some way or (ii) synthetic datasets. Both approaches have biases and severe limitations. Real data cannot be perfectly labelled; the real ground truth is unknowable and so will only be an approximation. In practical terms, the longer the length of the real data, the greater the "ground truth" errors will be. As a result, new machine learning methods will learn the errors of the existing technology. Furthermore, only simple datasets can be annotated and so this sets an upper limit on the complexity of the datasets that could be analysed by potential new tools. However, synthetic datasets also contain many biases and limitations, some of which may be entirely unanticipated or recognised. Therefore, the starting problem that our work addressed here was to create very large datasets of ion channel single molecule activity that could be used to develop single molecule analysis tools. We chose to investigate if GAN technology could provide a useful alternative source of data. In principle this would have the advantages of synthetic data in that one could produce unlimited amounts, but still retain nuance and subtle authenticity missed by mere simulation. We are also hopeful that such synthetic data could be used in development of ion channel modelling software and may allow for a novel type of data inference, extracting critical features in datasets that may be overlooked by traditional analysis. Visual inspection, and analysis of our similarity metrics demonstrates that DeepGANnel can reconstruct faithfully simulate a number of different ion channel phenotypes. Clearly the datasets are not identical (that would be simply a copy), but raises the question of whether the simulation is good enough to be useful. To examine

this, we show in Figure 3.7 an example experiment. We train our Deep-Channel model to analyse/idealise the data from phenotype C (Table 3.4, Figure 3.55C). Half of the small length of original raw data (75k datapoints equivalent to approx. 8 seconds of recording) is clearly insufficient to allow Deep-Channel to learn the appropriate features and idealise the remaining half of this dataset (Figure 3.7 A, B). However, DeepGANnel, once trained can produce any amount of data similar to this original dataset (See Figure 3.5C). We then trained Deep-Channel on this GAN dataset, 5x the size of the original, and performance is now acceptable (Figure 3.7D, E, F).

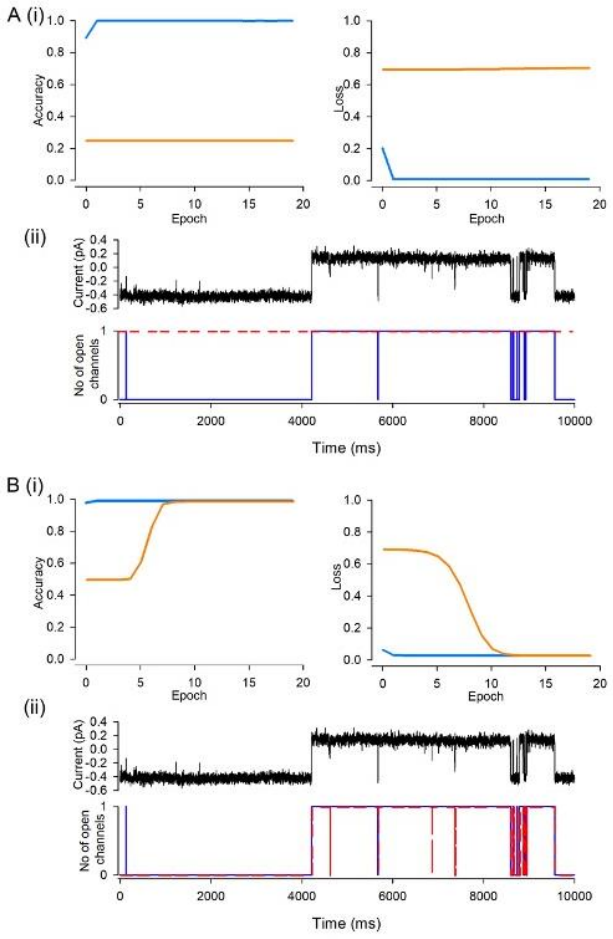


Figure 3.7: Potential of DeepGANnel to facilitate Deep Learning Training on Physiological time series data. *This demonstration uses data from ion channel Phenotype C. The original raw data is 150k sample points. This was split into two 75k data sets, for training and for validation by our previously published labelling network Deep-Channel (13). (A) Shows the training and validation accuracy; the accuracy on the training set itself is near perfect, but it fails to an accuracy of 0.5 (to predict open or closed) with the unseen validation dataset. (B) Shows similar to (A), but with losses. Complete failure with the unseen validation set. Ergo, model training fails as is illustrated in (C) where the raw signal (top) is shown above the synchronised ground truth black and prediction red. In the second part of this demonstration (D, E and F), we replace the training data with the GAN generated dataset of Phenotype*

C, which could be any length, but in this example is 5x larger than the original. Now Deep-Channel does much better. After about 5 epochs, performance against the validation dataset improves in terms of both accuracy (D) and loss (E) and the final product is well trained model (F). There is close agreement between the ground truth (black) and the prediction (red).

3.4.1 Comparisons of GAN methods to traditional synthesis methods

The new GAN synthesis method will have strengths and weaknesses compared to stochastic Markovian model-based processes. (1) Speed: Training a GAN takes a considerable time (hours), but simulation itself takes ms per record. The usual stochastic methods produce records at the ms per record timescale, but do not need training. They still need laborious hand customising to simulate a particular phenotype of channel. Our software only Markovian based simulations, written in Python, take about 8ms to generate one "record" whereas DeepGANnel, on our GPU workstations takes about 4ms (once trained). It should be noted that Stochastic models will take n times longer to simulate data records with n channels within, whereas the GAN method would need retraining to a multi-channel dataset; but still take the same 4ms per record once trained. The method we used to create the training set in our DeepChannel project (Celik et al., 2020) used stochastic simulation followed by passing these data through a real patch clamp amplifier. This method was the slowest of all, producing data at less than real time (so approximately 1000x slower than other methods). (2) Authenticity: Speed is not the objective of this work however, it is "authenticity". In the present paper, we provide metrics for authenticity in terms of UMAP/t-SNE cluster similarity for DeepGANnel, however there is no absolute way to do the same with stochastic methods. The more effort the user puts into it, analysing noise and reproducing these, measuring single channel properties and encoding this, the closer it would become, but this entire laborious process would need to be repeated for every type of ion channel and condition to be simulated. With DeepGANnel, the user simply points the script at a new set of seed data. Fundamentally, stochastic data may include unrecognised biases since every feature must be hand crafted, and DeepGANnel would include artefacts found in native data that would likely be omitted using a stochastic approach. The inclusion of these in training data

would be important for development of robust analysis software. (3) A priori assumptions: DeepGANnel needs none, but stochastic simulation requires every detail to be estimated. However, stochastically generated data has the advantage that the experimenter could potentially develop software similar to HJCFit (Colquhoun et al., 2003; Gibb et al., 2018) and QuB (Nicolai & Sachs, 2013), to recover the hidden Markov model (HMM), and validate accuracy. With a DeepGANnel simulated dataset the underlying HMM would be unknown, and need to be estimated with further software such as HJCFit, in the same way as one would do with real patch clamp data.

Training of deep learning models such as DeepGANnel require some thought about loss functions and metrics. For example, in this work we use T-SNE and UMAP to measure the efficacy of the method on maintain the "characteristic nature" of the channel, a concept of which is extremely complex hard to formally define. Other, traditional measures of a model performance (such as studying the open/closed distributions of the output) may be misleading; a model may be able to reproduce a statistically identical distribution in one measure but still "look" fundamentally different from the real data (for example, if the noise is largely different). Furthermore, some usually unwanted elements of the data (such as artefacts or baseline drift), are highly desirable in the GAN output as they more accurately represent what is expected from lab recordings.

3.4.2 Previous use of GAN for time-series data

The literature includes previous examples where timeseries data can be simulated with a GAN (For example, (Zhu, Ye, Fu, Liu, & Shen, 2019)), but these lack the synchronous labelling critical in single molecule analysis or many other physiological studies. Generating single-molecule timeseries with a basic GAN (Vanilla-GAN) model should be very effective at producing authentic ion molecule signal patch-clamp signal, but this would lack the output of the critical timepoint-by timepoint labelling necessary to meet our goals of creating valid alternative datasets. However, we have shown that using a 2D-CNN based model in a shape of (samples x 2 x 1280 x 1) is very effective; with a small amount of carefully annotated seed data generating unlimited synthetic copies. In

Figures 3.5 and 3.7 we show a direct comparison for a typical electrophysiological work up of both the original and DeepGANnel synthesised data. On the supplementary information and public repository (https://github.com/RichardBJ/DeepGANnel), we include a movie of the training process. The match between synthetic kinetic analysis and the real data is not perfect, but rather close. The notable exception is that the longer states (both open and closed) are missing or have a diminished representation. We attribute this to the necessity to use a finite window ("image" width or "record length") size. Also as stated in the methods this was cropped to remove leading and trailing artefacts. Perhaps models using far greater window lengths would be possible, but this does not appear to be a major problem for our purpose (since most single molecule events durations are within this window) and it would increase the model complexity many-fold. This model took 24-48 hours to train on our system and note that performance peaked but would deteriorate if it was left indefinitely. Our code allowed automatic adjustment of learning rate as epochs progressed, but we still chose to stop the modelling manually. The ever-increasing GPU power make ever larger window sizes less of an issue in the future. We summarise these strengths and weaknesses in Table 3.5.

Table 3.5: Likely strengths and weaknesses of DeepGANnel versus traditional synthesis methods.
This table summarises the pros and cons of DeepGANnel discussed and justified in the text. By definition such comparisons can only be subjective because Traditional Methods vary (entirely dependent on a priori assumptions, that could be simple or complex), as do computing platforms.

	Stochastic Simulation	**DeepGANnel**
A priori assumptions	Everything must be estimated or assumed; channel size, rate constants, open channel noise, thermal noise levels, artifact frequency etc.	None required.
Authenticity	Depends entirely on the accuracy of a priori-assumptions.	Highly authentic.

Speed	Moderate speed.	Slow to train, fast to simulate thereafter.
GPU needed	Typically, these are not used. Future stochastic models may use them.	Realistically these are needed for training, although not for simulation itself.
Markov Model	Could include Markovian model structure.	May include Markovian structure, but this is not guaranteed.
Need for seed data	No. The data can be completely imaginary.	Yes
Fully Labelled data	Yes	Yes

3.4.3 Future applications of GAN in electrophysiology

The potential for further exploitation, of GAN technology in electrophysiology beyond the current use for creation of datasets is immense. One future goal will be to simulate far more complex data signals, but we still have the limitation on how to acquire the fully annotated seed data in the first place. Potentially painstaking manual annotation of very short sections of data with known numbers of ion channel molecules by several human experts would be possible. Furthermore, it is possible that single-molecule GAN could be used more directly in electrophysiological modelling. Currently, single molecule behaviour within such models is derived by a set of differential equations based on a set of measured or even estimated parameters derived from curve fitting lab-recorded data (Feetham, Nunn, Lewis, Dart, & Barrett-Jolley, 2015), but it may be possible in the future to use GAN to generate more realistic stochastic behaviour directly. Additionally, future studies will investigate whether interpretability methods can be used to identify important, defining features within each different dataset that are missed either by eye or by standard single molecule analysis techniques. The DeepGANnel approach will prove especially useful for generation of "raw" data, along with ground truth annotations as analysis methods move to using a big data approach; creating large datasets where underlying models are unknown or too complex to simulate stochastically. Potentially also, our methods could be used to augment data in an analysis pipeline, for example to facilitate secondary analysis where only small samples of data are available.

The architecture we present here, using deep learning to generate physiological timeseries data with continuous annotations, could also be adapted easily for additional usability for equivalent systems in physiology. For example, action potential or electrocardiogram simulation. As proof-of-principle we show here that indeed DeepGANnel can easily synthesise telemetered ECG signal, again fully annotated (Figure 3.8). In this example the annotation dimension (annotations were provided by Elaheh Sayari is merely beat (binary state 1) or no beat binary (0), but this could easily be extended to include P-wave (categorical state 2), T-wave (categorical state 3), or abnormal event (categorical state 4)

etc with trivial code adaptation. Another example would potentially be Nanopore data which has similar data output to patch-clamp data.

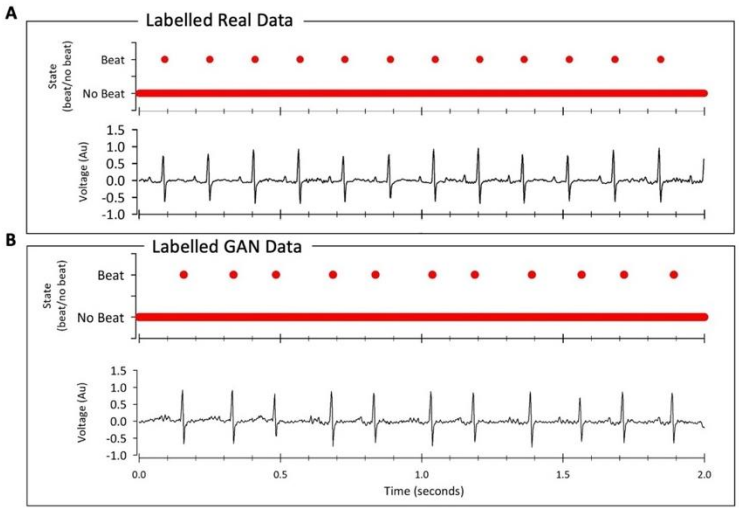


Figure 3.8: General usability of DeepGANnel for 2-dimensional physiological time series data.
In this example we fed rodent ECG data (dimension 1) along with a beat annotation (dimension 2, peak of r-wave, provided by Elaheh Sayari) into DeepGANnel and trained with 10,000 epochs until realistic rodent ECG data was generated. Data collected via electrocardiogram transmitters from male Wistar rats (ETA-F20; Data Sciences International, St Paul, MN, USA) as previously described (44), briefly ECG signal was digitized to a PC with a CED Micro1401 using Spike2 at 5 kHz. In principle this could also be encoded with further annotations such as t-wave (note rats do not show a significant p-wave).

In summary, GANs are increasingly proving a viable method to generate synthetic datasets for biological research, and here we show an implementation that allows simulation of time dependent single molecule (patch clamp ion channel protein) activity along with a continuous state annotation that is extendable for an array of physiological uses. For the remainder of this thesis, however, we used Markovian based synthesis based methods described in Chapter 2, rather than DeepGANnel, since one of the limitations reported above is the lack of a ground truth Markovian state, critical to development of the novel models developed in this PhD.

4 Testing Canonical Forms of BK Channel Markovian Models

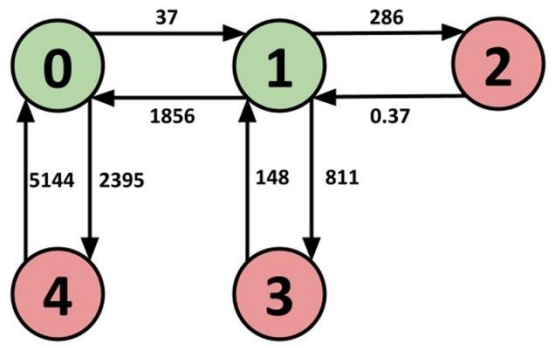
4.1 Introduction

As discussed previously, in normal physiology, ion channels pass-through a number of states in a Markovian process (Bruno, Yang, & Pearson, 2005; Sivilotti & Colquhoun, 2016). These states correspond to open and closed configurations of the channel; with the potential for many hidden states to correspond to either conformation. This leads to a situation where individual ion channel behaviour can be described by a continuous time hidden Markovian structure.

Continuous time hidden Markov models (CT-HMMs) can be defined with a set of hidden states S, visible states V, a hidden transition rate matrix Q, and an emission matrix E. The rows in Q sum to zero, in a way such that the diagonal entries q_{ii} are the negative sum of the other entries in the row. The dwell time in each state $i \in S$ is exponentially distributed with $f(t) = q_i e^{-q_i^t}$, and the probability of moving from state i to j being q_{ij}/q_i, where $q_i = \sum_k q_{ik}$. In addition to this hidden process, for each dwell a visible state $z \in V$ is selected from the emission matrix E, with probability v_{iz}/v_i, where $v_i = \sum_k v_{ik}$. In practice, the hidden Markov state is not visible, but the visible states from the emission matrix are, along with how long they spend in those visible states (the dwell times). These transition rate matrices govern the underlying mechanism of the process, but are non-trivial to obtain.

Some of these Markov transition rate matricies are simple such as the "5 state' Markovian model of a BK Channel described in Chapter 2.1.1 (Davies et al., 2010) (Figures 4.1, 4.2) whereas others can be highly complex with many more hidden states, such as the Cox model of a BK channel (Cox, Cui, & Aldrich, 1997) (Figures 4.3, 4.4).

Furthermore, we can define arbitrarily simple Markovian models such as the "3 state" model in Chapter 2.1.1 for an easier to understand example (Figures 4.5, 4.6).



$$\begin{bmatrix} -2432 & 37 & 0 & 0 & 2395 \\ 1856 & -2953 & 286 & 811 & 0 \\ 0.37 & 0 & -0.37 & 0 & 0 \\ 148 & 0 & 0 & -148 & 0 \\ 0 & 5144 & 0 & 0 & -5144 \end{bmatrix}$$

A **B**

Figure 4.1: The "5-state" Markovian model diagram (A) and its corresponding transition rate matrix (B).
The "5 state" model is taken from Davies et al. as a rate-fitted model of a BK channel's kinetics. A shows the model diagram, with green states representing open states and red closed states. The transition rate matrix (B) shows how this is interpreted mathematically, with the negatives on the diagonal corresponding to the mean time to spend in a state. The rest of the rates on a row show the proportional chance to change to that column's state – for example in open state 0 there is a far higher chance of travelling to closed state 4 than open state 1. The open-open connection makes training deep learning networks difficult, as a changes in state from 0 to 1 will not cause a change in conductance level.

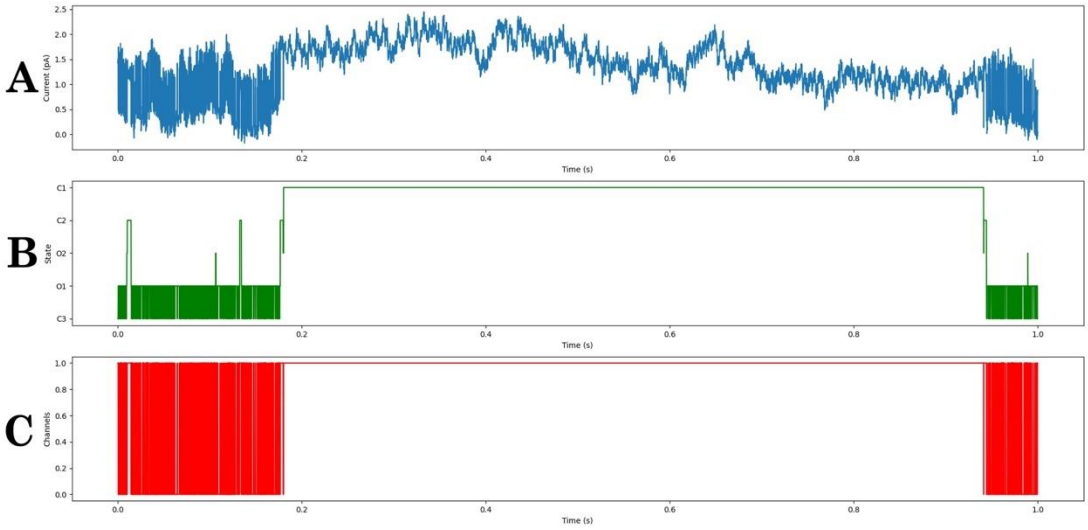
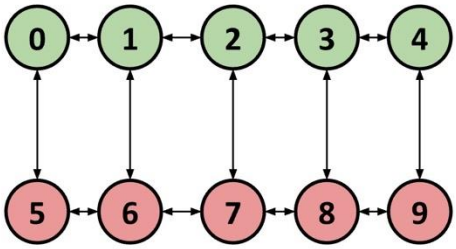


Figure 4.2: Example output of a simulation from the "5-state" model at 10kHz. (A), it's corresponding state at each timepoint (B), and the channel's open/closed configuration at each timepoint (C).
Data was simulated at 10kHz with 1/f^n noise added in post. In the 5-state model, we see a significant amount of "flickering"; events which are extremely short and only a few samples long. This is common within ion channel recording, but such events are sometimes filtered out using a dead-time filter.

A



B

$$\begin{bmatrix} -71612 & 68000 & 0 & 0 & 0 & 3612 & 0 & 0 & 0 & 0 \\ 1080 & -7856 & 5100 & 0 & 0 & 0 & 1676 & 0 & 0 & 0 \\ 0 & 2160 & -37134 & 34000 & 0 & 0 & 0 & 974 & 0 & 0 \\ 0 & 0 & 3240 & -20729 & 17000 & 0 & 0 & 0 & 489 & 0 \\ 0 & 0 & 0 & 4320 & -4444 & 0 & 0 & 0 & 0 & 124 \\ 2 & 0 & 0 & 0 & 0 & -68002 & 68000 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 9030 & -60030 & 51000 & 0 & 0 \\ 0 & 0 & 38 & 0 & 0 & 0 & 18060 & -52098 & 34000 & 0 \\ 0 & 0 & 0 & 160 & 0 & 0 & 0 & 27090 & -44250 & 17000 \\ 0 & 0 & 0 & 0 & 340 & 0 & 0 & 0 & 36120 & -36460 \end{bmatrix}$$

Figure 4.3: A ten state Markovian model diagram from Cox et al. (A) and its corresponding transition rate matrix (B).

This ten state model is more complex rate-fitted model of a BK channel's kinetics. A shows the model diagram, with green states representing open states and red closed states (due to the number of transitions, rates are omitted from the diagram but visible in the matrix). The transition rate matrix (B) shows how this is interpreted mathematically, with the negatives on the diagonal corresponding to the mean time to spend in a state as in Figure 4.1a. As with the 5-state model, we see many open-open transitions and close—closed transitions; this makes analysis significantly more difficult.

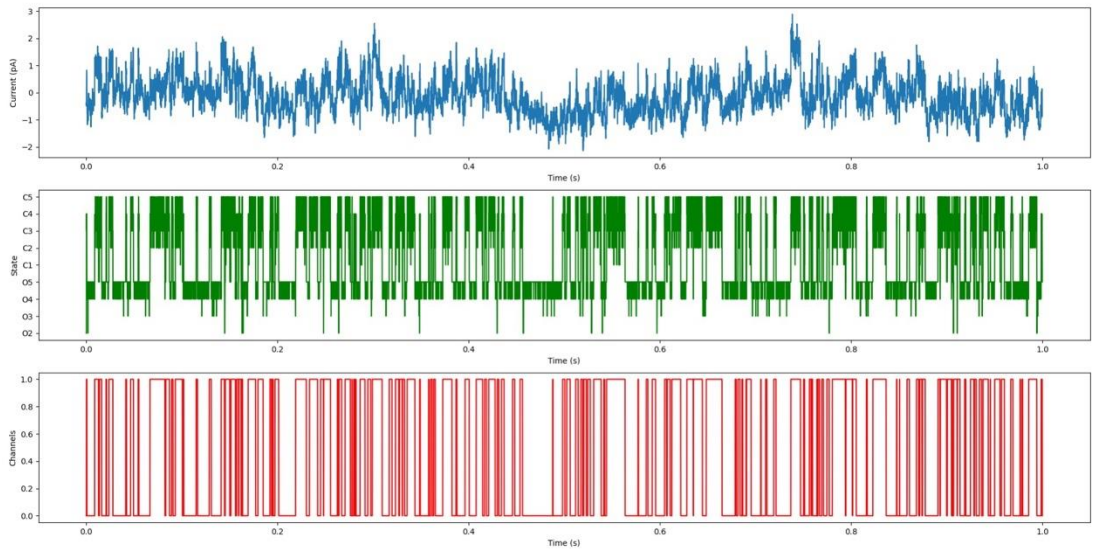
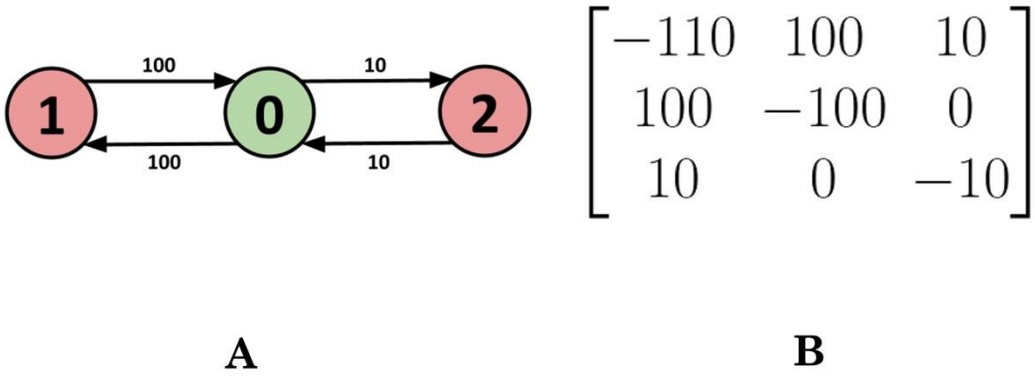


Figure 4.4: Example output of a simulation from the ten state model at 10kHz. (A), it's corresponding state at each timepoint (B), and the channel's open/closed configuration at each timepoint (C).

Data was simulated at 10kHz with 1/f^n noise added in post. Here we see the model rapidly changing within the open states or closed states having no impact on the channel's open/closed configuration (in particularly around 0.5s, where the channel switches between O_4 and O_5 many times).



$$\begin{bmatrix} -110 & 100 & 10 \\ 100 & -100 & 0 \\ 10 & 0 & -10 \end{bmatrix}$$

Figure 4.5: The "3 state" Markovian model diagram (A) and its corresponding transition rate matrix (B).
The "3 state" model is the simplest Markovian model with multiple hidden states corresponding to one open/closed visible state (in this case, two closed states in red). A successful Markovian deep learning model will not only detect the open/closed nature of the data, but identify which closed state it is in given its dwell time. The transition rate matrix (B) shows the structure of a Markovian transition rate matrix. The dwell times are entered in the non-diagonal entries, with the diagonal entries being equal to the negative sum of the other row values. The 3-state model topology was chosen as the simplest possible model that still exhibited multiple hidden states with the same open/closed configuration; with the rate constants chosen for a clear discrimination for the two types of closed events (a "long" close and a "short" close).

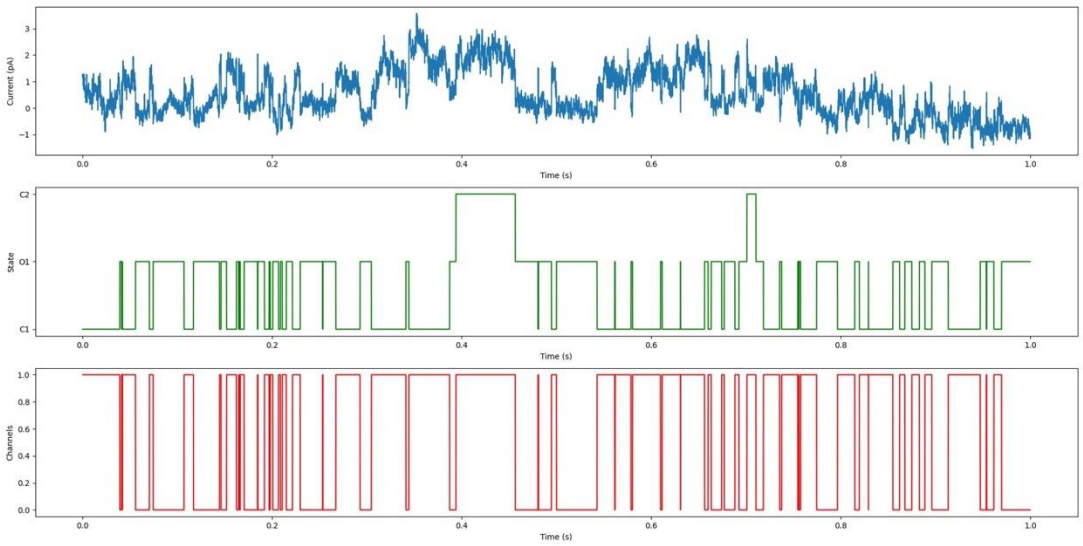


Figure 4.6: Example output of a simulation from the "3 state" model at 10kHz. (A), it's corresponding state at each timepoint (B), and the channel's open/closed configuration at each timepoint (C).

This model's parameters were chosen specifically for longer dwell times that are visually easy to resolve; with C2 forming (on average) dwell times ten times as long as C1. The stochastic nature of this process (and why C1 still exhibits some long dwell times), along with a method to mitigate the impact of it is discussed in more detail in Chapter 6.

Markov models are represented by their transition state matrices, Q, which can be partitioned by rearranging the rows and columns into rates between closed states Q_{CC}, closed to open states Q_{CO}, open to closed states Q_{OC}, and open states Q_{OO}:

$$Q = \begin{bmatrix} Q_{OO} & Q_{OC} \\ Q_{CO} & Q_{CC} \end{bmatrix}$$

From this representation the joint distributions for dwell times can be written as:

$$f_c(t_c) = \pi_c e^{Q_{CC}t_c} Q_{CO} u_o$$

$$f_o(t_o) = \pi_o e^{Q_{OO}t_o} Q_{OC} u_c$$

$$f_{co}(t_c, t_o) = \pi_c e^{Q_{CC}t_c} Q_{CO} e^{Q_{OO}t_o} Q_{OC} u_c$$

$$f_{oc}(t_o, t_c) = \pi_o e^{Q_{OO}t_o} Q_{OC} e^{Q_{CC}t_c} Q_{CO} u_o$$

Here π_c, π_o are row vectors whose ith element is the initial state probability that a closed/open interval begins in closed/open state i, and u_c, u_o are column vectors of ones, one for each open/closed state.

Kienker (Kienker, 1989) describes that if these joint distributions are equal for two ion channel Markov structures, their kinetic behaviour will be identical. Furthermore, Kienker's work identifies that multiple transition rate matrices produce the same set of distributions; meaning there can be many correct transition rate matricies fitted for one set of ion channel data.

One of our goals in this project (see Objectives, section 1.5 b)) is to attempt to directly recover Markovian state using deep learning architecture. Crucially, deep learning algorithms depend on a single source of "ground truth" to train; most Markovian models will have a number of functionally identical formulations; that when simulated will give

an equivalent timeseries and kinetic analyses. This may make model evaluation unreliable as a model may predict an equivalent, but different model to the physiological dogma (such as existing Markov schemes in the literature) and therefore give seemingly incorrect outputs.

A representational form for a class of equivalent Markov models that give equivalent functional outputs is known as the *canonical form*. For example the Bauer-Kienker Uncoupled (BKU) form (Barbi & Petracchi, 1992; Kienker, 1989) results in a structure with no closed-closed or open-open transitions (Figure 4.7). Restricting our dataset to only include canonical forms would eliminate this equivalence concern. Furthermore, the BKU form specifically may increase model performance, as a change in state always corresponds to a change in open/closed configuration; and therefore a large change in current. In Kienker's work however, he discusses that this form may contain negative rate constants leading to a physiologically infeasible model; but the conditions for which this infeasibility occurs is not explored.

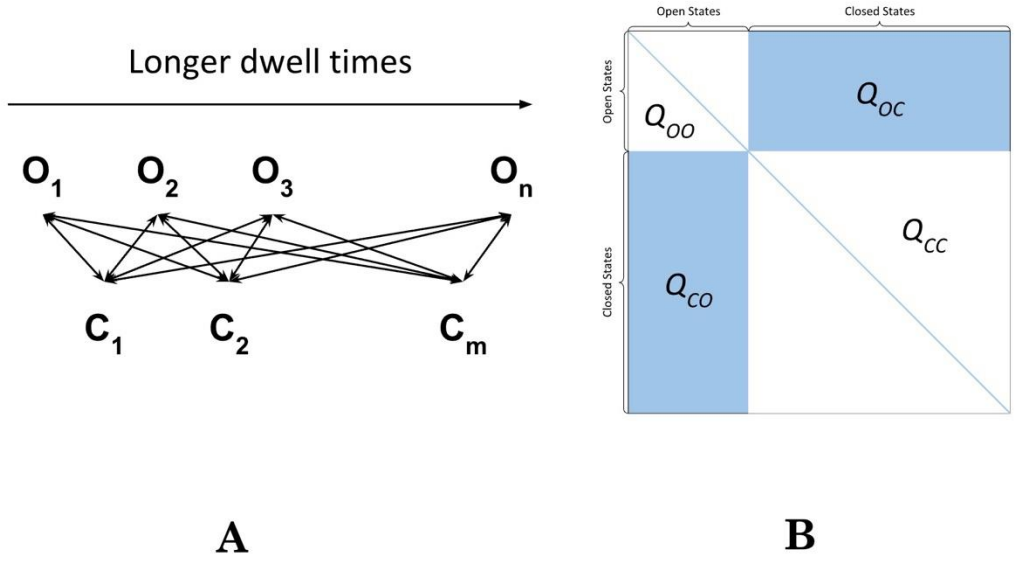


Figure 4.7: (A) Example of BKU form structure with ordering (B) BKU structure of a transition rate matrix. *Equivalences between continuous time hidden Markov models present a problem for machine learning algorithms, as the existence of a consistent, unique ground truth state is not possible, as a neural network model might internally derive a different, but equivalent model to the one used to generate the labels. An approach to solving this is to use an applied version of the Bauer-Kienker uncoupled form (A), known as a type of canonical form that exists for every ion channel Markovian model. The transition rate matrix of a BKU canonical form takes a unique structure (B); blue areas represent non-zero terms. Notice how the Q_{CC} and Q_{OO} matricies are diagonal (only have non-zero terms along the diagonal). This form is achieved by diagonalisation of the Q_cc and Q_oo matricies separately, followed by a*

calculation of the resulting Q_co and Q_oc matricies as these do not have to be diagonal. This structure directly results in a decoupled Markovian structure, where no closed-closed or open-open connections are present.

Therefore, the aim of this chapter is to generate the canonical forms for Markovian models of ion channels to examine their feasibility for use in deep learning training.

4.2 Methods

As detailed in Chapter 2.1.1, we use 2 Markov models for the data simulation for Markovian deep learning. Along with these two models, we will consider a third, more complex model for the calculation of a canonical form to examine if more complex Markovian models reliably convert to a canonical form.

We start by rearranging the transition rate matrix Q into the form above, splitting the matrix into rates between closed states Q_{CC}, closed to open states Q_{CO}, open to closed states Q_{OC}, and open states Q_{OO}:

$$Q = \begin{bmatrix} Q_{OO} & Q_{OC} \\ Q_{CO} & Q_{CC} \end{bmatrix}$$

Note that Q_{OO} and Q_{CC} must both be square, but Q_{CO} and Q_{OC} not necessarily so. Then if P is our matrix in BKU form, then by definition, P_{OO} and P_{CC} must both be diagonal matrices, since there are no connections between open and open, or closed and closed states. Kienker describes that to convert a matrix into BKU, we need to diagonalize Q_{OO} and Q_{CC}; typically this is achieved using a transformation matrix S of the form:

$$S = \begin{pmatrix} S_{OO} & 0 \\ 0 & S_{CC} \end{pmatrix} \tag{22}$$

Where S_{OO} and S_{CC} are made up of the eigenvectors of $-Q_{OO}$ and $-Q_{CC}$ respectively (negative since we want the diagonals to also be negative, to satisfy the Markov transition rate matrix definition). We also choose the multiplicities of these eigenvectors so that the

rows of the transformation matrices sum to one. We can then transform Q to BKU form by using:

$$P = S^{-1}QS \tag{23}$$

The resulting transition rate matrix for the method above is only unique up to the relabelling of states; however this can be fixed by labelling states in order of the negative diagonal value; if sorting in ascending order (most negative first), this corresponds to the 'shortest' states first, and the 'longest' states last. (Figure 4.7).

To achieve this with our data processing pipeline, the DeepMICA library includes code for converting a Markovian model to canonical form (https://github.com/stmball/DeepMICA/blob/main/generate.py#L611). For this work, the Markovian models were encoded into the DeepMICA library and converted using this method.

4.3 Results

The first of our models (the "3 state model") is already in canonical form; as there are no open-open or closed-closed connections.

The second of our models (the "5 state model") is not in canonical form – conversion to canonical form yields the transition rate matrix and diagram seen in Figure 4.8 and is physiologically invalid as there are negative non-diagonal rate constants between the first open state and the second and third closed states.

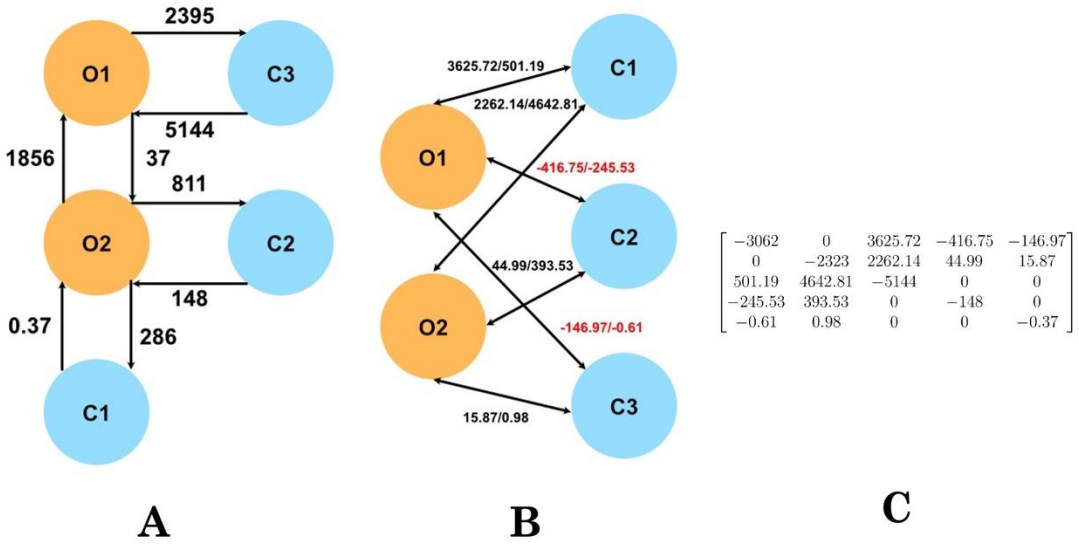


Figure 4.8: "5-state" Markovian model and its BKU Canonical Form (B, C).
An experimental model taken from literature (Davies et al., 2010) derived via curve fitting (A) does not produce a valid BKU canonical form (B,C) due to negative rate constants between states (shown in red). Since we cannot sample from an exponential distribution with negative constants, the canonical form is an invalid Markovian model and we cannot simulate data from it. B shows this diagrammatically; whereas C shows the canonical transition rate matrix.

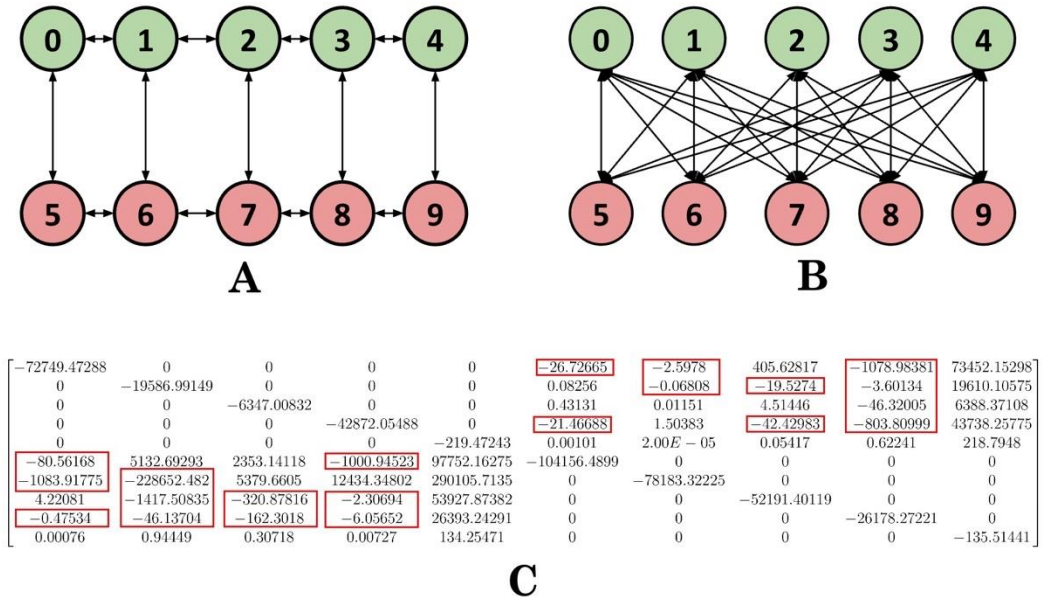


Figure 4.9: Ten state Markovian model (A) along with its BKU Canonical (B,C) .
The other experimental model from Cox et al. yields the canonical form seen in B and C. Due to the number of parameters in the canonical model, labels were omitted and are instead visible in the transition rate matrix C. We see that although the canonical structure is correct, we have many negative non-diagonal rate constants (circled in red) in the resulting transition rate matrix, making this an invalid Markovian form; and therefore impossible to simulate data from.

Finally, the third of our models from Cox et al. (Cox et al., 1997) is also not in canonical form initially. Conversion to canonical form again gives many non-diagonal rate constants and is therefore physiologically invalid (Figure 4.9).

4.4 Discussion

We have calculated the canonical forms for 3 Markovian models, two of which are from the physiological literature. Of these 3 models, one was already in canonical form, and the other two produced physiologically invalid canonical forms after the diagonalisation process was performed.

These canonical forms would have been helpful for deep learning as they would have improved reliability of the models in production; as there would be no chance of an equivalent but different model being predicted by the model compared to what was expected by the user. However the negative rate constants for the physiological models would be unhelpful in further analysis and therefore are not worth considering; no simulations can be done for a transition rate matrix with negative constants as the exponential distribution requires these parameters to be strictly positive.

Due to the linearity of the transformation given by Kienker (Kienker, 1989), small adjustments to the original transition rate matrix will only have a similarly small effect on the canonical matrix – meaning we cannot make a small consideration to our model to allow a valid canonical form. Similarly, drug action that affects ion channel kinetics significantly (and therefore produces large changes in the transition rate matrix) may push the Markovian model into having an invalid canonical form. This drawback makes the BKU form difficult to use in practice.

For a canonical form to exist without allowing for relabelling of state numbers, we order the states within the open and closed classes by their expected dwell times. Under the effects of substances the kinetic structure of the ion channel may change, and this

ordering may become inaccurate. This arbitrary ordering introduces discontinuities in the canonical form.

One solution would be to use a different canonical form; other canonical forms have been suggested as alternatives to the BKU canonical form such as the MIR form (Bruno et al., 2005), and another from Larget (Larget, 1998) however both of these canonical forms have open-open and closed-closed connections. Intuitively, these state transitions would be harder for a network to detect from a current trace as there would be no accompanying change in channel opening, and hence no "current jump" as with closed-open state transitions for the model to detect.

5 A Comparison of different neural network architectures for Naïve recovery of Markovian State on simulated data

5.1 Introduction

It is thought that the underlying mechanism in which ion channels open and close is Markovian; that is, in addition to possibly having a single "open" and "closed" state, ion channels may instead move through a number of different "open" and "closed" states each with different mean dwell times (McManus & Magleby, 1988; Yoshida, Oda, & Ikemoto, 1991). An important task in single channel analysis would be the ability to recover such Markovian states, especially in diseases or in the presence of substances that may cause subtle changes in kinetics and the Markovian process, with potentially dangerous effects.

It is also thought that Markovian states may have some relation to the physical configuration of the protein (Cox et al., 1997) and recovering this Markovian state from a patch clamp electrophysiology signal is therefore of great interest to researchers as it may encode additional structure-function information about ion channels. Although existing methods exist to recover Markovian states retrospectively from an ion channel signal, these either idealise the ion channel signal beforehand, or are still relatively computationally expensive.

Deep learning has been proven to recover complex information from noisy data and has also been shown to have high-accuracy idealisation (beating other methods) for ion channel recording idealisation, under a limited set of conditions (Celik et al., 2020)

Therefore, this chapter aims to develop and test deep learning models that can directly recover Markovian states, bypassing the traditional workflow of applying the Baum-

Welch or Viterbi algorithms to idealised ion channel data to recover the underlying Markovian state. We conduct a quantitative comparison of several deep learning model architectures, identifying the advantages and disadvantages of each. This analysis is carried out while adhering to common pre-processing steps in the pipeline.

5.2 Methods

5.2.1 Model Design

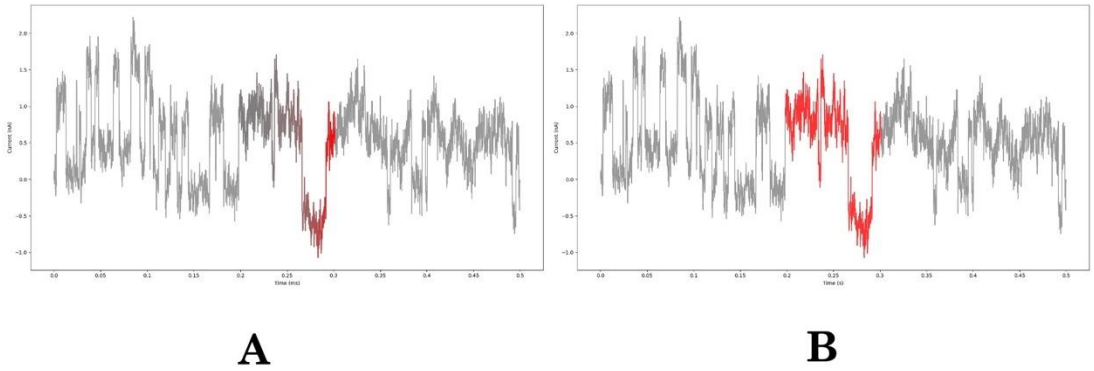


Figure 5.1: Different approaches to classification using RNN (A) and CNN (B) based architectures.
In the recurrent architecture, points are sequentially classified one-by-one in the model (shown in red), using previous information with decreasing weights as the time distance increases. For LSTM based RNN networks, this "fading" weighting (represented by red) on past data is not infinite and gets cut at a trained number of points. For CNN models, we process a number of points together, forming "windows". The advantage of this case is far larger throughput (in our case, 1024 points get processed at a time) – however the models tend to be a lot larger due to the fully connected layer at the end.

We design 6 model architectures: SimpleCNN, LSTM, DeepChannel, SplitCNN, ResNet amd UNet. The first two models are simple architectures centred around two types of deep learning cell (LSTM recurrent and 1D-CNN convolutional). These are essentially small pilot models providing a baseline to see what type of cell is most promising. The recurrent model predicts the individual points on a point-by-point basis in turn, with the recurrent nature of the model allowing for past points to be used as an input to the recurrent cells. Due to the nature of LSTM cells, the number of past points to be used as an input to the LSTM cell (forget gate parameter) is a finite parameter controlled by the backpropagation algorithm. On the other hand, the convolutional networks use a "window" based method,

where the input to the model is a fixed number of points, with the model giving a classification for each point in turn. This model results in a larger model (greater numbers of parameters to fit) but allows for a grouped processing of points, forcing the model to consider the entire window for each point it classifies. This window size forms a trade-off where more points exponentially increase model size, but gives more context to the model to classify each point. Figure 5.1 illustrates the difference in these two approaches, and Figures 5.2, 5.3 and 5.4 show model diagrams for the architectures used in this chapter.

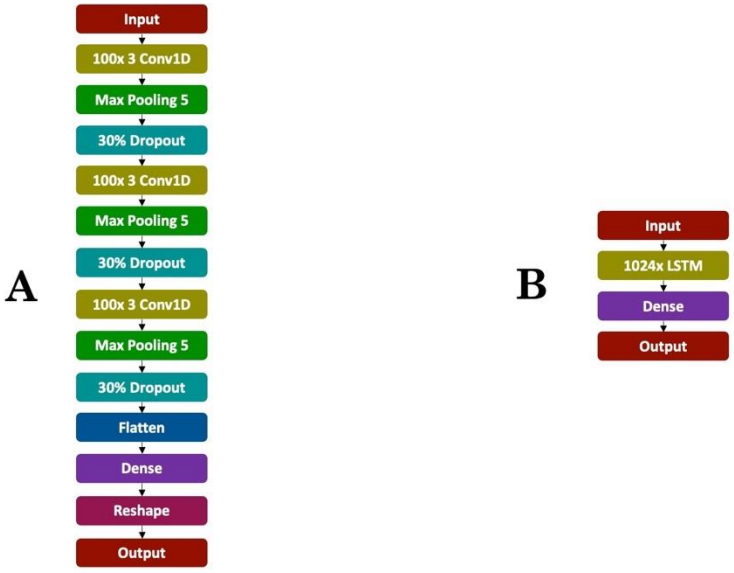


Figure 5.2: Model architectures for the "Simple CNN" (A) and "LSTM" (B) models.
These models are simple examples of neural networks of the recurrent and convolutional types. The "Simple_CNN" model has two layers of convolutions followed by a fully connected layer, with added dropouts and max pooling layers for performance. For the 3-state datasets this model has 11,777,872 parameters. The "LSTM" model has two layers of LSTM cells followed by a fully connected layer; this model has significantly fewer parameters than the CNN model (4,205,571 for the 3-state datasets), but predicts the points on a per-point basis rather than together in windows of 1024.

We then implement two models from problem specific literature; DeepChannel (Celik et al., 2020) is a hybrid recurrent-convolutional network that has been proven to successfully idealise real ion channel data, with greater accuracy than traditional algorithms in the scenarios used in the ground breaking paper (Celik et al 2020]. Like the simple recurrent model, it predicts the number of channels point by point; however, it made no attempt at recovering the underlying Markovian state. In this work, we retrain the model on the Markovian datasets to extend its functionality to Markovian recovery. The other model

(SplitCNN) from the literature we consider is DeepSleepNet (Supratak et al., 2017); a split-convolutional based model that has "coarse" and "fine" feature recognition through implementing a large kernel and small kernel for two different convolutional pathways. This model has been shown to be able to recover the underlying Markovian sleep stage state from EEG signals, so solves a similar problem to the one presented in this chapter. Figure 5.3 shows model diagrams for these implementations.

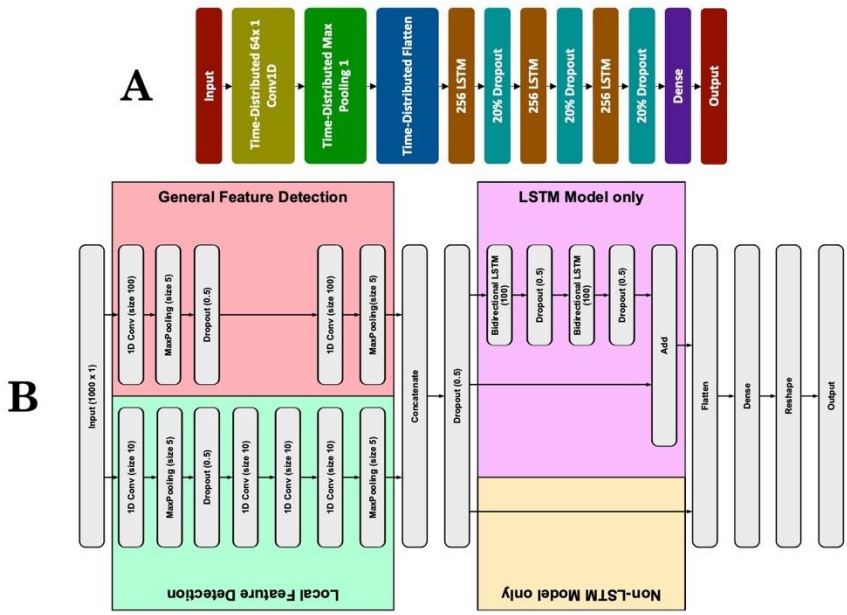


Figure 5.3: Model architecture diagrams for DeepChannel (A) and the SplitCNNModel (B).
Both of these models are adapted from the literature – DeepChannel (CITE) has shown state-of-the-art performance on ion channel idealisation, and the SplitCNN model approach has been seen to work for sleep stage detection in DeepSleepNet (CITE). Both models have been adapted for Markovian state recovery of simulated ion channel data. Similar to the "simpler" models, the RNN-based DeepChannel model has significantly fewer parameters (1,383,299 for the 3-state datasets) than the SplitCNN model (17,289,072), however again predicts point-by-point rather than in groups of 1024.

Finally, two models, ResNet and U-Net, are adapted from state of the art, highly sophisticated deep learning research on image analysis to time series analysis. ResNet (He et al., 2015) is an extremely deep neural network that utilises "skip" connections to avoid the effect of diminishing returns from additional layers. This model is typically used for holistic classification of images; taking a large two-dimensional image array and giving it a single class at the end; however I adapted this for use with one-dimensional time series arrays by including a large dense layer at the end for point-by-point classification. U-Net (Ronneberger et al., 2015) is a similar model that also uses skip connections, but is

usually used for image segmentation through a series of deconvolutional layers to recover the original image size, similarly to an auto-encoder. for the present work I adapted this by converting the 2 dimensional cells in the original schema to one dimensional cells; and as our problem is equivalent to segmentation (labelling each input point as a fixed number of classes), no further adaptations were needed. Figure 5.4 shows model diagrams for these architectures.

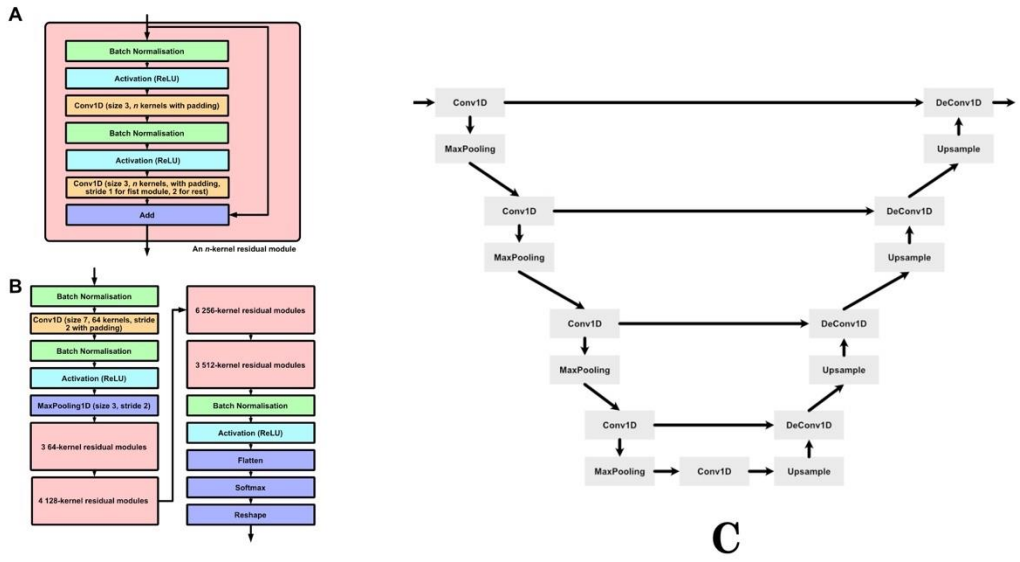


Figure 5.4: ResNet (A, B) and Unet (C) model architecture diagrams.
ResNet and UNet are state-of-the-art solutions for image classification and segmentation respectively; by converting the 2 dimensional neural cells into 1 dimensions, we can adapt them to be used with time series data. In ResNet's case, we have to adjust the output to be a classification for each point, rather than a classification for the whole timeseries (as in the typical implementation); but no such adjustment has to be made for UNet as our problem is equivalent to one-dimensional time series segmentation. The UNet model has 10,463,747 parameters for the 3 state datasets, and (due to the point-by-point classification adjustment) the ResNet parameter number is the largest of all the models at 57,570,756 for the 3 state datasets.

Figure 5.5 shows a comparison of parameter count for models training on the 3 state datasets; we can see the CNN models have vastly more parameters than their RNN counterparts; some caution should be taken into interpreting this as the models having poorer performance; as the CNN models can leverage GPU acceleration via CUDA acceleration as well as process files in batches in 1024 point windows, reducing the number of inputs needed to idealise a file.

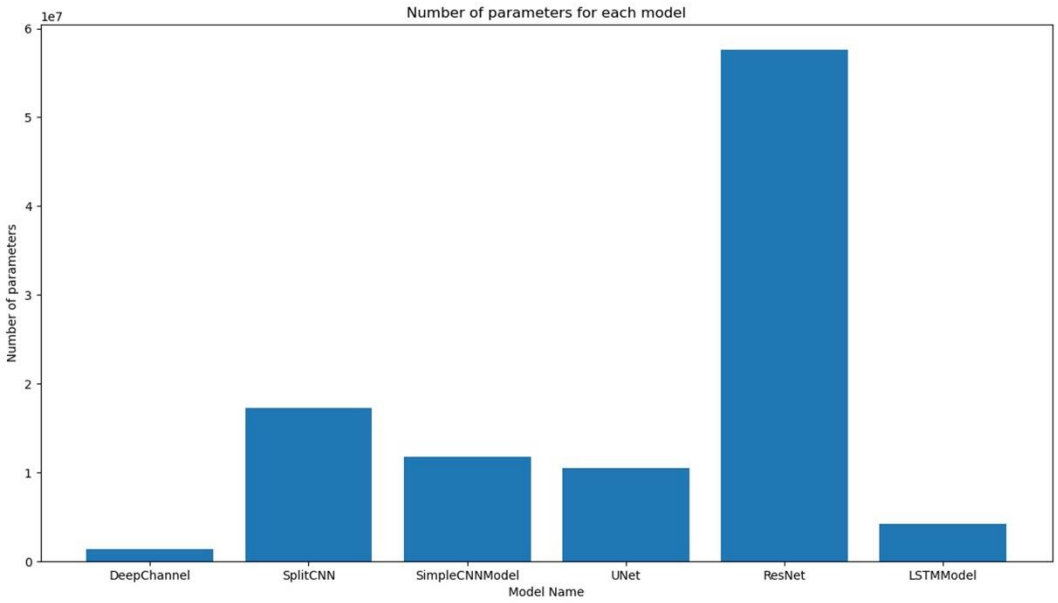


Figure 5.5: Comparison in model sizes by number of parameters for all models tested.
The number of parameters a model has positively affects its ability to learn abstract patterns, at the cost of performance and the risk of overfitting to data. Models such as ResNet utilize a significantly larger amount of parameters than the simpler recurrent models, due to the depth of their architectures. It's important to state that training time is not directly proportional to the number of parameters – models using CNN layers can utilize CUDA accelerated learning via the GPU, so train significantly faster. In addition, the CNN based models are predicting groups of 1024 points at once, meaning that their effective speed may be much faster.

5.2.2 Dataset Generation

Lab recorded data has no Markovian ground truth, so it is difficult to train a neural model as no loss function can be accurately calculated. We use a synthetic data pipeline (see Chapter 2.1.1) with two Markov configurations, and test for model's robustness against data with or without baseline drift, and with or without small perturbations of the Markovian model; resulting in a total of 8 different dataset classes for each combination above. For each combination below, 48 files were generated for training, 48 for validation, with 24 for testing, with each file having 10 minutes of data at 10kHz, as would be seen

in a realistic ion channel recording. A diagram with the dataset generation can be seen in Figure 5.6.

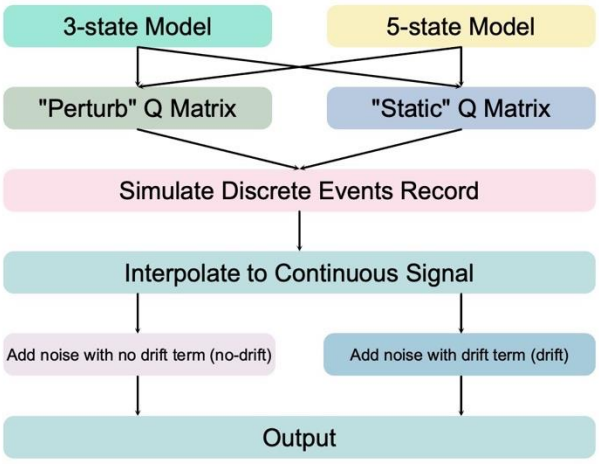


Figure 5.6: Data Generation Protocol.
For the Markovian datasets, 8 families of data were generated, each with 48 files of validation, 48 files of training, and 24 files of testing, each at 10mins of data at 10kHz each. The 8 dataset families come from the combinations of the different choices colored in the figure; first, either the 3-state or 5-state model is chosen (see Figure 5.7), and either sampled with fix parameters, or sampled with perturbing parameters (see methods) to obtain a sequence of states and dwell times that in total exceed 10mins. Finally the data is interpolated to a signal assuming a sample rate of 10kHz, using either scaled 1/f^n noise (with parameters chosen to qualitatively match lab recorded data), or this noise in addition to a "drift term"; another 1/f^n noise with a larger n, representing a slow but random change of the current over time.

5.2.2.1 Markovian Models

The two Markovian models used for data generation can be seen in Figure 5.7. The "3 state" model is the simplest non-trivial Markovian model to test for state recovery. We see a "short open" and a "long open" that are significantly different, so a successful model should be able to discern the difference between the events by considering the length of each of these events.

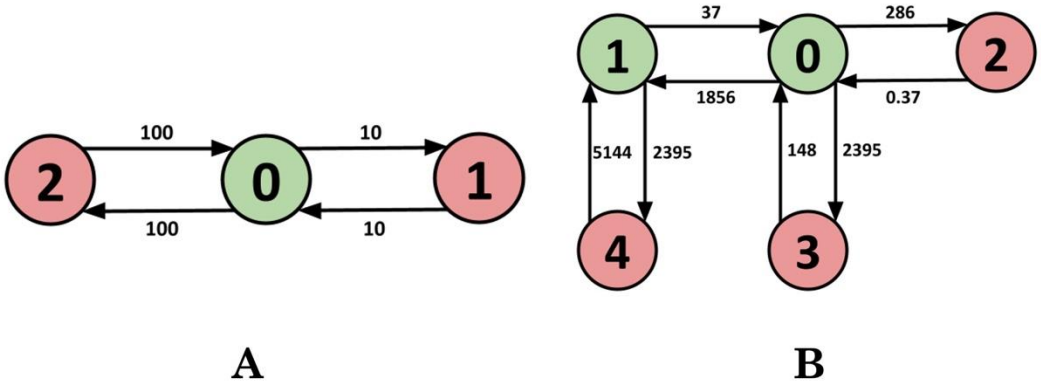


Figure 5.7: "3-state" (A) and "5-state" (B) Markovian model diagrams used for two different types of data generation for training.
To discern if our neural networks can detect the underlying Markovian state, we use two Markovian schema; the simplest possible Markovian model that still exhibits hidden states (A), and a rate-fitted Markovian model taken from Davies et al. (B). In total we generate 8 datasets, half of which are sampled from the 3-state model and half from the 5-state model. The motivation behind this is to test models' performance on both theoretical, "best case" Markovian processes versus what is more normally seen in the lab.

The "5-state" model comes from a Markovian model experimentally fitted from real ion channel data (Davies et al., 2010). It is significantly harder than the 3-state model for a number of reasons: the additional number of states naturally introduces a higher error rate, but the lack of a visible state change between the open states results in some confusion as to the hidden state the simulation currently holds. A successful neural model should still be able to infer that a particularly short-closed event (C3) is extremely likely to have two open events succeeding it if the next closed event is particularly long (C1). Our initial approach to this problem was to use an uncoupled canonical form such as BKU; however as we concluded in Chapter 4, this often leads to models that are invalid in "physiological space". Datasets are labelled with their Markovian model; either "3-state" for the 3 state Markovian model or "5-state" for the 5 state model.

5.2.2.2 Transition Rate Matrix Perturbation

For each Markov model, we consider two cases; one where the Markov model is static across all datafiles and one where we perturb the transition rate matrix to simulate a drug effecting the dynamics of the ion channel. This perturbation must be done carefully; if we change the structure of the Markovian model too much then we encounter some confusion with Markov model equivalencies, whereas not adjusting the transition rate matrix enough will not test for drug effects at all. We therefore implement an algorithm where a randomly chosen multiplier for each rate matrix entry is chosen from a triangular distribution (from 0.75x to 1.25x), and multiplied with the transition rate matrix for each file. This preserves the Markov model structure (no new connections across states are made), and proportionally changes each of the dwell times stochastically to maintain the order of magnitude. Although this might not exactly model drug action (where typically only a few rate entries are changed), this method tests for additional robustness due to recorded variances in kinetics. Datasets with perturbed transition rate matrices are labelled "perturbed", and those without, "static"

5.2.2.3 Noise and Baseline Drift

The noise seen in ion channels is complex and depends on the channel, conditions and equipment used. Therefore, we test for each dataset across a range of situations, along with the presence or absence of a baseline drift term. This long-term noise element proves difficult for other, traditional algorithms to overcome, yet is often present in real ion channel data. Overcoming this type of noise and achieving a recovery of Markovian state despite it is vital for real world application; as such, for all files we randomly add a channel scaled $1/f^n$ noise with n chosen randomly between 0.8 and 1.4, signal to noise ratio between 2 and 5, and channel scaling factor between 1.1 and 1.5. For the drift datasets, we additionally add a $1/f^n$ with n between 3.5 and 4.5 and signal to noise ratio above 5 (see Methods chapter 2.2 for how this affects the output signal). These values were chosen by inspecting lab recorded data and comparing the outputs of the simulations qualitatively.

An example of both non-drift and drift data samples can be seen in Figure 5.8. In total these procedures produced 8 different simulated ion channel record "phenotypes". Datasets are labelled "no drift" for those without the additional drift term added, and "drift" for those which have.

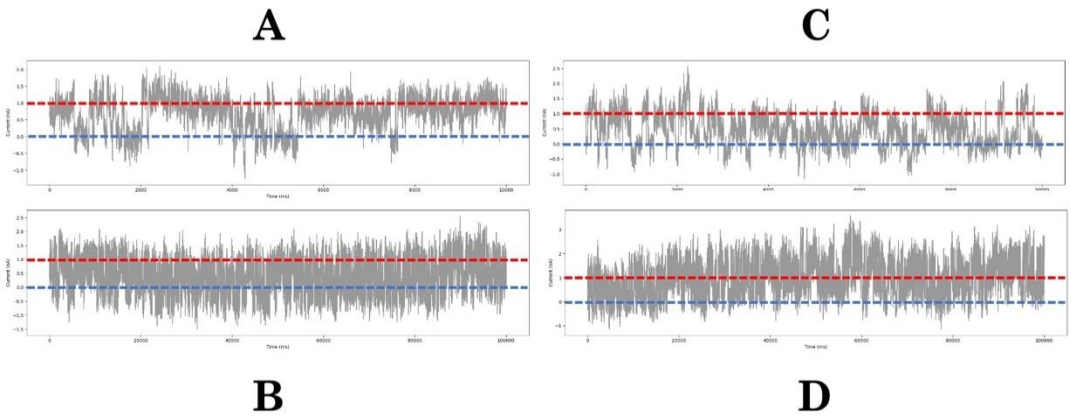


Figure 5.8: Training data examples for non-drift (A,B) and drift (C,D) datasets.
For half of the dataset families, a "drift" noise term was added to simulate the long term baseline drift experienced when recording real data in a lab. The non-drift dataset (A, B) stays mainly between 0 and 1.5pA; with some variation due to the random nature of the noise. The drift datasets (C, D) on the other hand have a "noise" term added causing a slower, but larger variation of the current over time, as is typically seen in lab recordings. A and C show these respective datasets over a single second of simulation, whereas B and D show the larger trend over 10 seconds of data. Dotted blue and red lines show the default closed and open conductance levels respectively.

5.2.3 Training Protocol

For training, we employed two Keras callback functions to dictate when training ended; if the model's performance against the validation set did not improve within 5 epochs, the learning rate was reduced by a factor of 10, and if the performance still did not improve within 20 epochs, training stopped entirely. After training, each model was then evaluated against each file in the testing set, where accuracy, F1 Score and Cohen's Kappa Score were all recorded for both the state recovery and open/closed idealisation. In each case we present results from both the full Markovian recovery and the so called "reduced" recovery which is simply detection of open and closed, essentially idealisation.

5.3 Results

After training all 6 neural networks on the 8 dataset phenotypes, we recorded the micro F1Score and Cohen's Kappa score for the model's prediction versus the ground truth for each point in the testing dataset. Figures 5.9 to 5.32 show the performance metrics for each model.

5.3.1.1 Static 3-State Model Without Drift

This dataset represents what one would expect to be the easiest for a model to decode. Figure 5.9 shows an example of raw data with the recovered idealization below, with both model prediction and ground truth for each of the 6 models (LSTM, DeepChannel, SimpleCNN, SplitCNN, ResNet and U-Net). Visibly, there is some success although areas of failure, furthermore the LSTM and DeepChannel data look similar, but different to all the other 4 models that in fact look similar to each other. This is supported by the quantitative data; Figure 5.10 shows Cohen's Kappa validation results for each model applied to these data with 24 replicates. Figure 5.10 is the reduced result representing full Markovian recovery. The most successful models are the Simple and Split CNN models, achieving respective Kappa scores of 0.8164 ± 0.0725 and 0.8475 ± 0.0735 (n=24). These perform better than the very large ResNet and UNet models (0.7708 ± 0.1962, 0.7951 ± 0.0301, n=24 respectively) in this scenario and statistically significantly better than the original, published, DeepChannel model or a simple LSTM recurrent model (0.3850 ± 0.0652, 0.3877 ± 0.0618, n=24 respectively).

In terms of simply open/closed idealization ("reduced" Kappa), the pattern is generally similar, but DeepChannel and LSTM models do perform more closely (0.7266 ± 0.1556, 0.7418 ± 0.1467, n=24 respectively) to the Simple and SplitCNN models (0.9475 ± 0.0461, 0.9538 ± 0.0510, n=24 respectively). The fact that all these models have Cohen's Kappa significantly greater than zero indicates that they do all provide highly significant idealization and recovery of Markovian state beyond chance. Figures 5.11 show the same

data but analyzed with micro F1. The results parallel those of Cohen's Kappa, but note that with the simple idealization ("reduced F1") all models achieve micro F1 of over 90%.

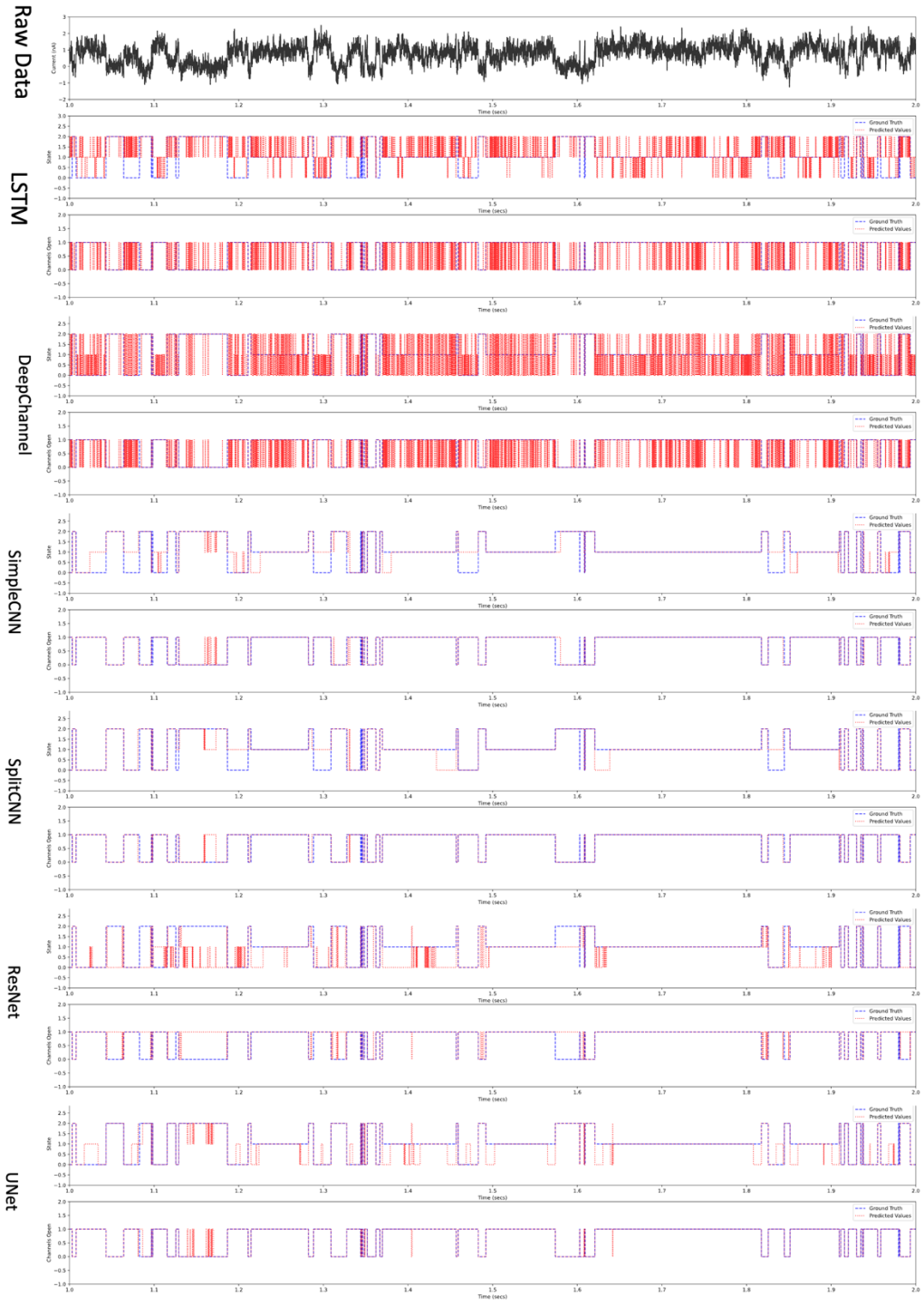


Figure 5.9: Sample Model Traces for "Static Three State Model with No Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. We see LSTM and DeepChannel models experience low performance and a high amount of "flickering" between states, whereas the SimpleCNN, SplitCNN, ResNet and UNet models are all fairly accurate with their predictions.

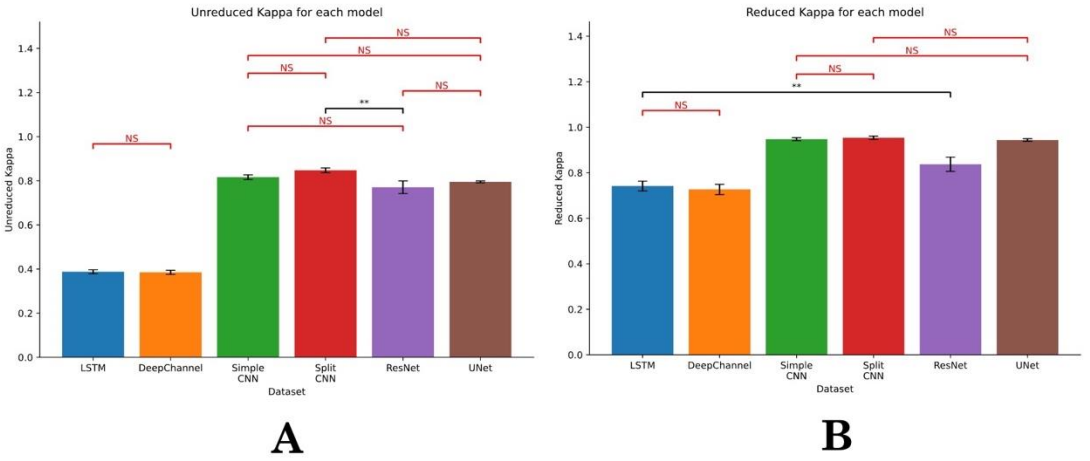


Figure 5.10: Model Training Metrics for "Static Three State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).

Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

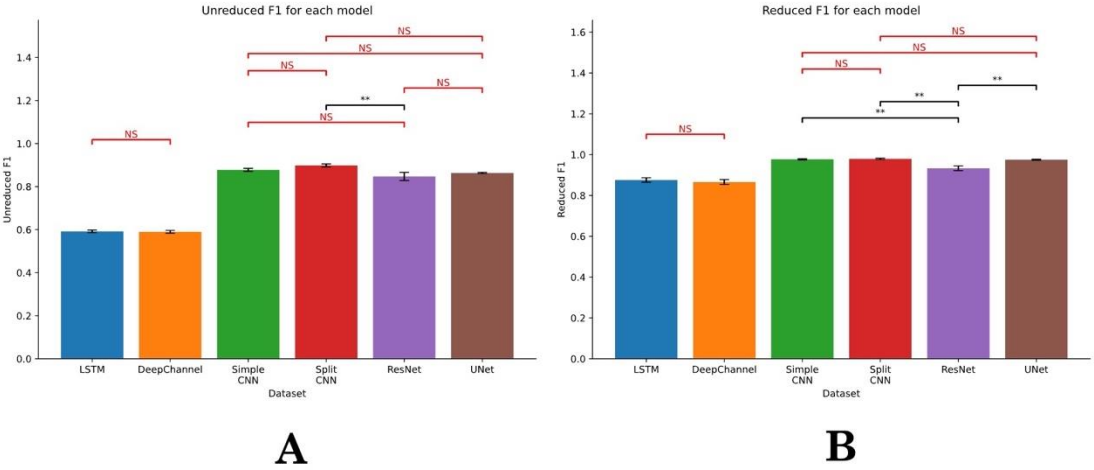


Figure 5.11: Model Training Metrics for "Static Three State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).

Two way ANOVA showed significant differences of the models for micro F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.2 Static Three State Model With Drift.

With this dataset phenotype there is the addition of a drift noise term. Figure 5.12 shows an example of raw data with the recovered idealization below, with both model prediction and ground truth for each of the 6 models. As with the no-drift data above, the raw data the LSTM and DeepChannel look very similar and this is matched by the metrics. Figure 5.13 show the Kappa scores for the full Markovian and simple idealization respectively. On the full Markovian ("unreduced") predictions the SimpleCNN is the strongest (0.7883 ± 0.0467, n=24) whereas DeepChannel and the LSTM perform worse (0.3547 ± 0.0713, 0.3640 ± 0.0672, n=24 respectively), (with the exception of ResNet), although still much better than chance. Strikingly, ResNet has failed with a Kappa score of 0. This is apparent from the raw data (Figure 5.10A) and the F1/Kappa metrics. For the simple reduced idealization, recovery simply of open or closed state the margin is smaller, but the overall pattern of performance is similar. The micro F1 scores, presented in Figure 5.14 are analogous, but note that ResNet now performs very poorly.

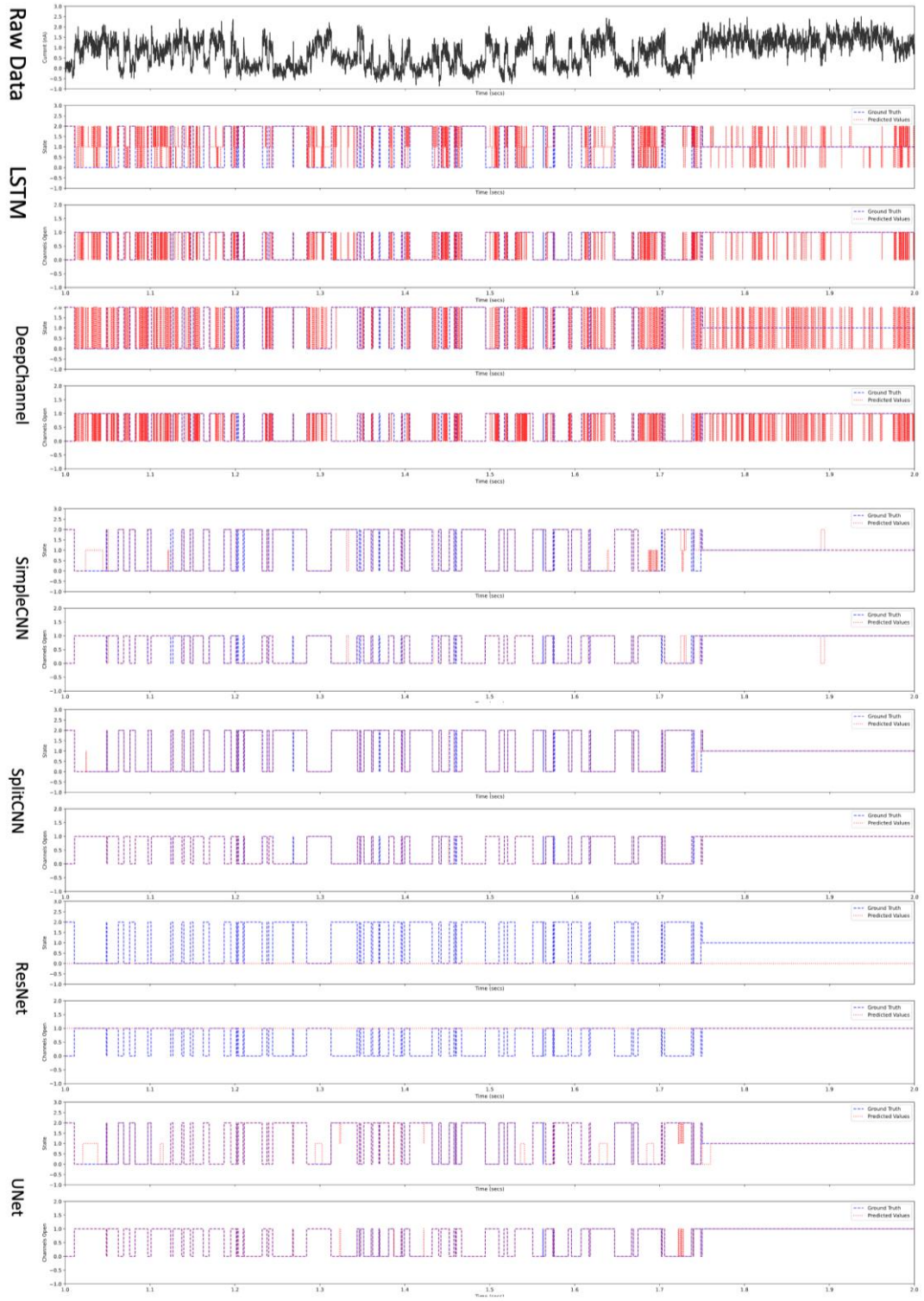


Figure 5.12: Sample Model Traces for "Static Three State Model with Drift" Dataset
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. As with the non-drift dataset, we see LSTM and DeepChannel models experience low performance and a high amount of "flickering" between states, whereas the SimpleCNN, SplitCNN, ResNet and UNet models are all fairly accurate with their predictions.

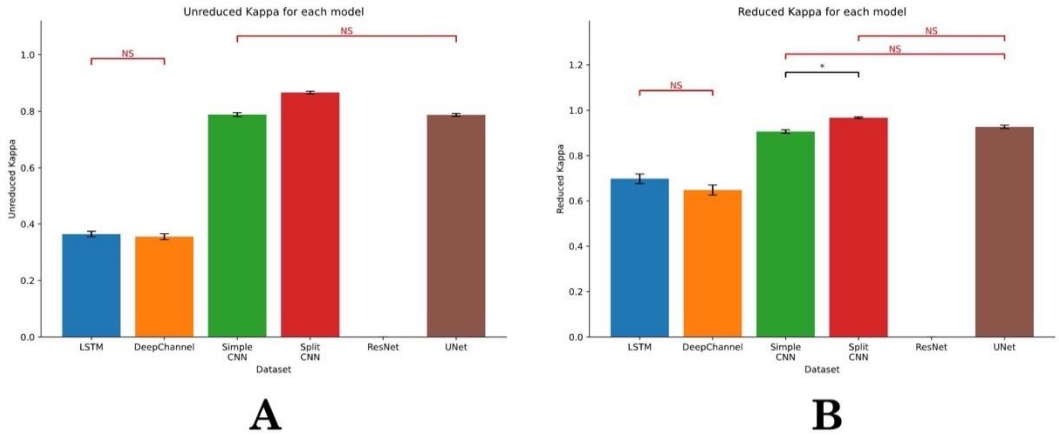


Figure 5.13: Model Training Metrics for "Static Three State with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

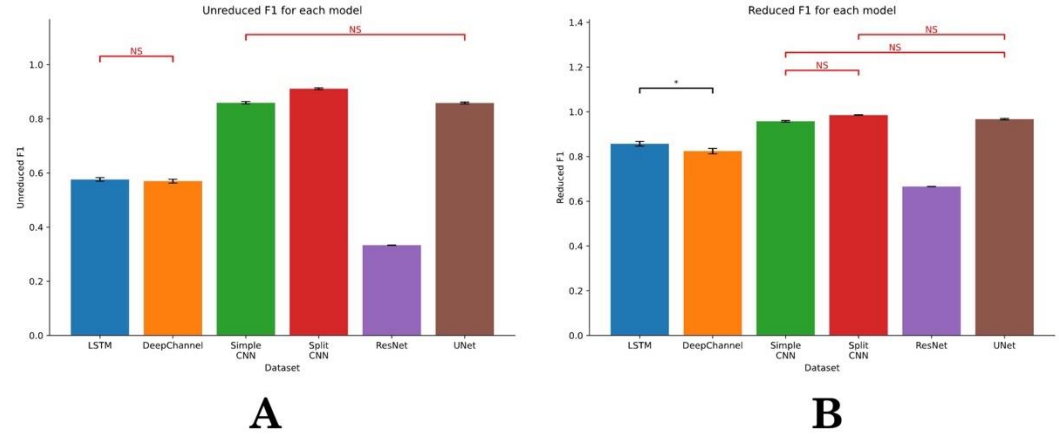


Figure 5.14: Model Training Metrics for "Static Three State with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.3 Perturbed Three State Model Without Drift.

The underlying Markovian scheme here is the same as above, however prior to dwell times being sampled from the transition rate matrix, the values were perturbed as described in Chapter 5.3.1.2 to simulate drug action. Raw data are shown in Figure 5.15 and it is immediately apparent that there has been, again, a total failure of ResNet. Amongst the others, performances look subjectively similar to the static datasets. From the Kappa and micro F1metrics (Figures 5.16, 5.17) the performances parallel that of the static data, with the best model being the SplitCNN returning a Kappa score of 0.8659 ± 0.0352 for full Markovian recovery and micro F1 of 0.9106 ± 0.0235. In terms of Cohen's Kappa the Simple and SplitCNN perform twice as well as the simple LSTM or DeepChannel.

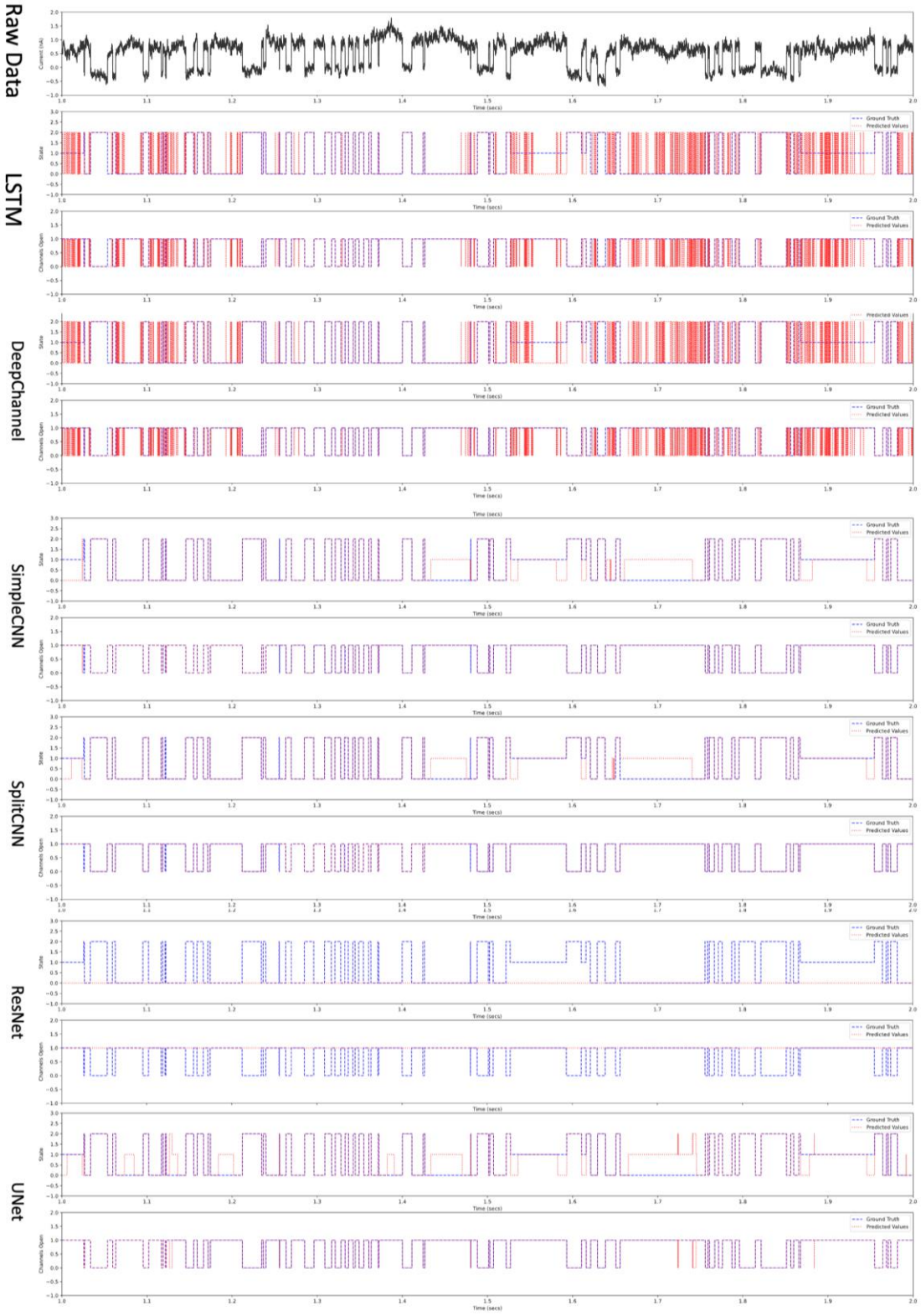


Figure 5.15: Sample Model Traces for "Perturbed Three State Model with Drift" Dataset
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. For this sample of data we see a less noisy sample, which subjectively improves the performance of the recurrent based models (LSTM and DeepChannel). We still see however, that the convolutional based networks show far stronger performance compared to the RNN counterparts.

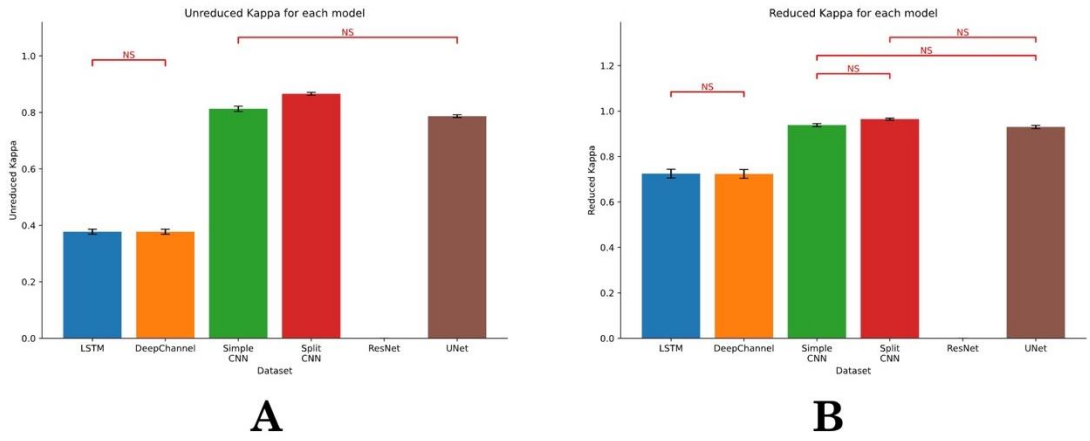


Figure 5.16: Model Training Metrics for "Perturbed Three State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

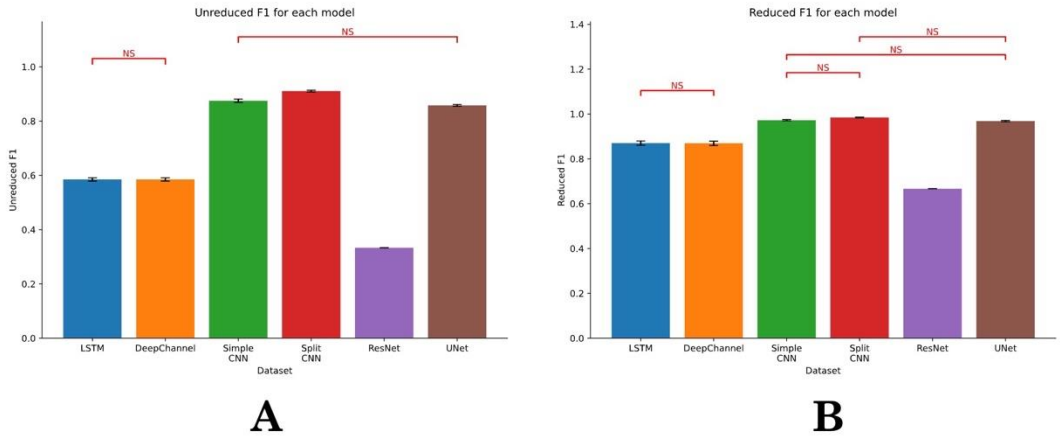


Figure 5.17: Model Training Metrics for "Perturbed Three State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.4 Perturbed Three State Model with Drift.

These datasets were similar to the previous set, but with the addition of drift as described earlier. This would be expected to be the most challenging for the 6 models to recover information from so far. Raw data with Markov recovery and idealizations are shown in Figure 5.18. Again, subjectively LSTM and DeepChannel look very similar to each other and the other four models looking distinct and reasonably successful. The metrics are shown in Figures 5.19, 5.20. ResNet fails to exhibit statistically significant (for Kappa score 0; corresponding to randomly guessing the state) recovery either of full Markovian state or "reduced" open/closed state idealization. SimpleCNN, SplitCNN and UNet all deliver good performance with Kappa scores of 0.8248 ± 0.0477, 0.8669 ± 0.0366 and 0.7971 ± 0.0314 respectively, n=24.

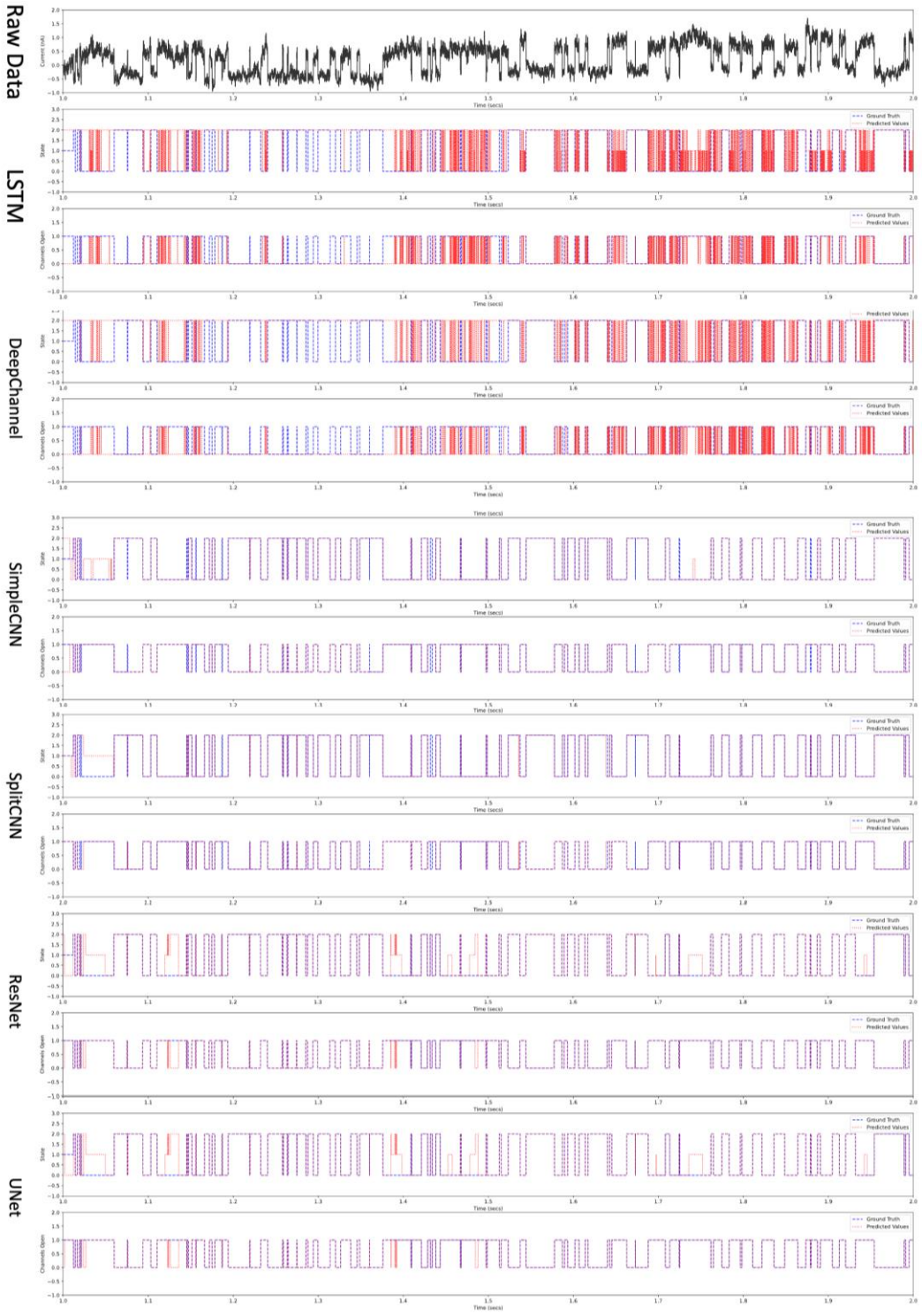


Figure 5.18: Model Training Metrics for "Perturbed Three State Model with Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. For this sample of data we see a less noisy sample, which subjectively improves the performance of the recurrent based models (LSTM and DeepChannel). We still see however, that the convolutional based networks show far stronger performance compared to the RNN counterparts.

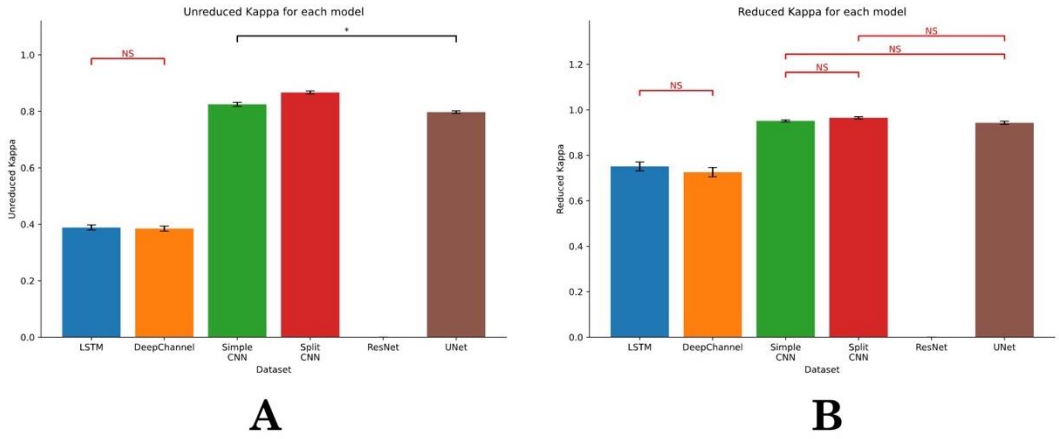


Figure 5.19: Model Training Metrics for "Perturbed Three State Model with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

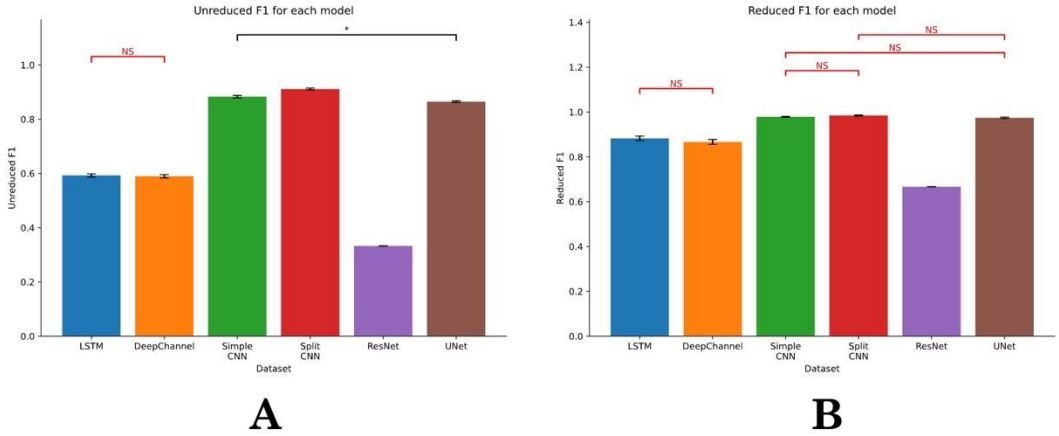


Figure 5.20: Model Training Metrics for "Perturbed Three State Model with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.5 Static Five State Model Without Drift

The subjective appearance of these datasets is very different from the 3-state model. The nature of the originating 5-state model leaves to long closed sojourns with bursts of openings (top panels of Figures 5.21, 5.24, 5.27, 5.30). With the static 5-state data, the raw representative example (Figure 5.21) shows a distinct burst in the centre of the window, subjectively, this has been detected by U-Net, SimpleCNN, SplitCNN and not ResNet. LSTM and DeepChannel, which appear to be influenced by the events but do not (subjectively) appear to have detected them convincingly. The two different metrics shown; Cohen's Kappa (Figure 5.22) and micro F1 (Figure 5.23) show report rather different values. Kappa for U-Net is near perfect at 0.9665 ± 0.0155 for the full Markovian ("unreduced") recovery and the simple reduced idealization. Whereas ResNet fails entirely by this metric and the other models are moderately successful. However, by assessment with F1, all models appear reasonably successful. This may be due to the severe imbalancing of the dataset in the five state model's data.

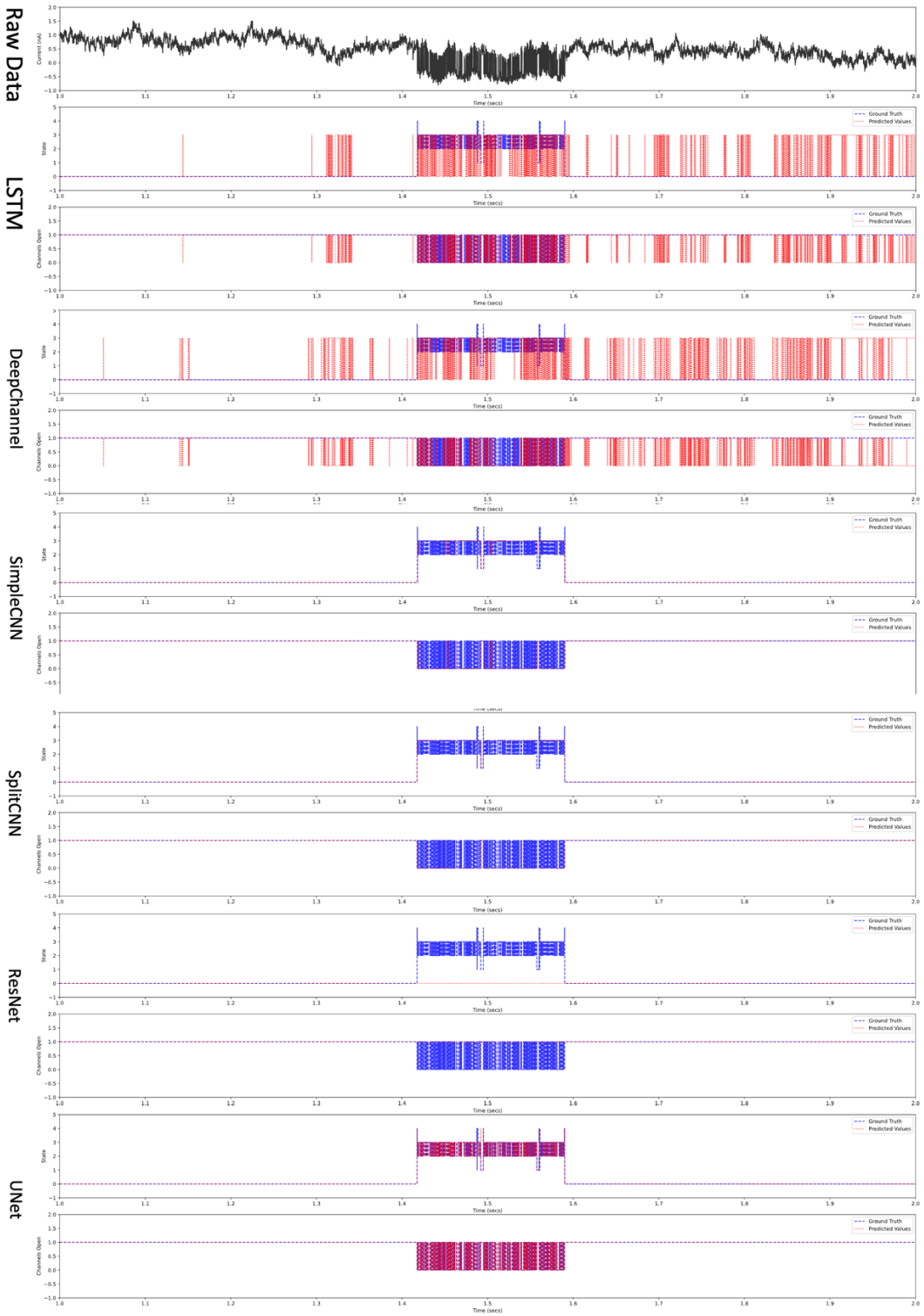
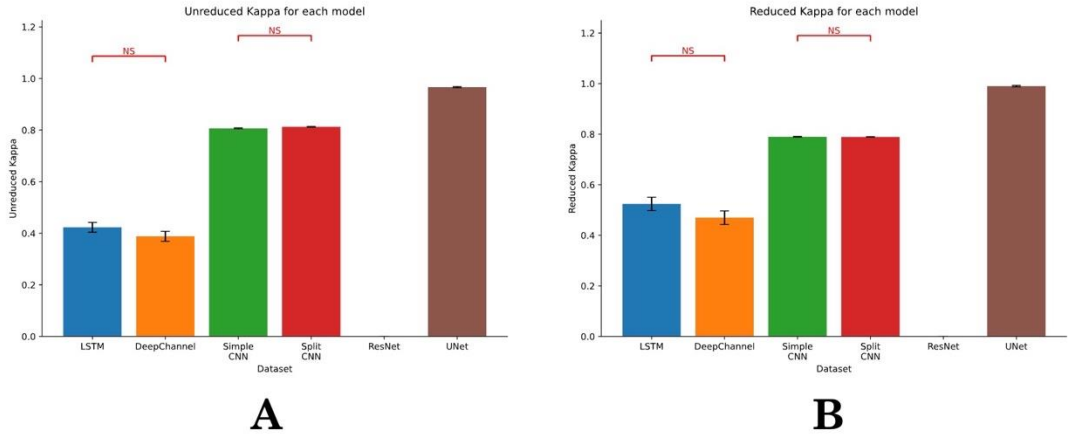


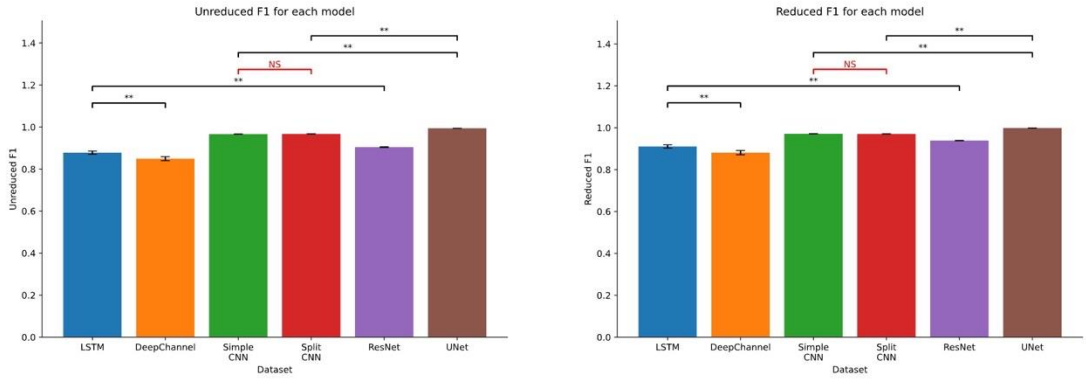
Figure 5.21: Model Training Metrics for "Static Five State Model with No Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. The Five state models exhibit periods of flickering that (subjectively) the recurrent networks have difficulty analyzing. The convolutional networks appear to have a fairly high accuracy, with the exception of ResNet that predicts the same state across the whole file.



A **B**

Figure 5.22: Model Training Metrics for "Static Five State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).

Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*



A **B**

Figure 5.23: Model Training Metrics for "Static Five State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).

Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.6 Static Five State Model With Drift

These datasets use the same 5-state model as above and the same noise levels, but add a realistic level of drift. Representative raw data are shown in Figure 5.24 although over this length of record, the drift is not obvious. Subjective inspection of these raw data traces suggest that only the Simple and Split CNN models have been successful. This is reflected quantitatively in the Cohen's Kappa metric graphs in Figure 5.25. There is a modal collapse of the larger U-Net and ResNet models (no statistically significant state recovery or idealisations compared to random guessing), good success with simple and splitCNN (Kappas of 0.8243 ± 0.0100 and 0.8168 ± 0.0088, n=24 respectively) on both full Markovian recovery and reduced open closed state detection. The commonly used micro F1 metric for all models is similar and approximately 0.9.

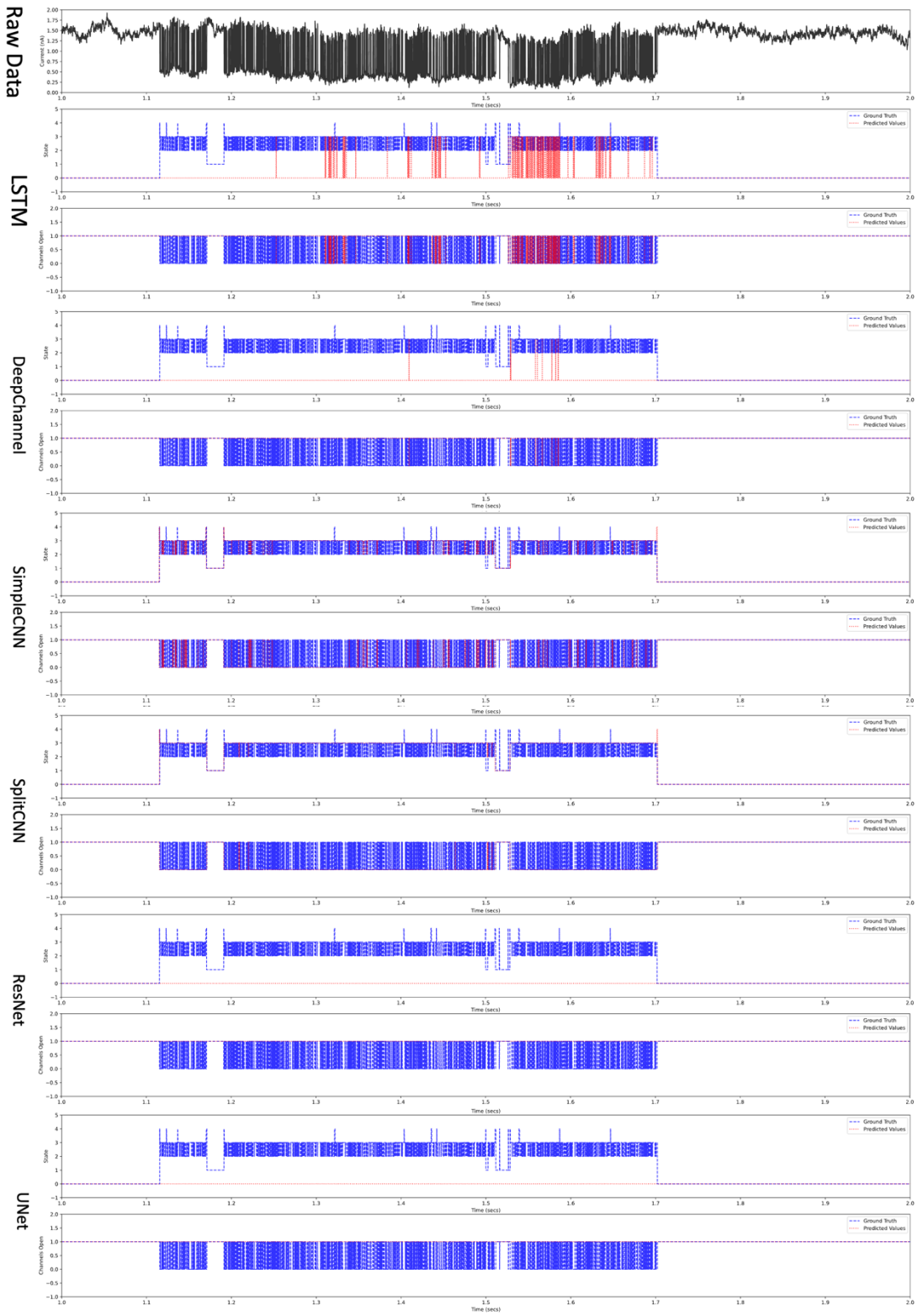


Figure 5.24: Model Training Metrics for "Static Five State Model with Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. The Five state models exhibit periods of flickering that (subjectively) the recurrent networks have difficulty analyzing. The convolutional networks appear to have a fairly high accuracy, with the exception of ResNet that predicts the same state across the whole file.

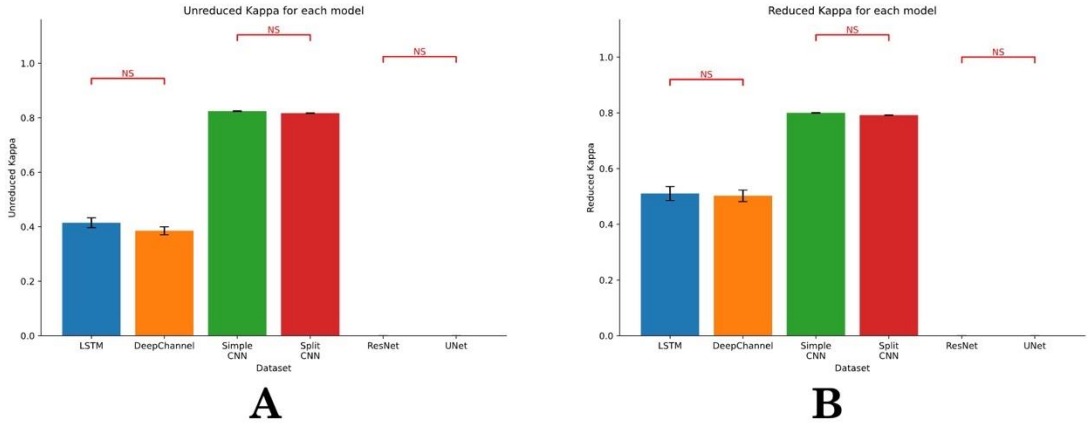


Figure 5.25: Model Training Metrics for "Static Five State Model with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

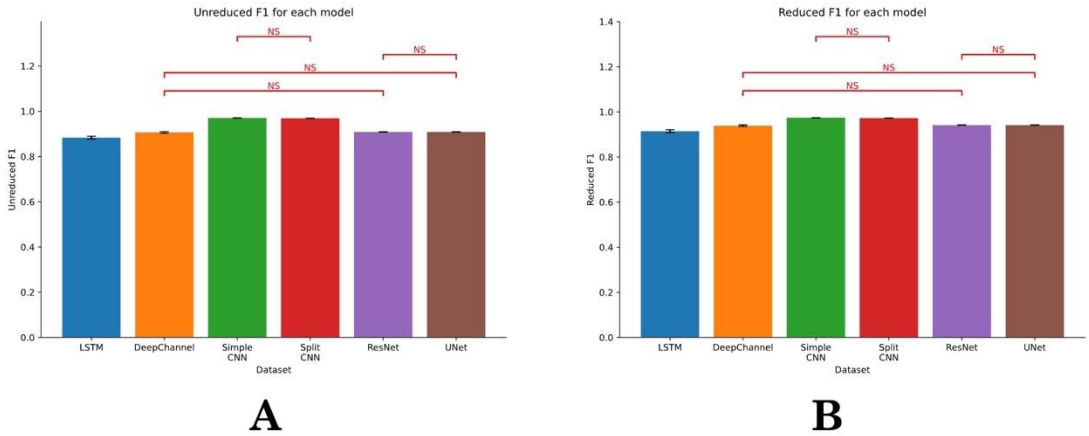


Figure 5.26: Model Training Metrics for "Static Five State Model with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.7 Perturbed Five State Model Without Drift

The results for the penultimate dataset (Figure 5.27), across the 6 models tested were very similar to that for the static 5-state with no drift. The U-Net model has proven the most effective by assessment with Cohen's Kappa (Figure 5.28) with similar performance from Simple and SplitCNNs (Kappas of 0.9658 ± 0.0173, 0.8297 ± 0.0123 and 0.8425 ± 0.0173. respectively, n=24). Again, ResNet fails entirely. It appears that perturbation of transition rate parameters has not substantially altered overall model effectiveness.

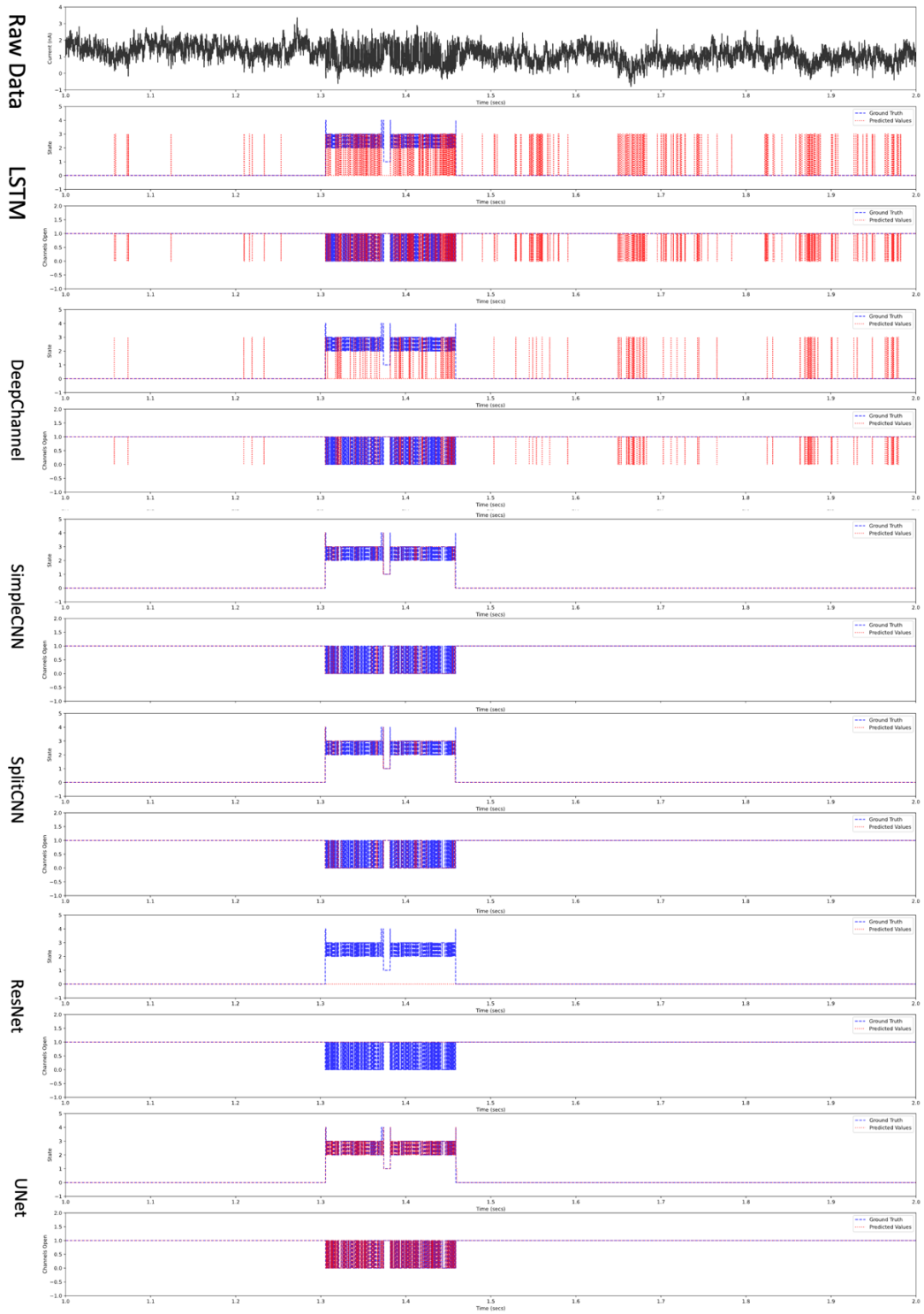


Figure 5.27: Sample Data traces for "Perturbed Five State Model with No Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. The Five state models exhibit periods of flickering that (subjectively) the recurrent networks have difficulty analyzing. The convolutional networks appear to have a fairly high accuracy, with the exception of ResNet that predicts the same state across the whole file.

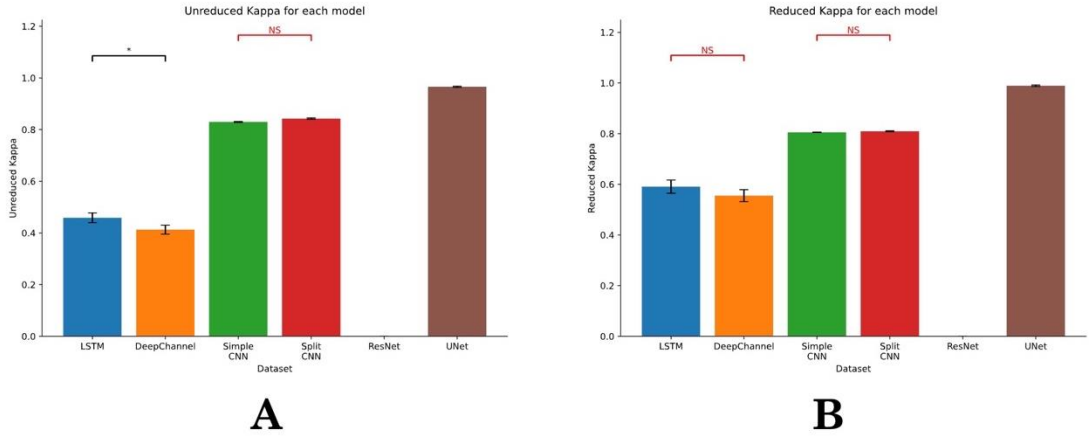


Figure 5.28: Model Training Metrics for "Perturbed Five State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

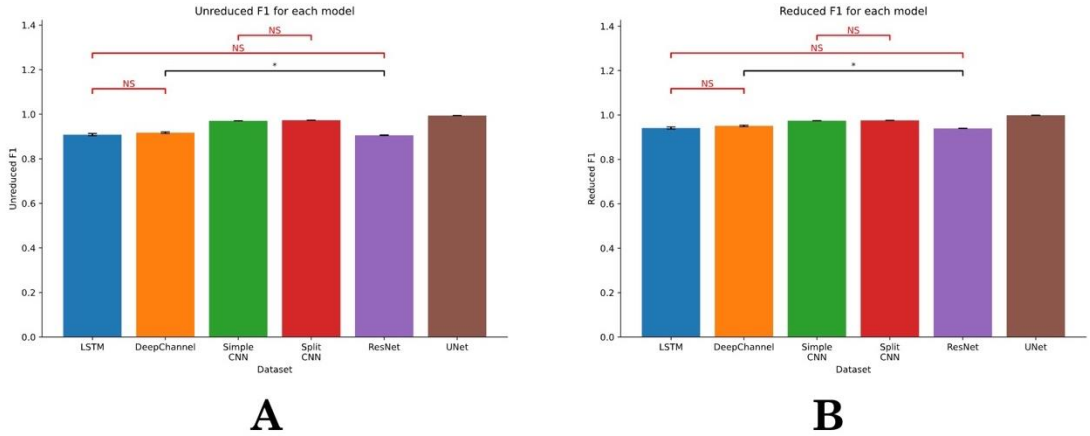


Figure 5.29: Model Training Metrics for "Perturbed Five State Model with No Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.8 Perturbed Five State Model With Drift

This dataset now adds drift to the above scenario and so this might be expected to be the most challenging of all the datasets tested our 6 models with (Figure 5.30). Subjective inspection of the representative raw data in Figure 5.30 suggests that U-Net has done the best job with Simple and Split CNNs also performing well. With the exception of the poorly performing DeepChannel model the micro F1 scores (Figure 5.32) are again all very similar and in the 0.9 range, but the Cohen's Kappa (Figure 5.31) reveal U-Net to be the best performing model with values of 0.9707 ± 0.0105 and 0.9946 ± 0.0119 for the full Markovian recovery and simple idealization respectively. It should be noted that SimpleCNN and SplitCNN also performed well; achieving 0.8197 ± 0.0100 and 0.8386 ± 0.0100 respectively for Markovian state recovery.

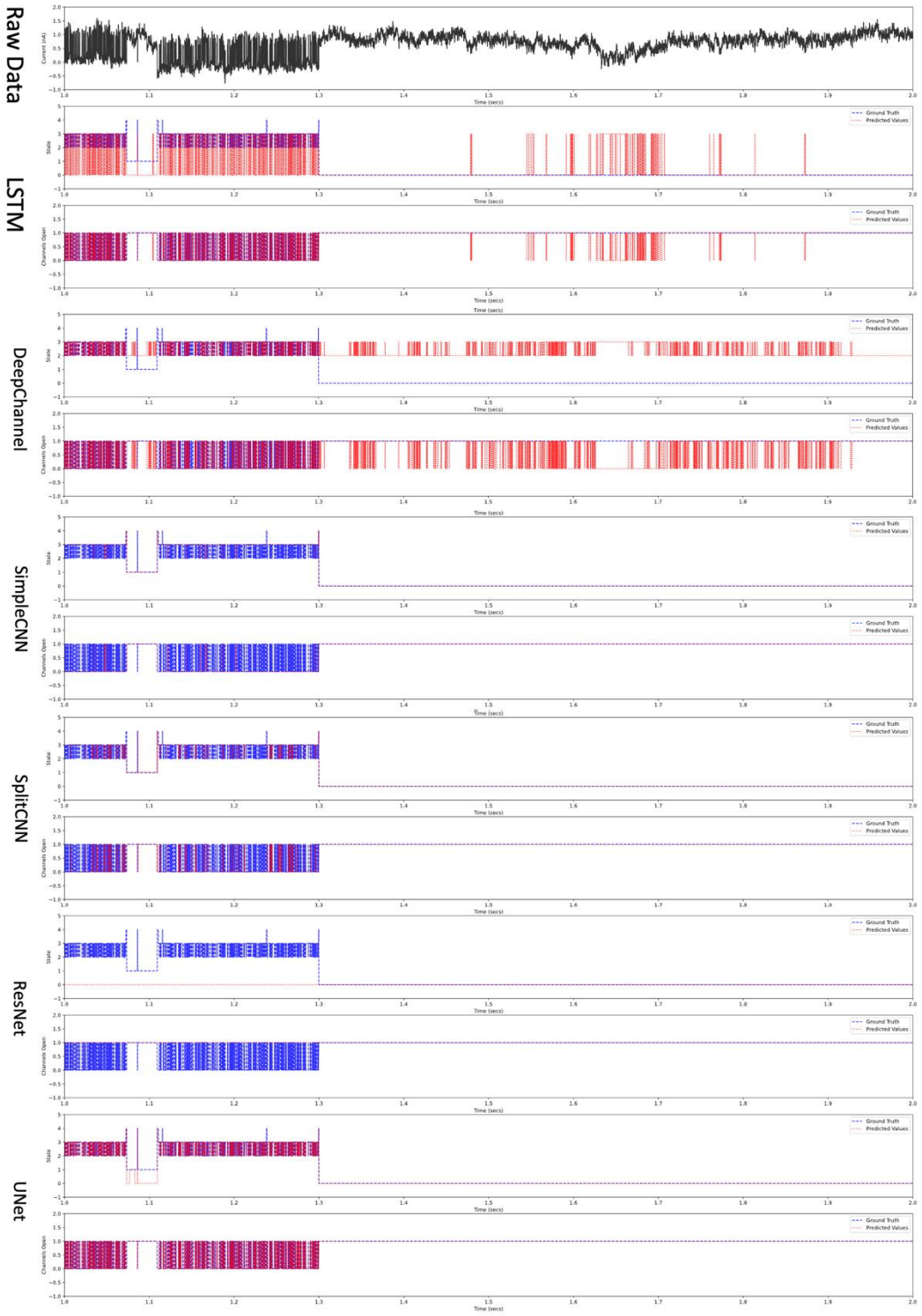


Figure 5.30: Sample Data Traces for "Perturbed Five State Model with Drift" Dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. The Five state models exhibit periods of flickering that (subjectively) the recurrent networks have difficulty analyzing. The convolutional networks appear to have a fairly high accuracy, with the exception of ResNet that predicts the same state across the whole file.

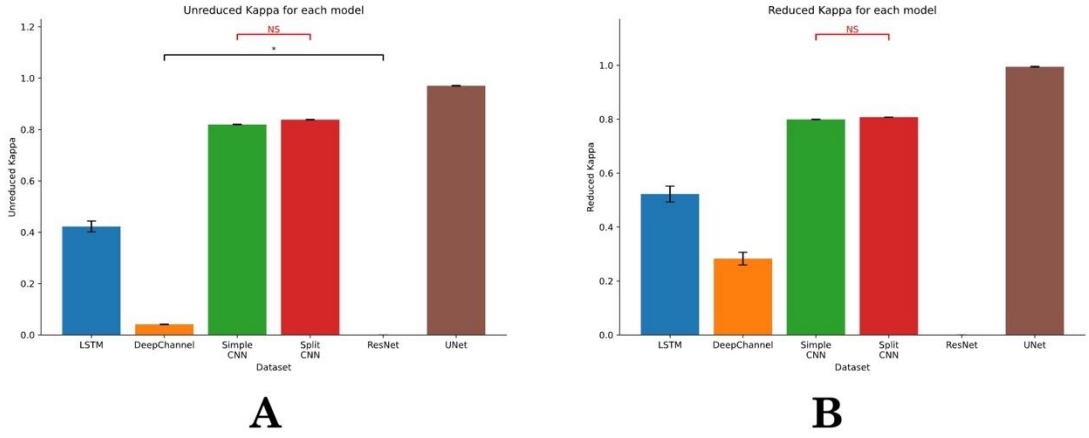


Figure 5.31: Model Training Metrics for "Perturbed Five State Model with Drift" Dataset for both Markovian recovery (A) and channel recovery (B).
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

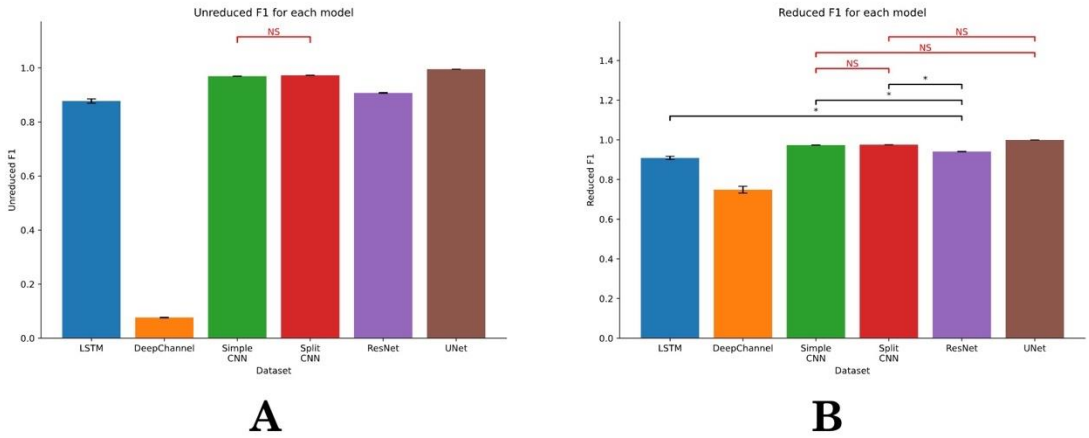


Figure 5.32: Model Training Metrics for "Five State, Hard, Drift" Dataset for both Markovian recovery (A) and channel recovery (B)
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

5.3.9 Training Progress

During training, the training and validation metrics were recorded to examine how quickly each model trained; Figures 5.33, 5.34 show the model training progress for two datasets across all models. Finally, as discussed before, the number of parameters in each of the models is vastly different; so training times were also monitored to compare training times for the GPU enabled CNN networks against the non-GPU enabled RNN networks. As an example of the relative training efficiency, Figure 5.35 shows the total time for training for the static 3-state no drift dataset. This shows that, whilst performance was not ideal, DeepChannel was the fastest to train followed by ResNet and LSTM. The three generally best performing models, SimpleCNN, SplitCNN and U-Net took the longest time to train.

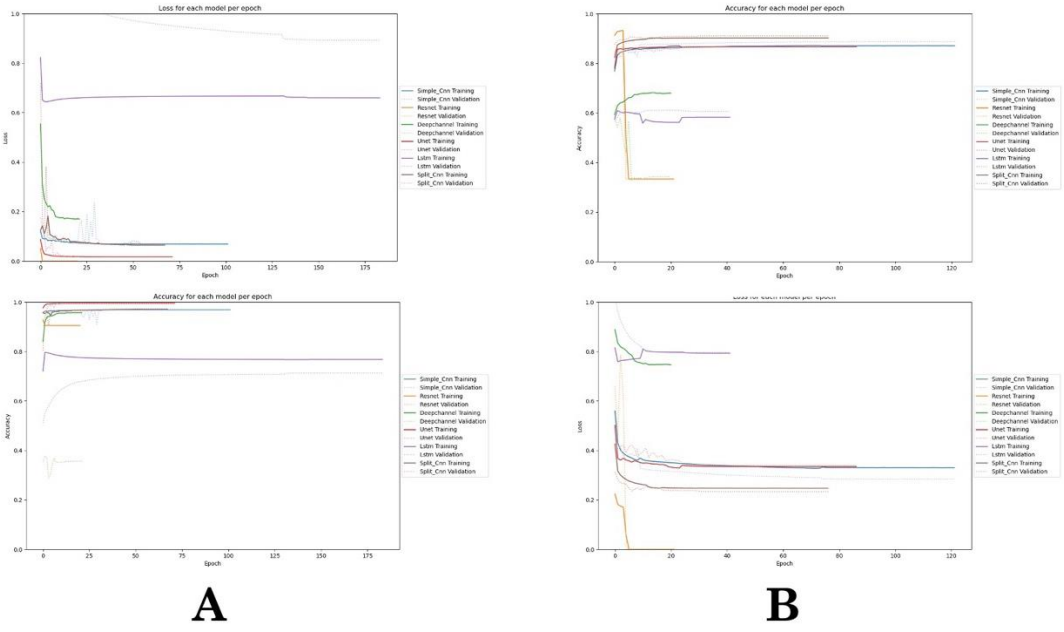


Figure 5.33: Model training process for each model for two different dataset families
A shows each models performance versus time for the Static Three State Without Drift dataset, and B for the Perturbed 5 State With Drift. Since the training protocol allows for models to continue training until they plateau, we see some models continue to get small marginal gains for quite a while before stopping. The Simple_CNN, Split_CNN and Unet models seem to have the most stable training processes, whereas ResNet, LSTM and DeepChannel models have some difficulty on the harder datasets.

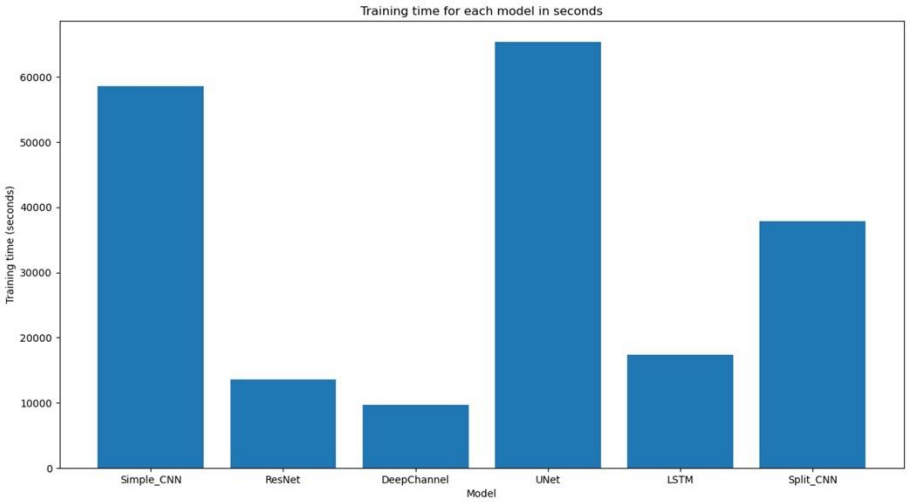


Figure 5.34: Model training times for each model in seconds for the Static Three State Model with No Drift dataset.
Here we can see with the exception of ResNet, that continued to make marginal gains for a number of epochs, the CNN models take the same order of magnitude of time to train as the RNN models, due to CUDA acceleration and the batch processing.

5.4 Discussion

This experiment was done as a method to see which model architectures had the most potential to take through to the rest of the work; it is clear from the results that the convolutional methods have greater accuracy and versatility over several different types of problems. However, we can instantly see a few problems from training.

5.4.1 Markovian Analysis

Firstly, since the data generation is stochastic, manual examination of the outputs reveals that in numerous points, the ground truth is not the most likely state for the observed dwell time. This is natural to expect as the ground truth comes from a stochastic simulation; however it is unfair to judge a model against a random entity. Therefore, developing a method to test the model against most likely state, rather than the simulated state is highly desirable.

In addition, the Markovian models used themselves in the data generation are another form of error. The "simple", 3-state model, with transition rate matrix created without reference to the literature creates arguably more stable, realistic data than the 5-state model taken from literature. One reason for this is that the data simulation pipeline is not perfect; for example the "flickering" seen in the 5-state output would likely be filtered out either from filtering, or from the mechanism of opening and closing not happening instantaneously.

5.4.2 Model Design and Training

Since the input window size of the convolutional models is what gives the model the context around a point to classify it accurately; we have a clear incentive to make this as large as possible; however, increasing this size also vastly increases the size of the final dense layer in all but the U-Net architectures (U-Net upsamples using deconvolutional layers). In fact, >90% of the models' parameters come from this final dense layer, which is a cause for concern as it becomes unclear whether the convolutional layers are causing good performance, or if the model is essentially just a fully connected model. It is important therefore to try and develop ways of reducing this final dense layer as it may be causing overfitting or is where the convolutional models are getting a performance edge over the recurrent models.

The training protocol gives all the models the "best shot" at predicting the testing dataset, but this results in some models training for a lot more time (both in terms of epochs and real-time) than others. For example, the simple LSTM model trained for at least 175 epochs whereas the DeepChannel model was finished before 25. As this synthetic data can be tuned to be closer to lab-recorded data, requiring retraining of the model each time; this increase in training time slows down model development, particularly in hyperparameter optimisation where training occurs many times.

A common observation, noted in various studies (Carvalho, Pereira, & Cardoso, 2019; Handelman et al., 2019) including our own here, is the contrasting perspective provided

by different performance metrics. For instance, the micro F1 score can still be high even if a model apparently completely fails, especially when there's a dominant class. On the other hand, Cohen's Kappa is generally more robust to this issue as it is more suited in imbalanced data. However, it might be beneficial to develop a custom metric that surpasses both in effectiveness, i.e., one that aligns more closely with an electrophysiologist's definition of success. For the subsequent Chapters we will therefore focus on the Cohen's Kappa rather than F1.

From this work the UNet, SimpleCNN and SplitCNN models are performing well above the LSTM, DeepChannel and ResNet models. As these models are of different levels of complexity, and with different design philosophies, we'll take only these top models through to further tests.

6 Improving Dataset Quality with the Viterbi Algorithm

6.1 Introduction

The process of developing a deep neural network requires labels to train against; these are true values for the neural network to compare to the output and adjust the weights depending on how closely the neural network output matches with the truthful output.

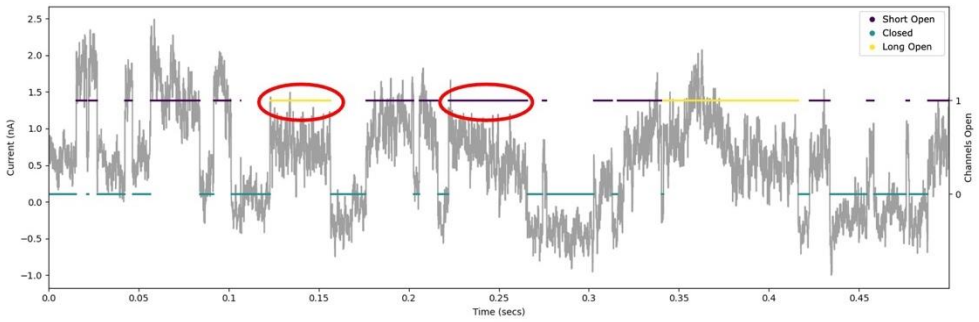


Figure 6.1: An example of Markovian "unluckiness".
Here an example of data where the Markovian simulation gives a state that is not the most likely outcome. Due to the probabilistic nature of the Markovian process, it is possible for an event to happen with a state that is not the most likely to occur – for example, in the figure above we see a particularly long "short" opening that would typically be expected from the second open state, rather than the first. When training a neural network, this stochastic nature creates confusion in training – we cannot reasonably expect the models to predict the output of a stochastic process, so should investigate a way to remove the chance element from the system.

In the previous chapter, we trained the neural networks on the Markovian state from a simulation; however this is a stochastic process; for example in the "simple" Markovian model we have two "open" states; one short and one long. Since moving from the closed state to one of these open states is done via sampling from two exponential distributions, we might have an uncharacteristically long "short" opening or a short "long" opening (Figure 6.1). This creates downstream problems in training; there is no way for a neural network to always accurately predict the next simulated Markovian state even given perfect historical information, since the state is chosen stochastically. As is, the model will make unnecessary adjustments if it correctly guesses the "most likely" state, but simply

due to chance gets the simulated state wrong. This will incorrectly increase the loss function for the batch and cause unhelpful changes in parameters in the network. Perhaps more importantly is that in this case, this will have some further consequences on the channel's idealisation, since the model does not discern between the different errors of "incorrect state, correct number of channels" and "incorrect state, incorrect number of channels". Therefore, this so-called "stochastic error" both affects the Markovian state recovery as well as the channel idealisation.

In this Chapter we aim to improve deep learning Markovian state recovery performance by relabelling training sets with the most likely state rather than standard, simulated Markovian labels. By doing so, we expect to increase the speed of training as the gradient descent algorithm should make more consistent changes, and by our training protocol reach convergence sooner.

6.2 Methods

6.2.1 Dataset re-labelling Algorithm

As described above, the stochastic error phenomenon described above affects the Markovian state recovery as well as the channel idealisation. Therefore we felt the solution to this issue would be to provide the neural network the most likely state for the simulation to be in given the observations rather than the simulated state from the data generation. There are a number of algorithms for the discrete Markov case for finding the most likely state given a series of observations (Baum, Petrie, Soules, & Weiss, 1970; Viterbi, 1967) for both cases where the transition matrix is known (Viterbi) and unknown (Baum-Welch). Usage of these algorithms in ion channel idealisation is not unusual; the existing SKM method (Qin, 2004) extends the Viterbi algorithm to find the most likely transition rate parameters and uses them as a basis to construct the idealisation in a recursive manner.

Our approach is slightly different; instead of repeatedly applying the Viterbi algorithm and using parameter re-estimation to approximate the most likely Markovian state, since

we know the transition rate matrix *a priori*, we can simply run the Viterbi algorithm once on our observations to get the most likely Markovian state for each point and use this to train the deep learning model rather than the simulated state.

The Viterbi algorithm has two steps, the forward and the backward pass; and works in the following way. First, given a series of known observations $y_0, y_1, \dots y_T$ sampled from a known $N \times M$ emission matrix B, and a known $N \times N$ transition matrix A, if we have initial state probabilities $\Pi = \{\pi_0, \pi_1, \dots, \pi_{N-1}\}$, state space $S = \{s_0, s_1, \dots, s_{N-1}\}$ and emission space $E = \{e_0, e_1, \dots, e_{M-1}\}$, then we construct two new matrices C_1, C_2, both of sizes $T \times N$. C_1 corresponds to the probability of being at each state given each observation and C_2 the pointer to the previous most likely state. We set the first row of C_1 to be equal to the initial probabilities to equal $\{\pi_i \cdot B[i, y_0] \forall\, i \in S;\}$; i.e. the probability of seeing the first observation given the initial states.

We then start the forward pass; we iteratively fill C_1 and C_2 by considering the previous row in both matrices using the following formula:

$$C_1[j, i] = \max_k \{C_1[k, i-1] \cdot A[k, j] \cdot B[j, y[i]]\}$$

$$C_2[j, i] = \operatorname*{argmax}_k \{C_1[k, i-1] \cdot A[k, j] \cdot B[j, y[i]]\}$$

This constructs the two matrices with the probabilities of each state as well as the most likely previous state. From here, we can start at the bottom of the matrix and back trace the most likely route, using the following method: Set $D_T = \operatorname*{argmax}_k \{C_2[k, T]\}$ and $E_T = s_{D_T}$. Then we can calculate the most likely sequence E by iterating backwards through C_2:

$$D_{i-1} = C_2[D_i, i]$$

$$E_{i-1} = s_{D_{i-1}}$$

The complexity of this algorithm is $O(TS^2;)$; with no dependency on the size of the emission matrix.

The Viterbi algorithm works on discrete hidden Markov models, however it can be extended to continuous Markovian systems in a number of ways. One way, utilised in software such as QuB is to turn the system into a discrete system by only considering the state at a certain interval. As a signal has a sample rate (e.g. 10kHz), and any information between samples is lost, this is sufficient for most cases. Another approach, explored here, is turning the system discrete by considering the emission matrix not just by the number of channels open and closed, but also with the observed dwell time for the observation. This significantly grows the emission matrix from $N \times M$ to $N \times T$, where T is the number of events in a record; but since the Viterbi algorithm has no performance dependency on the size of the emission matrix, the only computational cost in this method is construction of the new matrix.

Construction of the new emission matrix is straightforward; we set any indexes with states corresponding to a different number of channels than observed to 0 (since for example the probability of an open state giving closed data is nil), then for the indexes with the correct number of channels, calculate the probability $P(E > e_i|S = s_j)$- where e_i is the observed dwell time and s_j is each state. To do this, we use the CDFs for the exponential function $(1 - e^{-\lambda t})$ with rate parameter A_{s_j}. Then we normalise the vector of probabilities to avoid vanishing terms.

6.2.2 Model training

To test if training the neural networks on the most likely state (via the Viterbi algorithm) performed better than the output of the simulation, we train each model in an identical fashion to the previous chapter. We measure the F1-Score and Cohen's Kappa Score for both the Markovian state recovery and "reduced" channel idealisation to measure how using the Viterbi algorithm affects model performance, comparing the predicted state to the simulated, stochastic state. We also measure the number of epochs used to train before early stopping; as the protocol allows training to continue indefinitely until performance plateaus; using the Viterbi algorithm as pre-processing is still of interest if the model

performance is similar but training time is reduced. Figure 6.2 gives an example of the output of the pre-processing, showing not only the most likely state, but the probabilities of each state for each point.

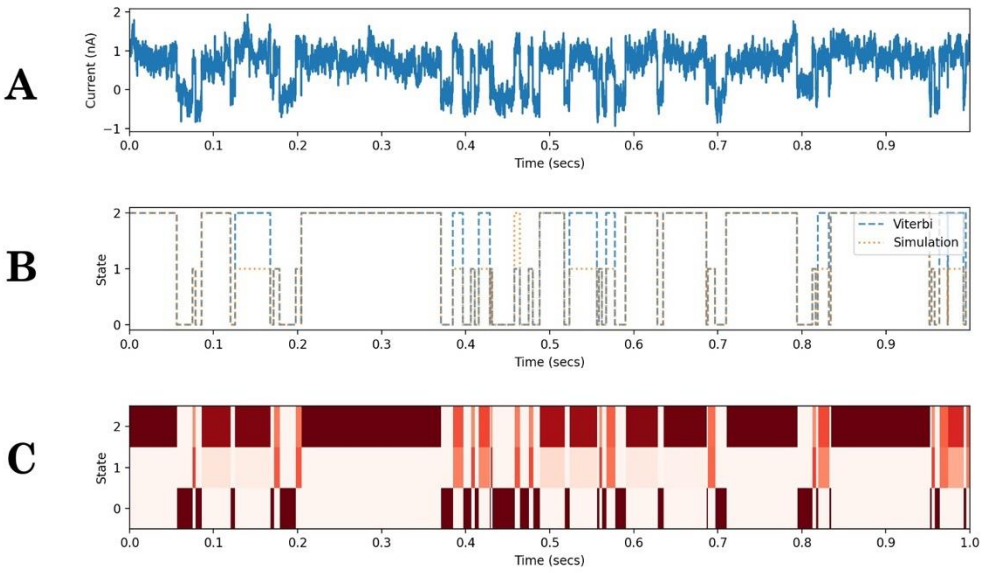


Figure 6.2: Example data trace (A), the difference between the most likely state and simulated state (B), and the probability distribution of being at a given state at any time (C)
In cases where we experience Markovian "unluckiness"; for example an uncharacteristically long "short close", it is not fair to punish a neural network for an incorrect prediction, as we can't expect AI models to predict a stochastic process. Therefore, we implement the Viterbi algorithm, producing a new labelling of the states for the most likely state given the open/closed conductance level record instead of the simulated state. In B, we see that there are many places where the Simulated (orange dotted) record deviates from the Viterbi (blue dashed) record, which are areas where a model may well reasonably predict the most likely state, but unfortunately get "unlucky", with the other closed state being randomly picked. C shows the probability distribution using the forwards-backwards algorithm for each time point. Note that the probabilities are not only a product of the dwell (in this case, longer opens are more likely to be state 2 than 1), but will also be a function of the previous state's probabilities as well (in this case, less so, as both state 1 and 2 always return to 0.

6.3 Results

After training the models on the Viterbi pre-processed data, to produce the new "most likely state" labels, models were trained until their performance showed no significant improvement as described in the training protocol in Chapter 5. From here, we compared the performance of the new models to those also used in Chapter 5. We used the same eight training/testing datasets as before (3-state low noise "static", with and without noise, 3-state high noise "perturbed", with and without drift and then a further four analogous

datasets which was based on the 5-state Markovian scheme). However, following the performance characterisations in Chapter 5 we now use only SimpleCNN, SplitCNN and U-Net. Since we also demonstrated in Chapter 5, that the F1-scores bore little relation to Cohen's kappa or visual inspection of the raw data, we will present only Cohen's Kappa below, but the more commonly used metric, F1, will be available in the supplementary figures.

6.3.1 Three State Model Datasets

Figure 6.3 shows the raw data for the simpler of the two Markovian state models and the static transition rate matrix in the absence of drift. For each of the three well performing CNN models we show the prediction outcomes for the conventionally labelled data alongside the augmented labels with Viterbi "most-likely-state". From subjective inspection of these raw records, there is not a clear difference between the standard and most-likely-state. Figure 6.4 show the Cohen's kappa metrics. Whilst performance overall is reasonable with full Markovian Kappas of 0.8164 ± 0.073,0.8194 ± 0.0730, 0.8475 ± 0.0735, 0.8369 ± 0.0851, 0.7951 ± 0.0302, 0.7954 ± 0.030 for SimpleCNN, SimpleCNN with Viterbi, SplitCNN, SplitCNN with Viterbi, UNet and UNet with Viterbi respectively; there are no statistically significant improvements in performance of any of the models when compared to their Viterbi counterpart (e.g. SimpleCNN versus SimpleCNN Viterbi). Similarly, there is the same lack of statistically significant improvement in model performance for the *3-state-static with drift* (Figures 6.5, 6.6), *3-state-perturbed with no drift* (Figures 6.7, 6.8) or *3-state perturbed with drift* (Figures 6.9, 6.10).

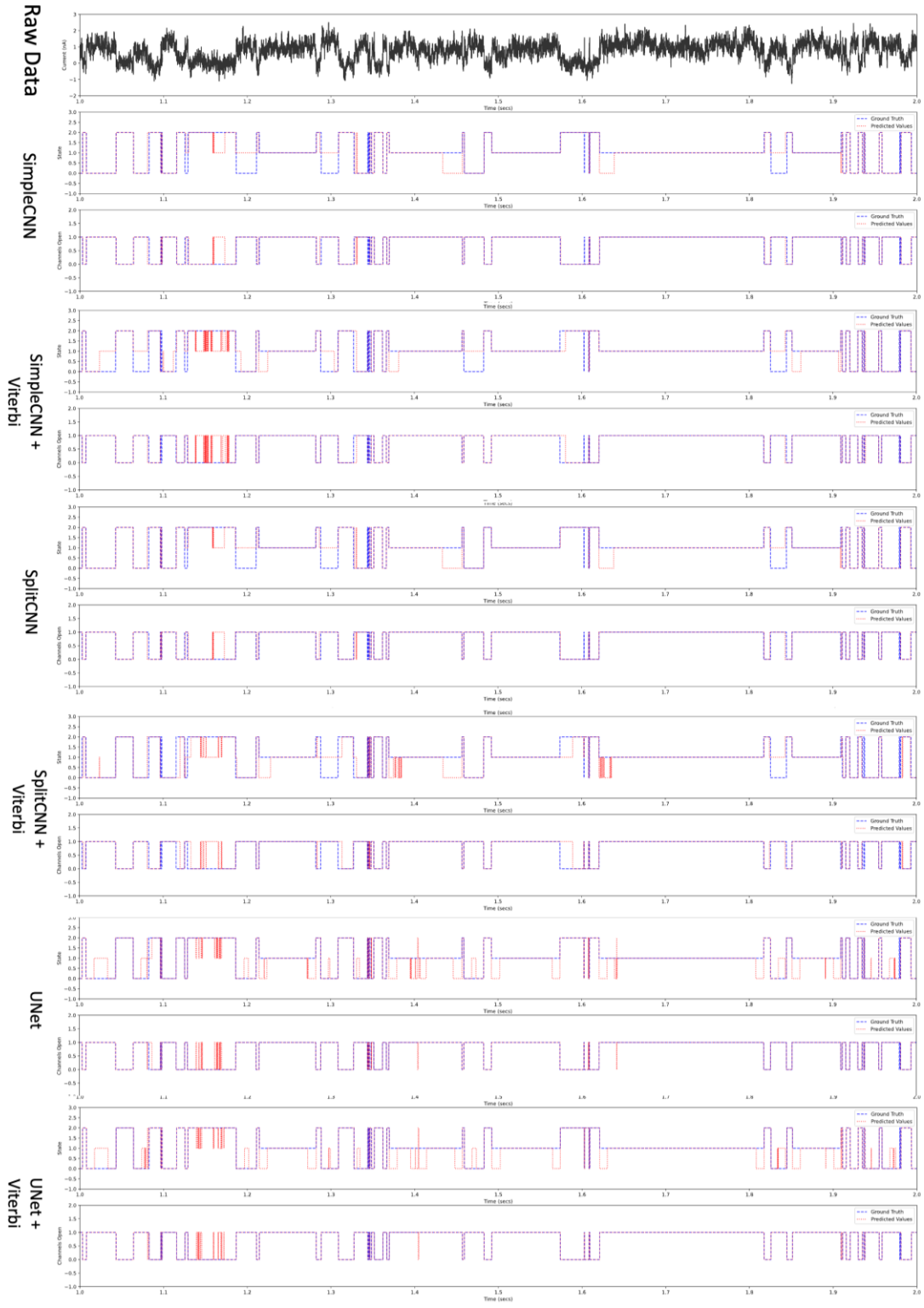


Figure 6.3: Example raw data trace (black), state recovery and channel idealisations for models on the "Static three state model without Drift" dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. We see that performance across the board is fairly similar, with some incorrect idealisations; but no significant qualitative differences as in Chapter 5.

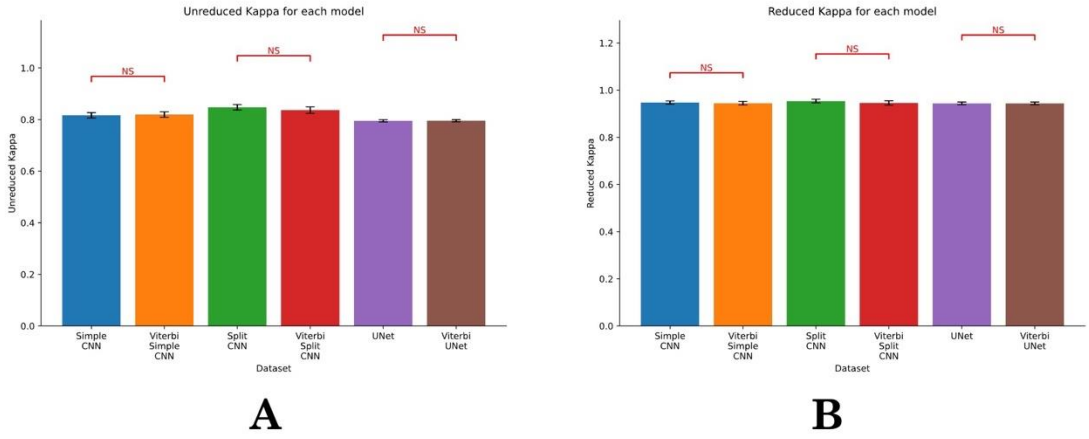


Figure 6.4: Model Training Metrics for "Static Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

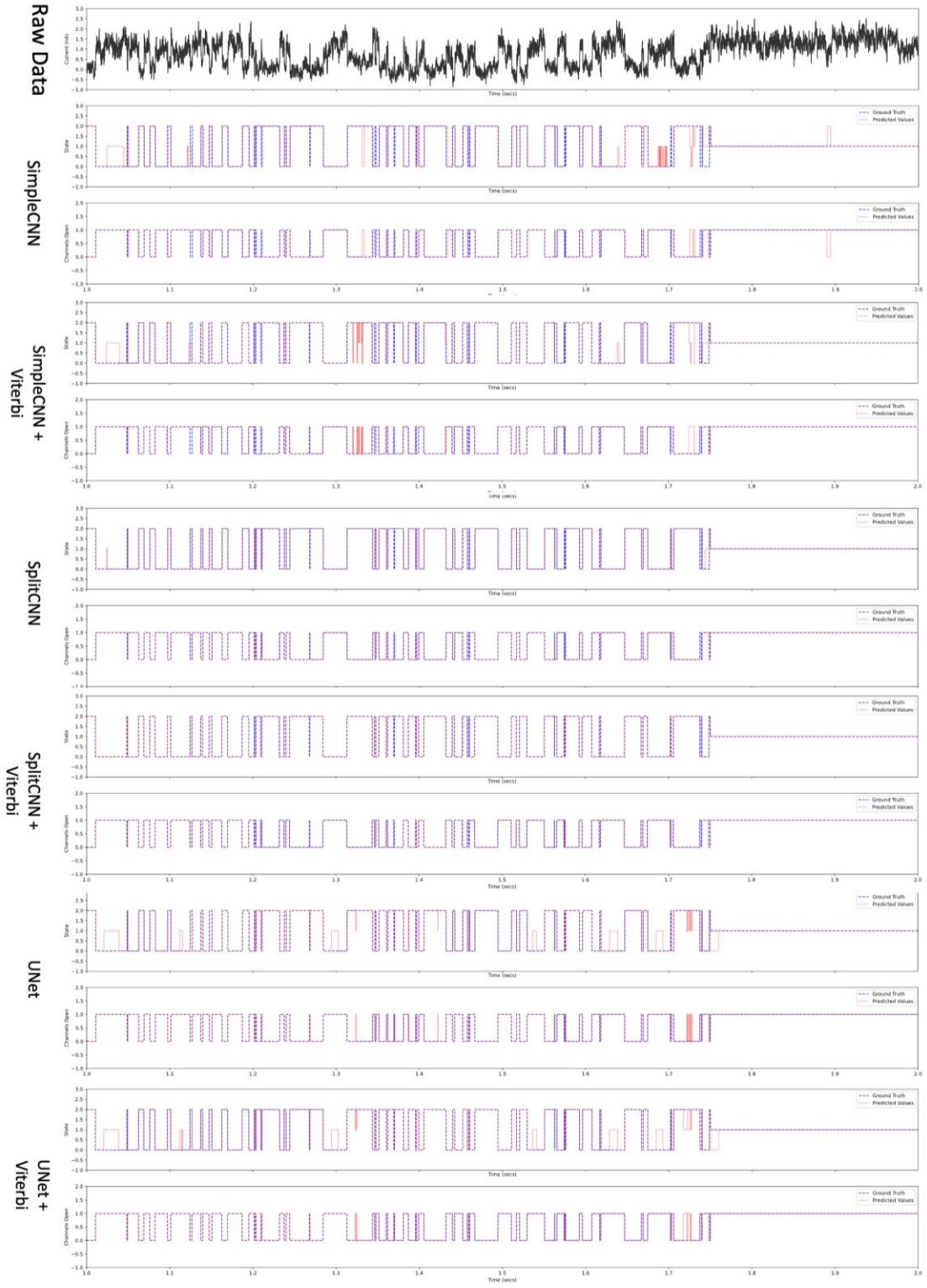


Figure 6.5: Example raw data trace (black), state recovery and channel idealisations for models on the "Static three state model with Drift" dataset.
Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. As with the static three state model with drift dataset, we see similar performance across all models.

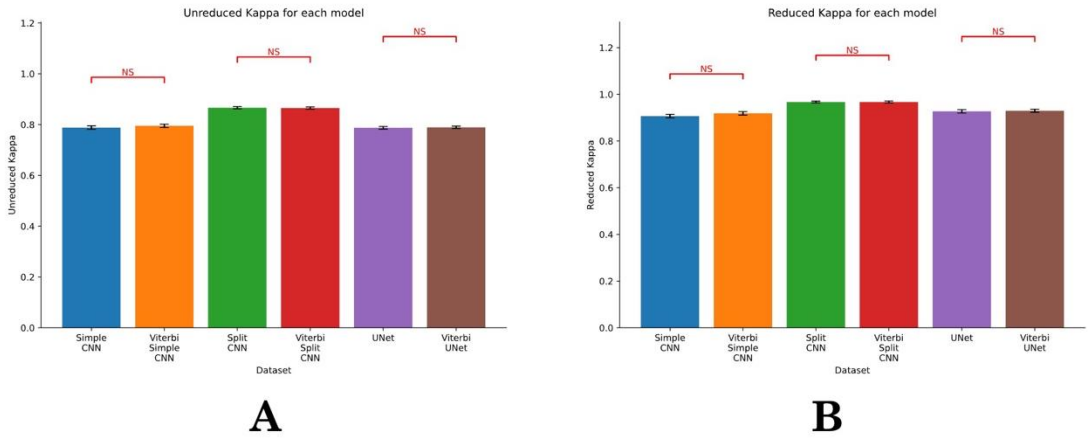


Figure 6.6: Model Training Metrics for "Static Three State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

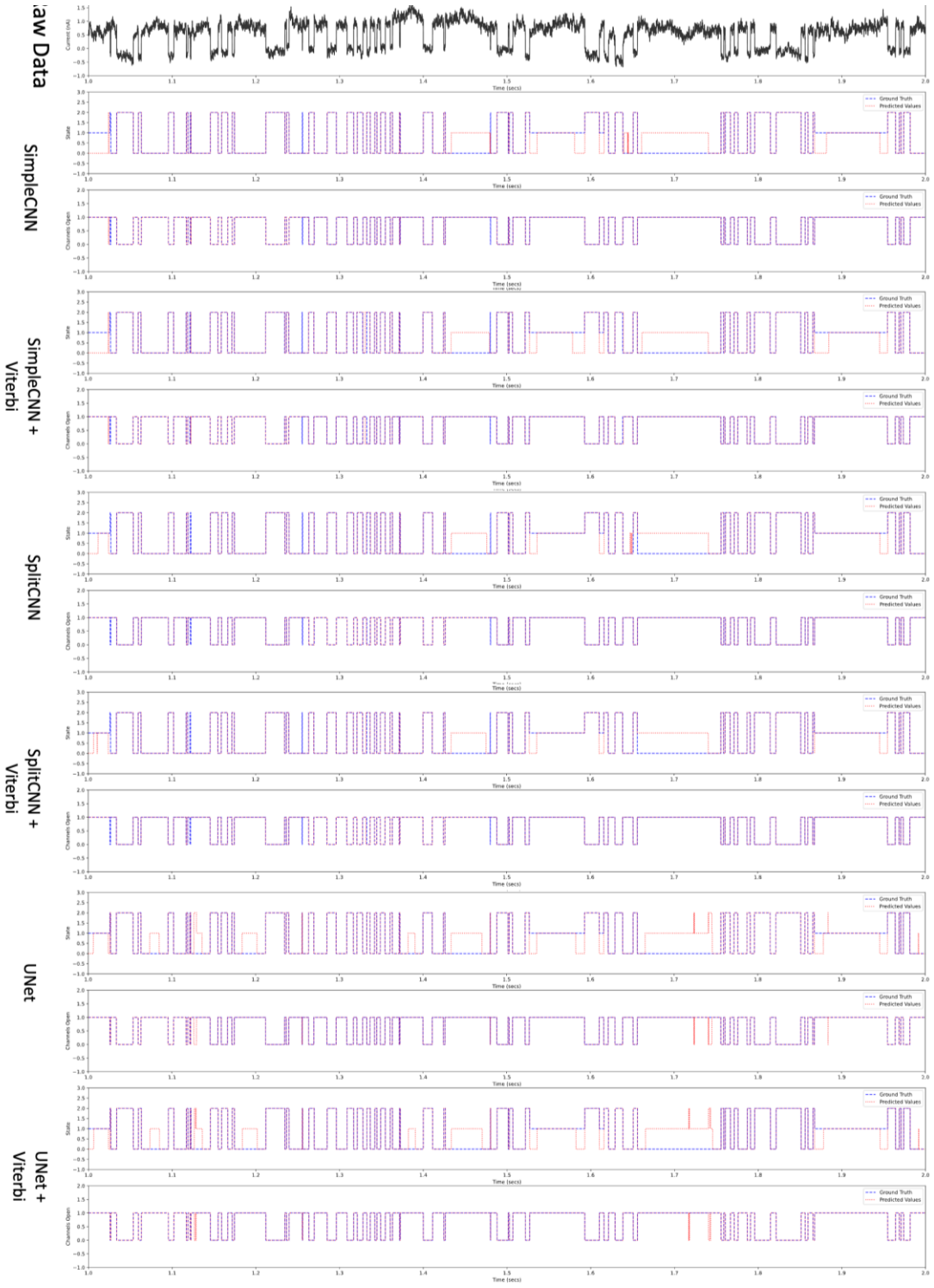


Figure 6.7: Example raw data trace (black), state recovery and channel idealisations for models on the "Perturbed three state model without Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. As with the static three state model with drift dataset, we see similar performance across all models.

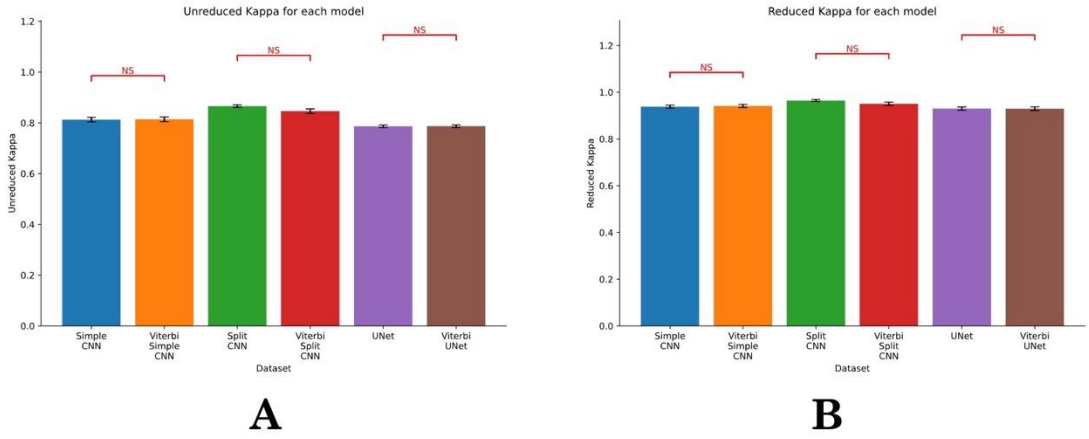


Figure 6.8: Model Training Metrics for "Perturbed Three State Model without Drift" Dataset
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

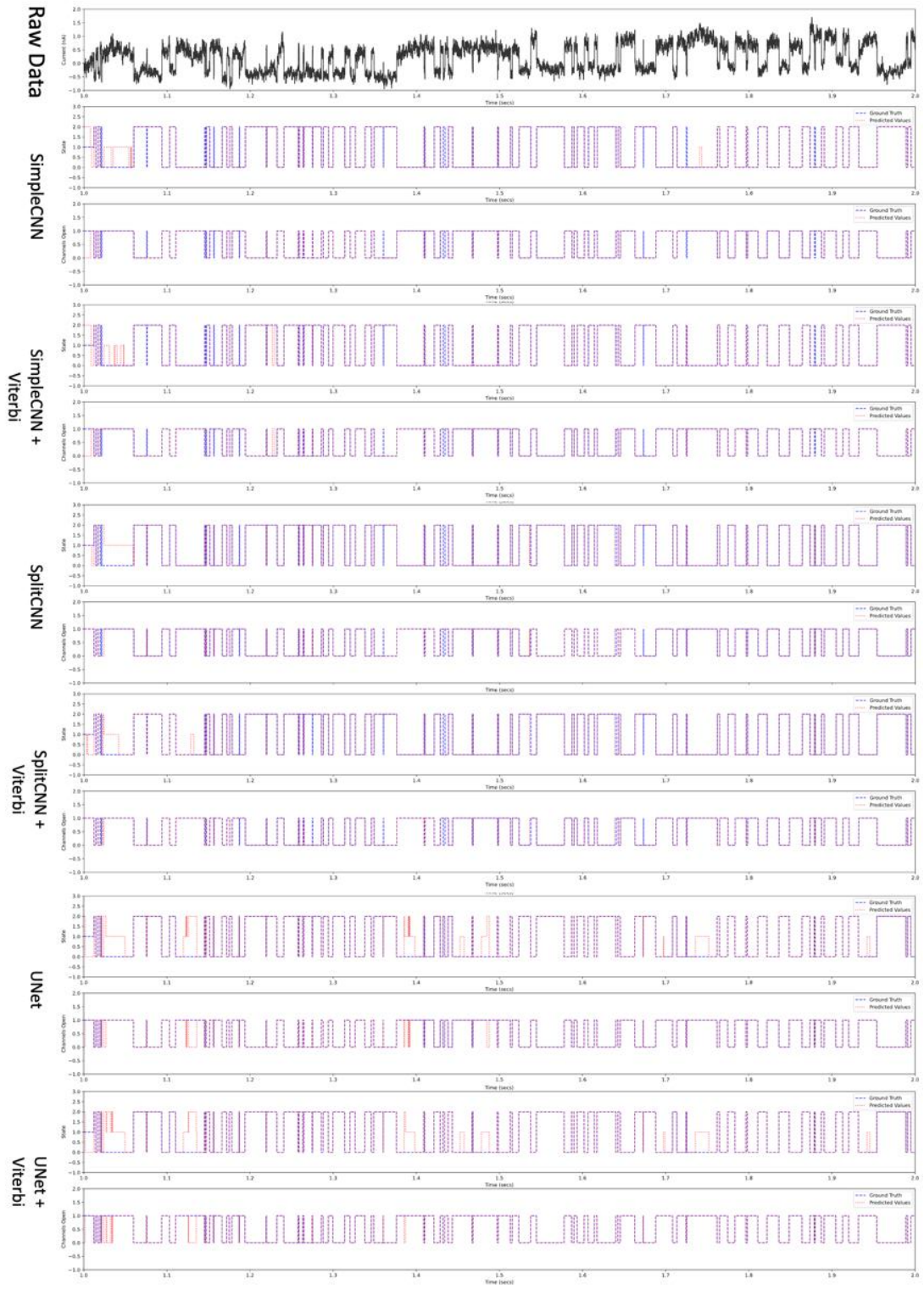


Figure 6.9: Example raw data trace (black), state recovery and channel idealisations for models on the "Perturbed three state model with Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. As with the static three state model with drift dataset, we see similar performance across all models.

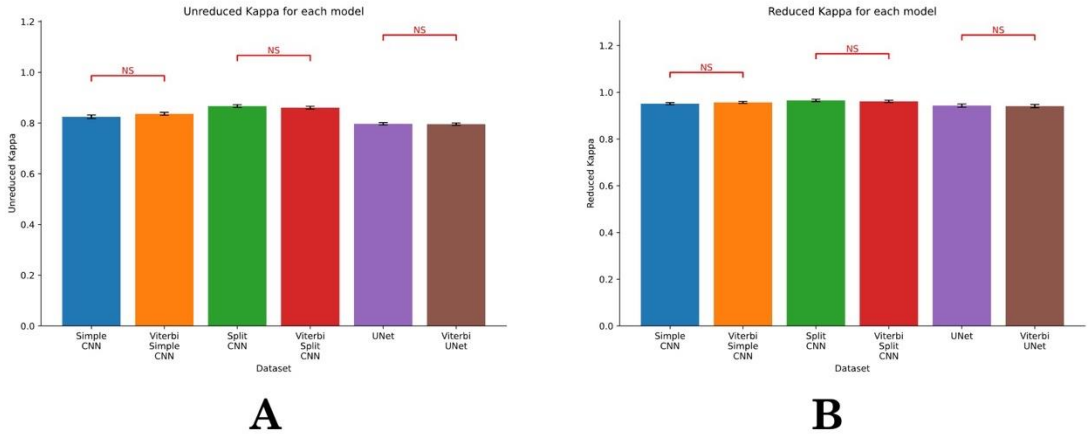


Figure 6.10: Model Training Metrics for "Perturbed Three State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

6.3.2 Static Five State Model without Drift.

Here we simulated data using the more complex 5-state Markovian scheme described in Chapter 5 and again compare the performance between the standard model state labels and the case where we have replaced the labels with the Viterbi "most-likely" state. Figure 6.11 shows representative examples of the raw comparisons. The Cohen's kappa metrics are shown in 6.12 for the full Markovian recovery and reduced open/closed idealization respectively. Here, unlike the 3-state equivalent above we do see statistically significant performance increases for the SimpleCNN and SplitCNN, but the improvements are small. For example, SimpleCNN improves from 0.8068 ± 0.0122 to 0.8144 ± 0.0104, $p < 0.05$, $n=24$ on implementation of the Viterbi most-likely labels.

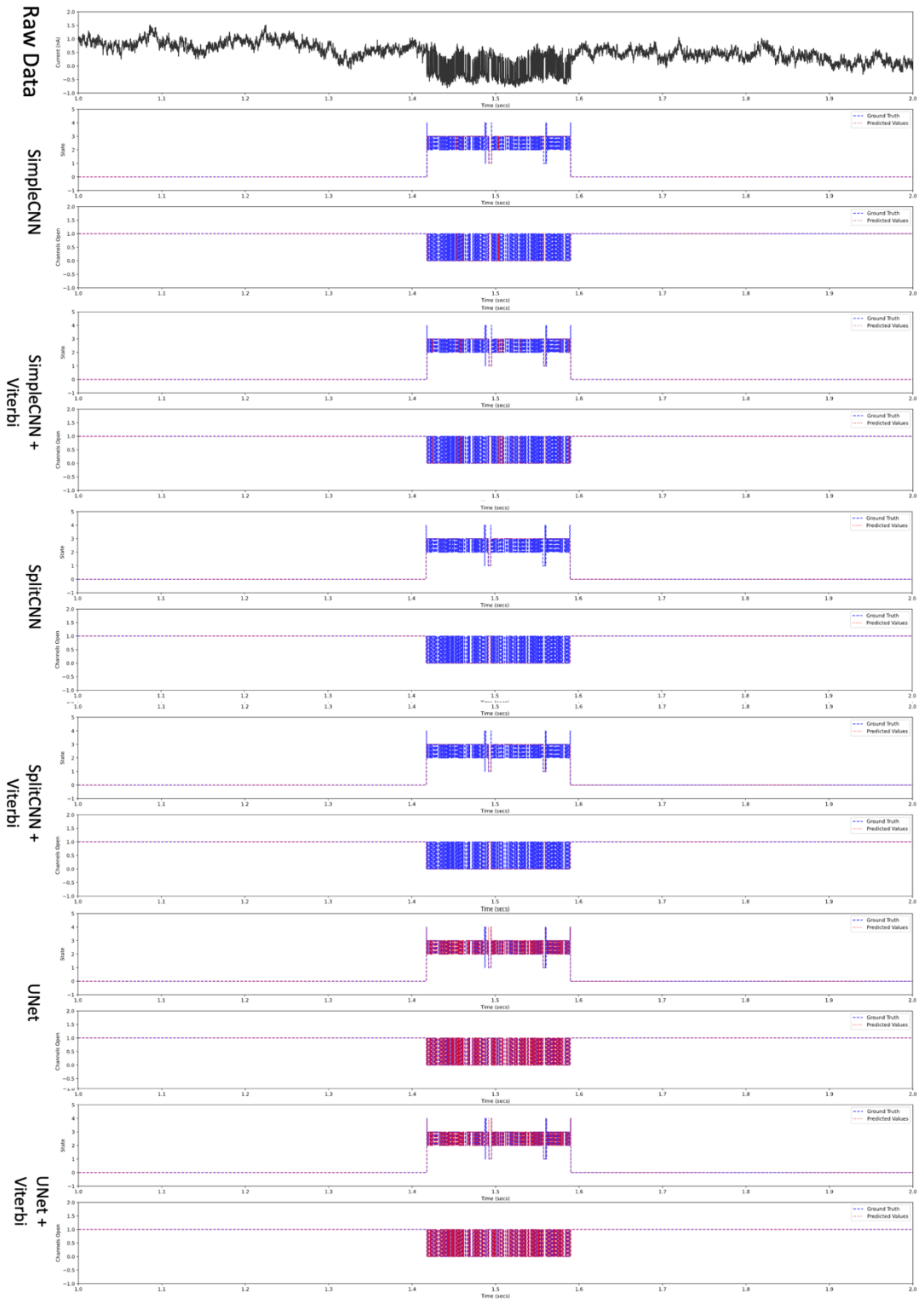


Figure 6.11: Example raw data trace (black), state recovery and channel idealisations for models on the "Static Five state model without Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. Here we see a split in the model performance; both UNet models are the only models to correctly identify the flickering block, whereas the Simple and Split CNN models both fail on these sections, detecting it as one large event.

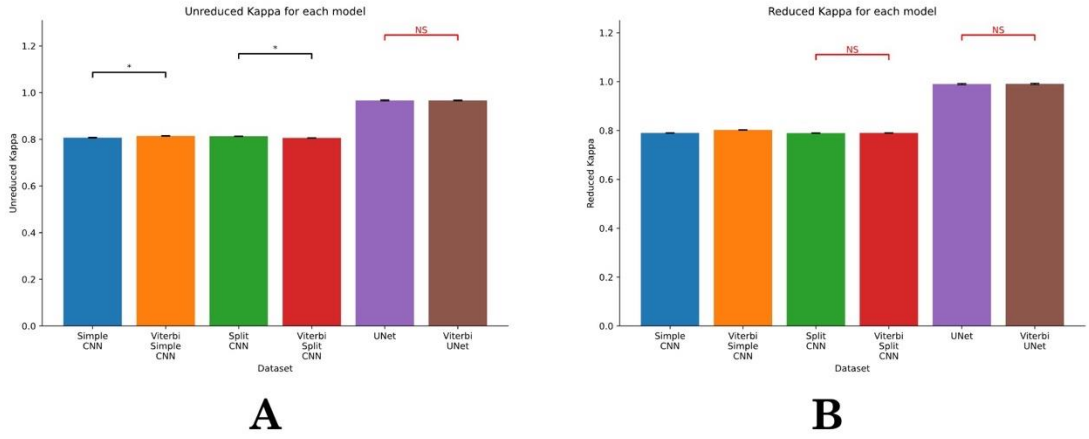


Figure 6.12: Model Training Metrics for "Static Five State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

6.3.3 Static Five State Model with Drift

As reported in Chapter 5, we find U-Net fails to predict target labels statistically significantly (Figures 6.13, 6.14), but for the remaining comparisons, Simple and SplitCNN there are very small, but statistically significant improvements with the Viterbi most-likely labels. Interestingly, with the slightly more complex dataset, *Five State Hard without Drift* (Figures 6.15, 6.16) there is no significant improvement with Viterbi labelling, but again with the most complex case *Perturbed Five State Model with Drift* (Figures 6.17, 6.18) there is a mix of statistically significant model performance change with Viterbi labels and others not so. For example, there are no statistically significant improvements in the SimpleCNN or U-Net model performances, there is a small *reduction* in performance with the SplitCNN (from 0.8386 ± 0.0098 to 0.8072 ± 0.0062 $p < 0.05$, $n = 24$).

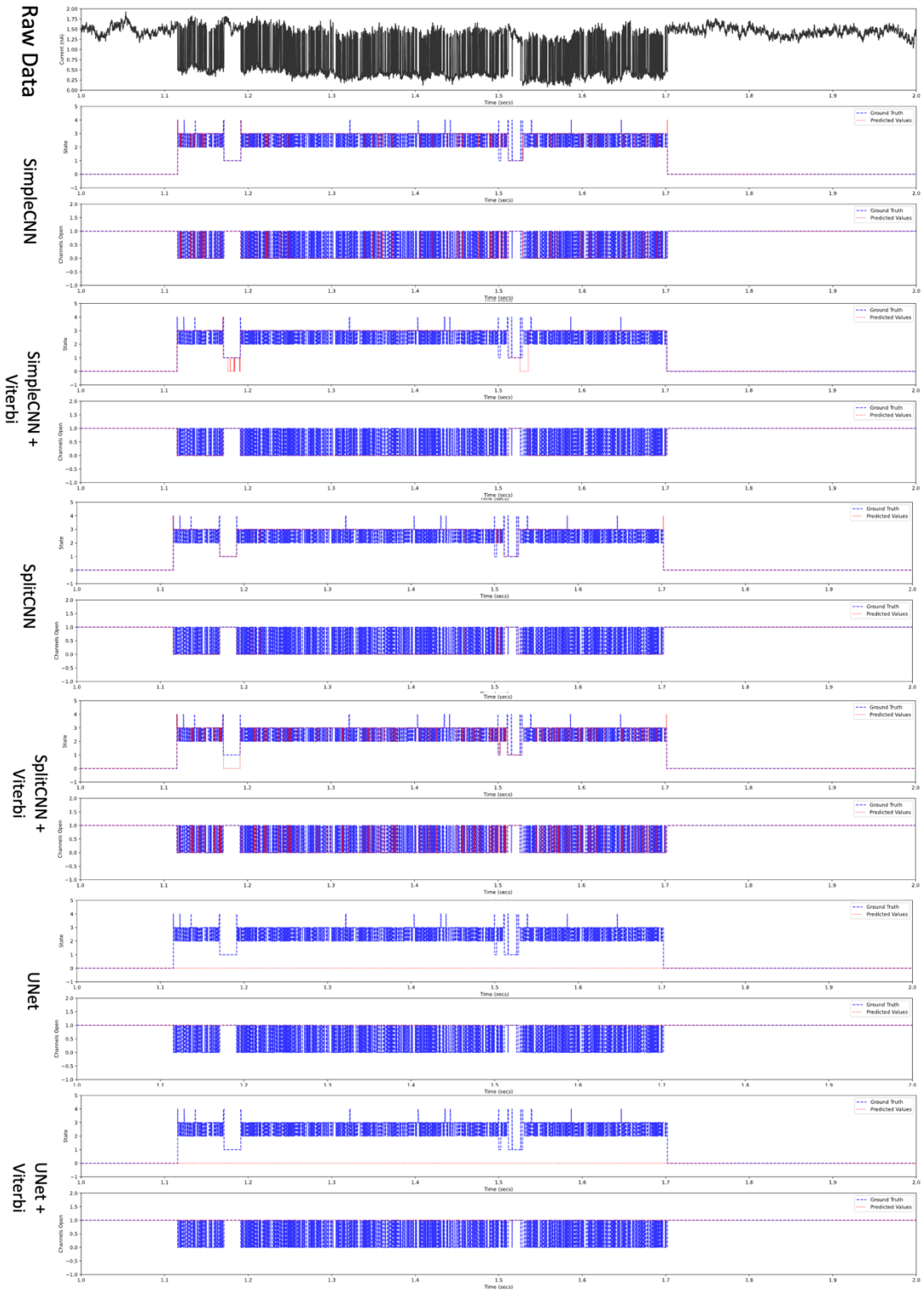


Figure 6.13: Example raw data trace (black), state recovery and channel idealisations for models on the "Static Five state model with Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. Contrary to the "static five state model without drift", both models suffer modal collapse here, with the Viterbi correction showing no impact on model performance.

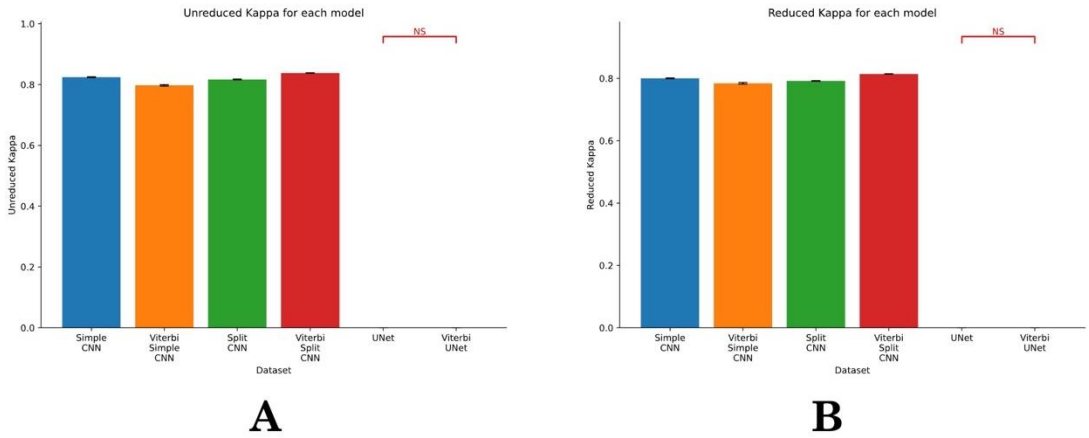


Figure 6.14: Model Training Metrics for "Static Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

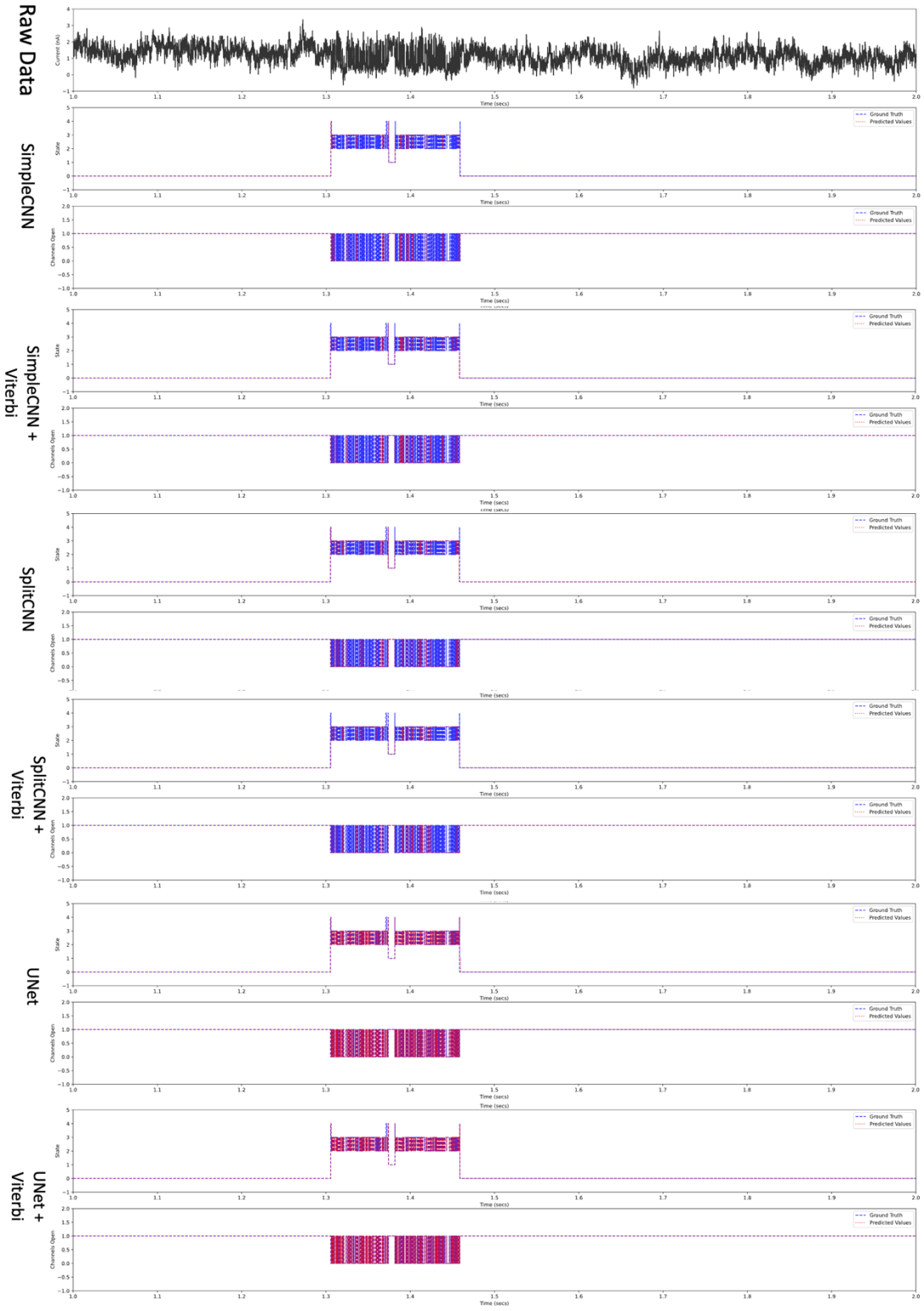


Figure 6.15: Example raw data trace (black), state recovery and channel idealisations for models on the "Perturbed Five state model without Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. Here the UNet models perform the best again, correctly identifying the flickering block in the center of the trace example. However there is still no significant improvement between the Viterbi and non-Viterbi versions of each model architecture.

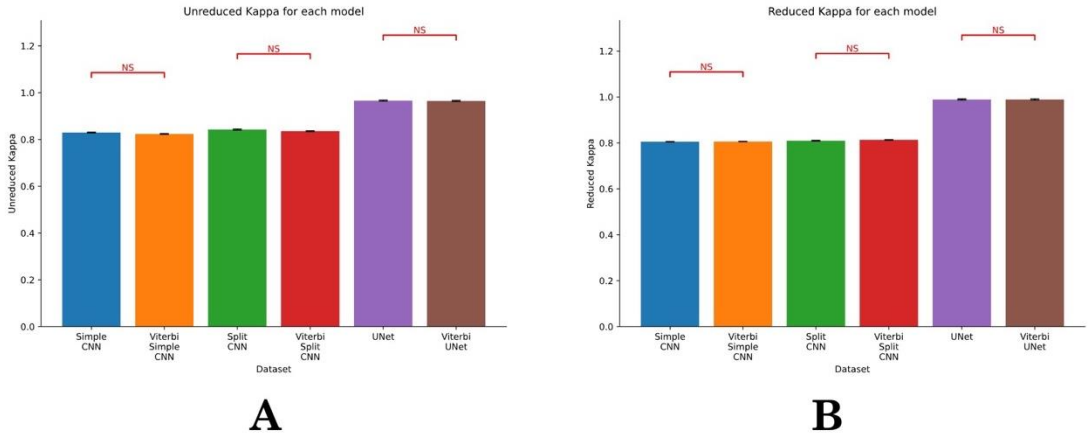


Figure 6.16: Model Training Metrics for "Perturbed Five State Model without Drift" Dataset
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

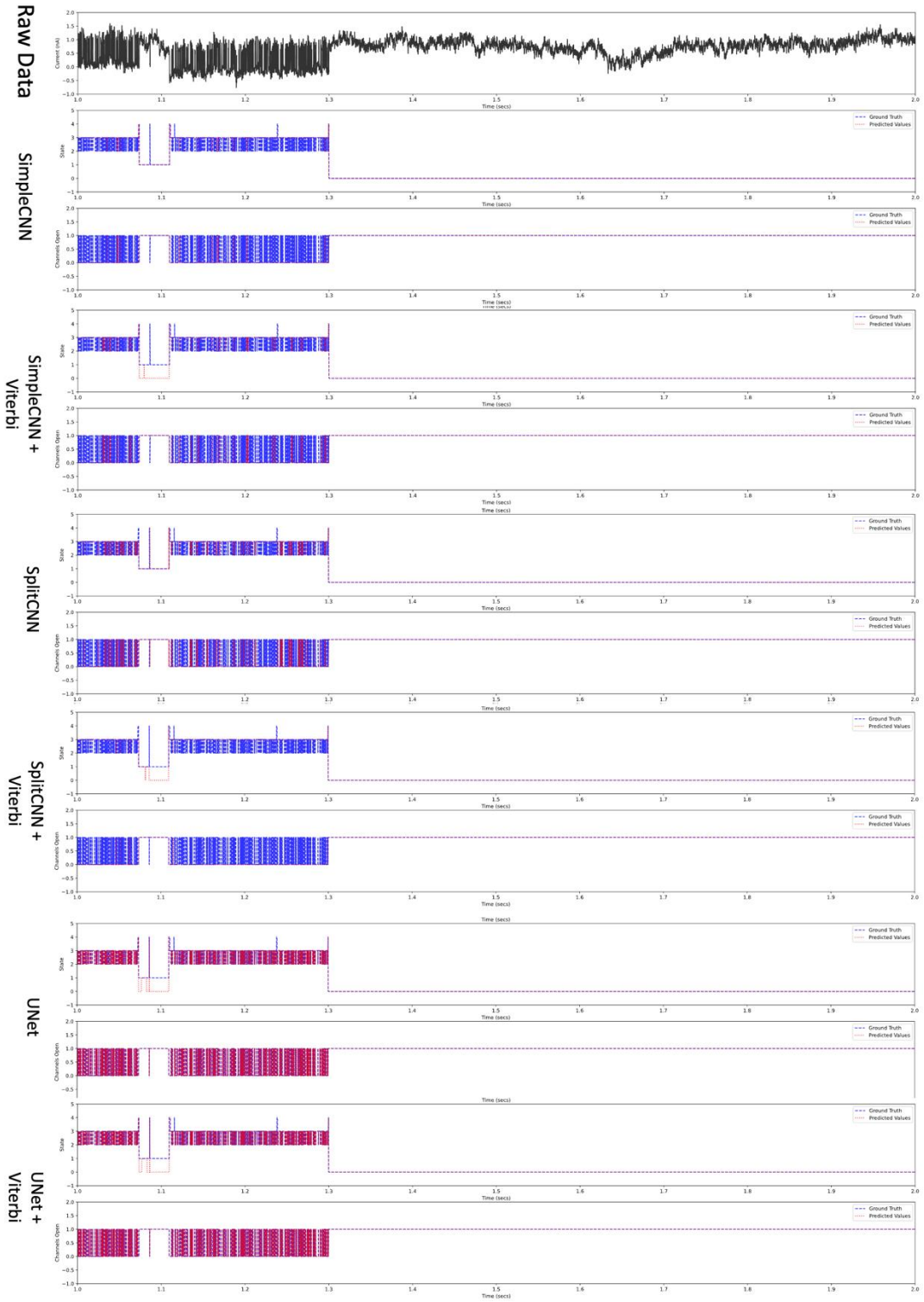


Figure 6.17: Example raw data trace (black), state recovery and channel idealisations for models on the "Perturbed Five state model with Drift" dataset.

Shown is 1 second of data at 10kHz in black along with each model's predictions (red) versus the ground truth (blue). For each model, both the Markovian recovery (left) and conductance level (open/closed) (right) is shown. This is thought to be the hardest of our datasets, however UNet again shows the highest performance, correctly identifying the flickering block at the start of the trace. There is still no significant difference between the UNet model trained on the simulated data versus the one with the additional Viterbi pre-processing step.

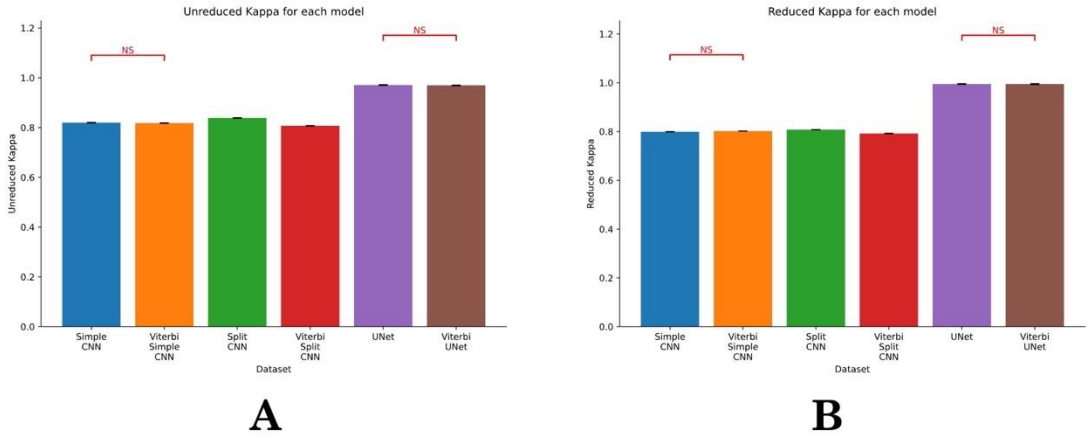


Figure 6.18: Model Training Metrics for "Perturbed Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

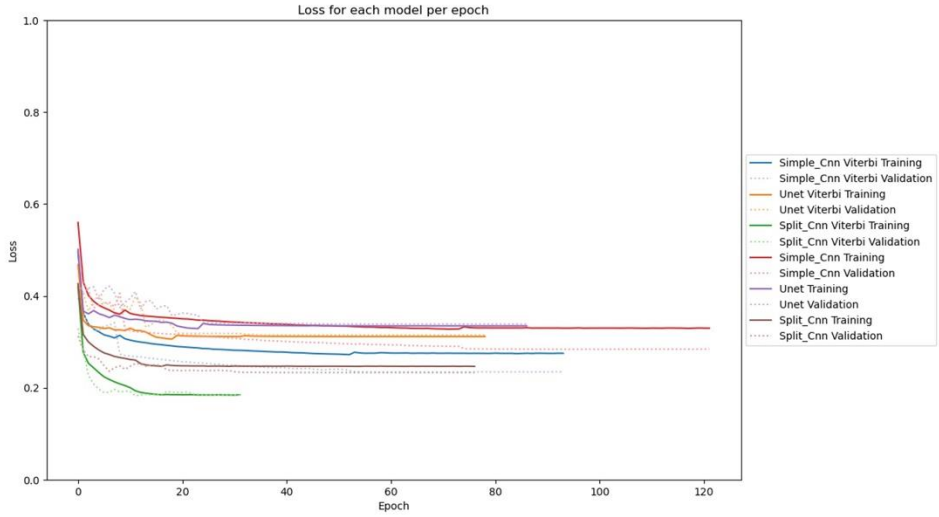


Figure 6.19: Model training loss per epoch for each model using Viterbi versus Simulated labels for the Perturbed Five State with Drift dataset.
AAlthough we see little change in the model performance of Viterbi versus simulated models, we do see a large decrease in training time before early stopping. In this case, the SplitCNN model took half as many epochs to reach peak performance with the Viterbi training labels than the simulated ones (green versus brown); with similar improvements in the SimpleCNN (red versus blue) and UNet (orange versus purple) datasets.
B..

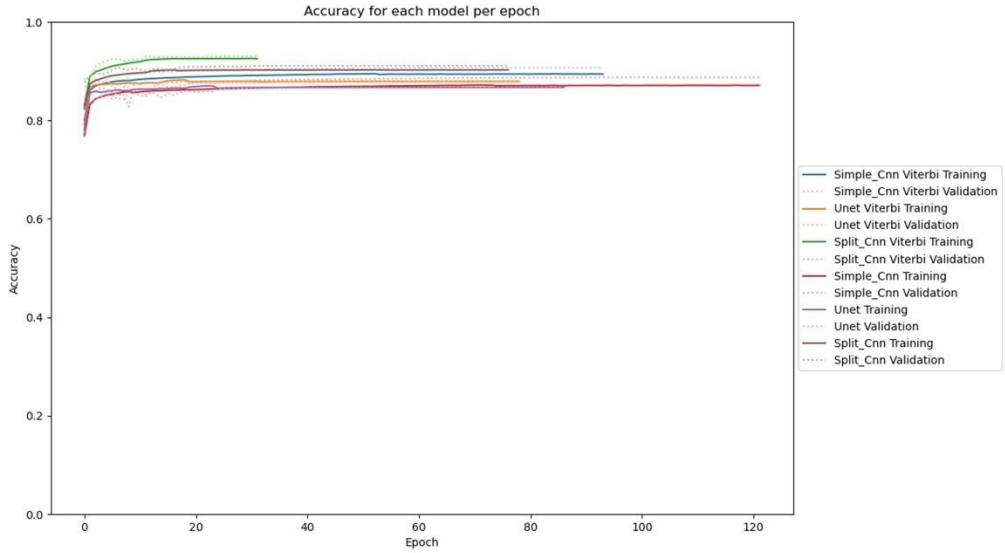


Figure 6.20: Model training accuracy per epoch for each model Viterbi versus Simulated labels for the Perturbed Five State with Drift dataset.

Although we see little change in the model performance of Viterbi versus simulated models, we do see a large decrease in training time before early stopping. In this case, the SplitCNN model took half as many epochs to reach peak performance with the Viterbi training labels than the simulated ones (green versus brown); with similar improvements in the SimpleCNN (red versus blue) and UNet (orange versus purple) datasets.

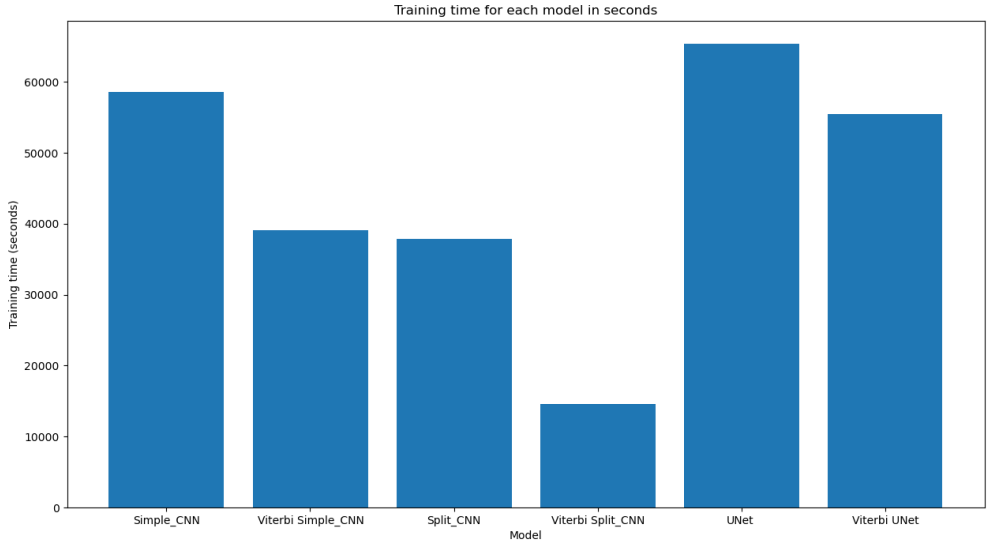


Figure 6.21: Model training times for each model in seconds.

In addition to training the models, we logged how long each model took to train. Here we can see that the Viterbi versions of the Split and Simple CNNs took significantly less time than the non-Viterbi versions, suggesting more efficient training.

6.3.4 Training Progress

Training progress was also tracked and compared to the previous chapter in Figure 6.19 amd Figure 6.20. From this we can see across the board improvement in model training speed from all 3 models, along with higher training metrics throughout. Indeed, when considering model training times (Figure 6.21), we see a significant improvement in how long the models trained before they stopped improving.

6.4 Discussion

We have seen that in this case, there is no consistent end-model performance improvement; and on those occasions where a statistically significant improvement was detected, the improvement was very small and unlikely to be noticed by an end-user. The application of the Viterbi algorithm for general machine learning is not unique, having been used for speech recognition natural language processing previously (Botros, Siddiqi, & Deiri, 1993); however the application of the Viterbi algorithm directly to the time series problem and deep learning is novel.

6.4.1 Training Times

There was a significant training time decrease for each model, however and this is likely due to the training process; the preprocessing's aim was to reduce the "confusion" during training because of the stochastic element in the dataset; this meant that the model was likely to eventually get correct state predictions on average, but take far longer. Therefore, although there is a slight initial cost to calculating the most likely state through the Viterbi algorithm; there is clear value in using this preprocessing pipeline to improve development cycles in practice.

In addition, the models were evaluated on the simulated state, rather than the most likely state. This was done to keep the benchmark the same from the previous chapter, however introduces a stochastic element to model evaluation resulting on a perfect performance never being possible. For comparative evaluation as the datasets are the same this does

not impact the validity of the results. Since Viterbi pre-processing significantly reduced training times without harming the model's performance (and in some slight cases, improving it); we will be continuing to use Viterbi pre-processing for later work in this thesis.

7 Improving Model Performance Through Progressive Windowing

7.1 Introduction

In previous work, we have split data into a sequence of "windows", with each point in the window being labelled with a state in the output layer. The size of this window is a hyperparameter arbitrarily controlled by the operator, with larger windows exponentially increasing the model size and training times. By splitting the data into such windows, we lose some context surrounding each point, as the CNN models only consider data inside the current window being analysed; for hidden Markov models this is potentially sub-optimal since each state encodes some information about the points surrounding it, as some states are inaccessible from others, and the dwell times of previous and future events could be used to infer the likely state of the current event. Therefore, the more surrounding data we have for each point, the more equipped a model should be to accurately predict the hidden Markovian state present at that time; but this comes at a large computational cost as increasing the window size exponentially increases the number of parameters in the model.

Events that are closer to a given point are more helpful for predicting that point's state than those further away, since the Markovian property has a reliance on only the previous state (although this implicitly may hold some information about previous states). Therefore, in theory the penalty for lack of context would not be uniform across the window; points in the centre of the window have a similar amount of context to either side, whereas points lying at the edges of the windows have a context imbalance, in extreme cases not knowing the previous state at all. Further investigation of this potential issue confirms this theory. By plotting where predictions errors occur within windows, we can show that there is indeed a significant performance decrease (considerably greater error

rate] towards the left and right edges of the windows (Figure 7.1). These Markov state prediction errors also occur in the channel idealisation, but to a lesser extent.

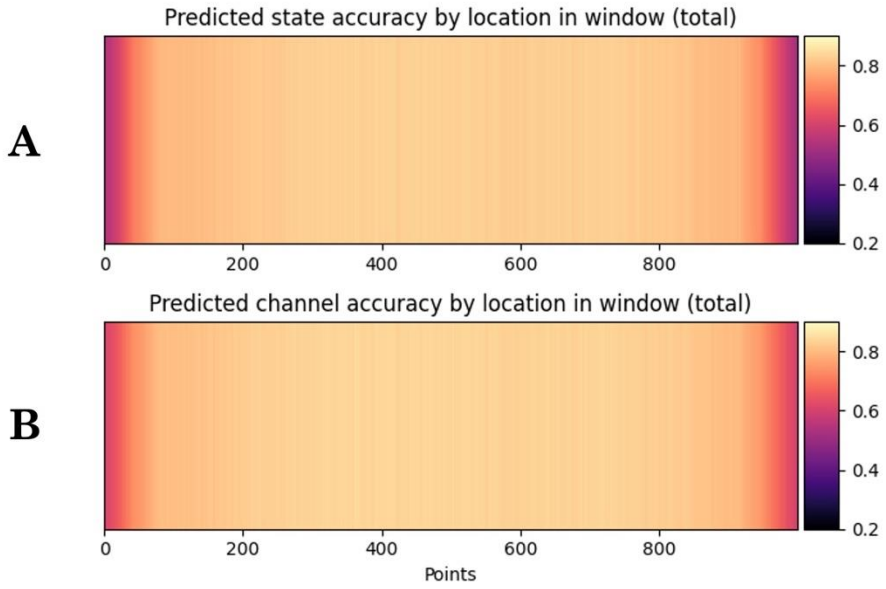


Figure 7.1: State and Channel accuracy for one of the "full window" models, on a per-window basis.
We take all the windows analysed by the CNN models and calculate the error rate for each position in the window. Here we can clearly see that the edges of the prediction perform on average significantly worse than the centre of the model. This is likely due to a lack of surrounding information and context for the edge points. Also note that this effect appears slightly more pronounced for the Markovian state prediction, perhaps since that is more associated with contextual data.

One approach to resolve this error would be to return to the approach used in DeepChannel (Celik et al., 2020), with the network using a LSTM model architecture to predict one point at a time. This comes at a severe functionality cost as we saw in Chapter 5 with our novel models significantly outperforming detection accuracy over the original DeepChannel model. Furthermore, detection speed was considerably slower since instead of processing a window of 1024 points at a time, we are predicting a single point for each run. Therefore, in this chapter we will consider whether prediction accuracy is improved if the model predicts the middle 512 points (half the original window, for simplicity of reforming the idealisation in post). We will then use overlapping windows to still cover the entire dataset

Aim: Improve prediction of Markov states with intelligent progressive windowing.

7.2 Methods

We train models similarly to chapters 5 and 6, on the same 8 datasets, with the same training protocol, including Viterbi analysis. However, in this case, we adjust the model architectures to only output the centre 512 points of the input. For the SimpleCNN and SplitCNN model, this is simply a case of reducing the size of the last dense layer; for the U-Net model, however, this is slightly more complex, requiring us to remove the last inverse convolutional layer to output a signal half the size of the original. This change brings us outside of the domain U-Net was designed for (1-to-1 image segmentation).

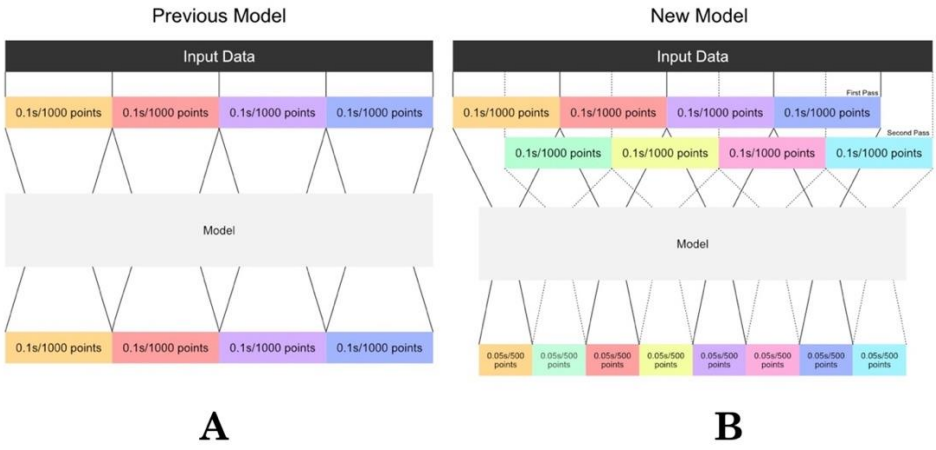


Figure 7.2: Comparison of processing for previous models and the new method.
In previous methods (A), there was a one-to-one correspondence between the input and output points. In the new model (B), each input window is only mapped to its centre 500 points. This means that 2 passes are required to completely predict the data, the first for points 250-750, 1250-1750 etc and the second for 750-1250, 1750-2250 etc. One disadvantage of this method is the first and last 250 points are not predicted "in place", but sewn together to make a new 1000 window consisting of the first and last 500 points. This creates a discontinuity in the data.

For training and testing, we now make two passes through each datafile (Figure 7.2). The first operates from points 0-1024, 1024-2048 *et cetera* predicting the middle points 256-768, 1280-1792 *et cetera*. The second pass predicts points 512-1536,1536-2560 predicting the points 768-1280, 1792-2304 etc. Then, these windows are combined in a way to get the full prediction of the file. One immediate drawback of this method is that the first and last 256 points of the record must be predicted by other means, as they are at the extremities of the datafile and cannot be contained in an output window. For this, we join the first

and last 512 points of the file (with a discontinuity in the centre), and use this to predict these points.

We then compare these new models with the ones without the reduced output size and measure both the F1-Score (*shown in the supplementary figures*) and Cohen's Kappa Score for both the state and channel predictions. We also monitor the training progress over the process, and record the time take for each model to train.

7.3 Results

We have 8 datasets below, the simpler 3-state model with high/low noise, with and without drift, and the more complex 5-state model with the same noise and drift combinations. We then compare the performance of our 3 "best" models (SimpleCNN, SplitCNN and U-Net) with and with the new progressive windowing. In all cases we again calculate metrics for the full Markovian state prediction and the so-called reduced case where we simply predict open and closed, which is equivalent to ordinary ion channel idealisation.

7.3.1 Static Three State Model without Drift

Inspection of the raw data (Figure 7.3) shows some events that were falsely predicted by the SimpleCNN, but not with the progressive SimpleCNN windows. Conversely, there seem (subjectively) to be considerably more false events in the Windowed U-Net model. Quantitatively (Figure 7.4), there were no statistical differences in Cohen's Kappa with any of the three comparisons except the U-Net which performed significantly worse on both Markov state recovery and reduced inference (ie idealisation).

The progressive windowing models performed similarly on the remaining three-state datasets; *Static Three-state-Model with drift*, *Perturbed Three-state-model without drift*, *Perturbed three-state-model with drift*. In each case the progressive window U-Net performs visibly worse (Figures 7.5, 7.7 and 7.9) than the control and this is supported by the quantitative analyses with Cohen's Kappa (Figures 7.6, 7.8 and 7.10). However, there

was a consistent and statistically significant improvement with progressive windowing of the SplitCNN model, in terms of our primary objective, Markov state recovery.

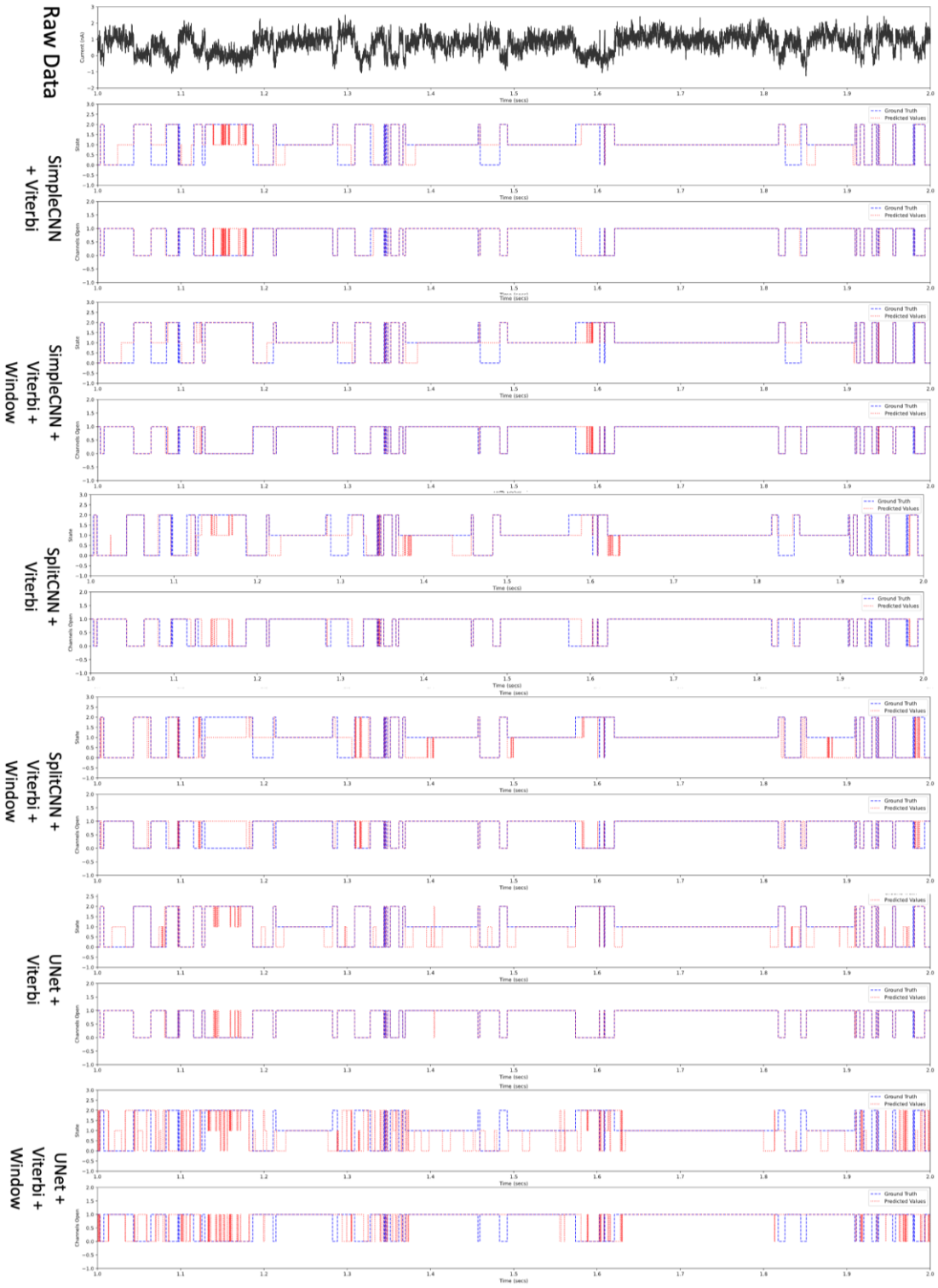


Figure 7.3: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Static Three State Model without Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

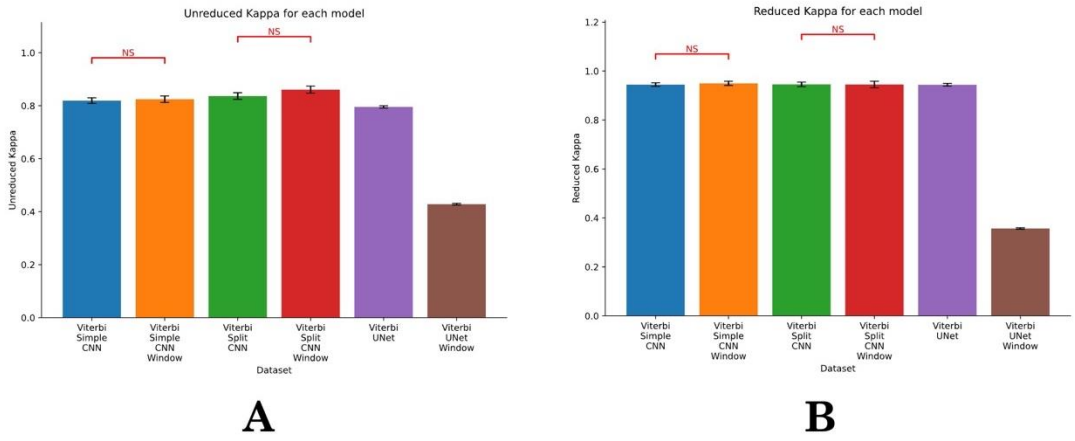


Figure 7.4: Model Training Metrics for "Static Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

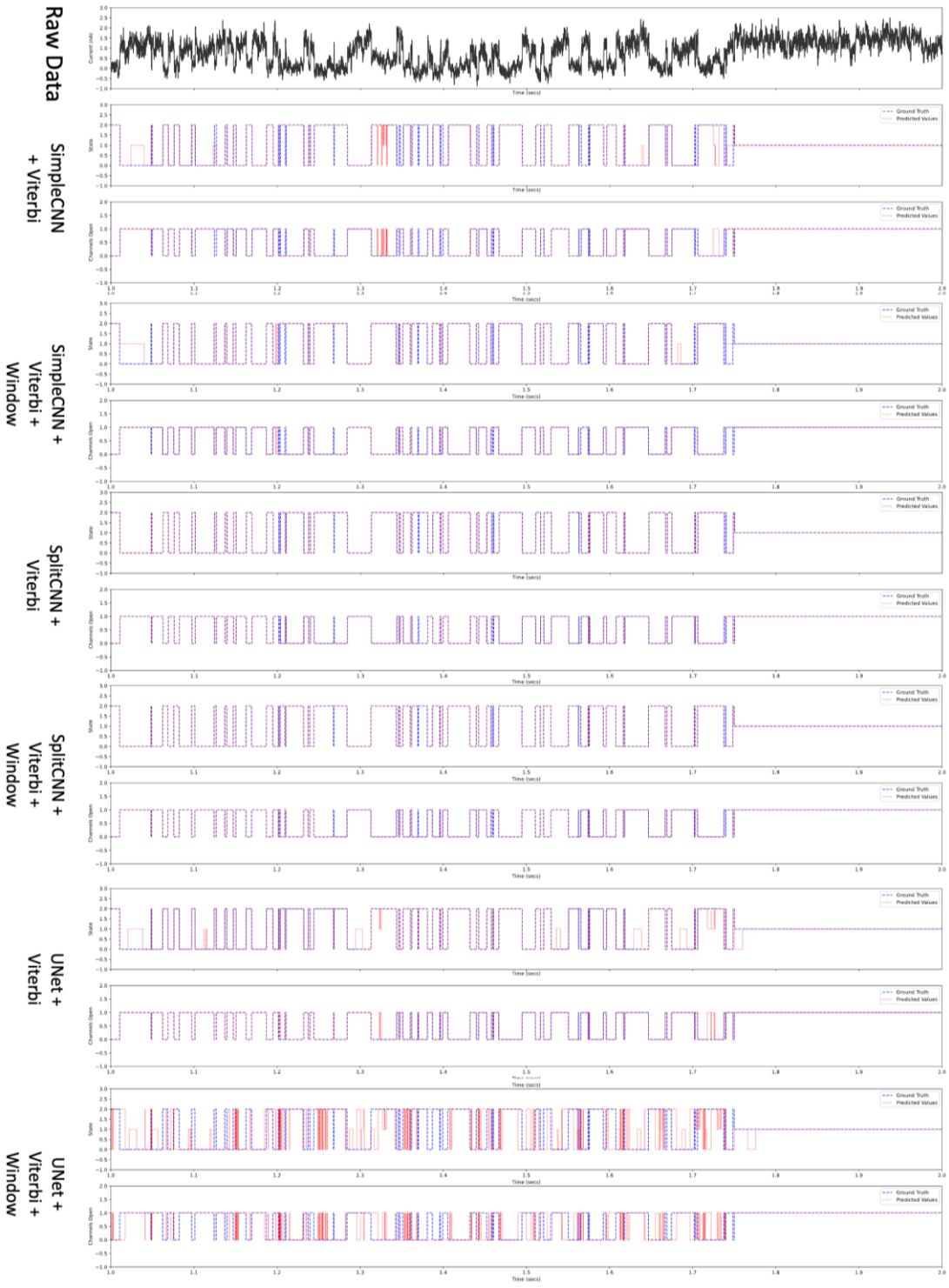


Figure 7.5: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Static Three State Model with Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

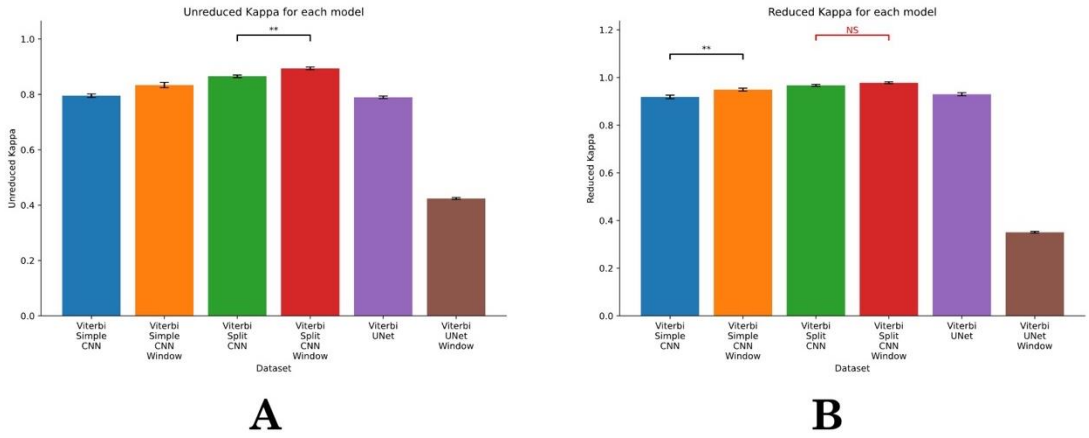


Figure 7.6: Model Training Metrics for "Static Three State Model with Drift" Dataset
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

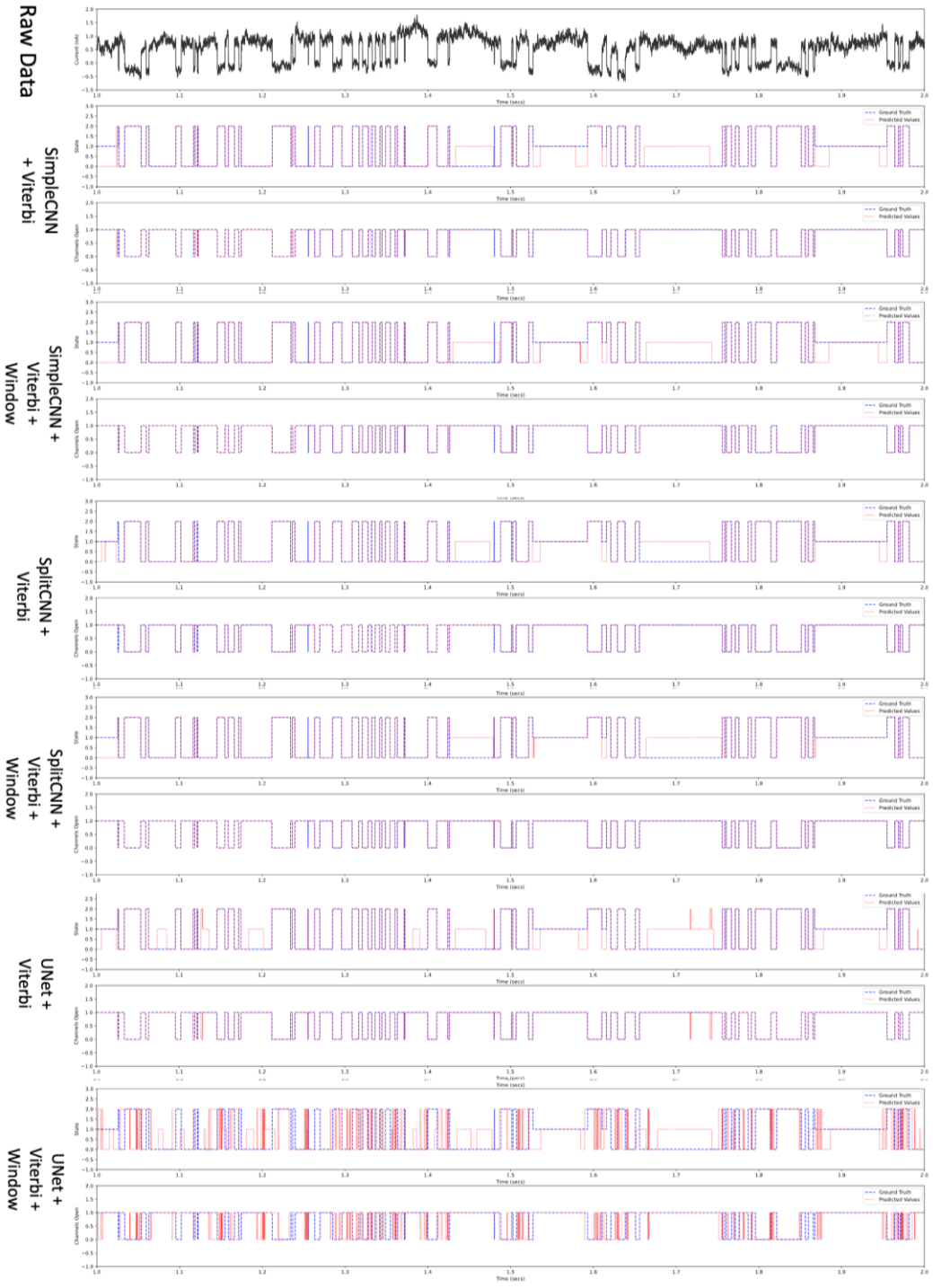


Figure 7.7: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Perturbed Three State Model without Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

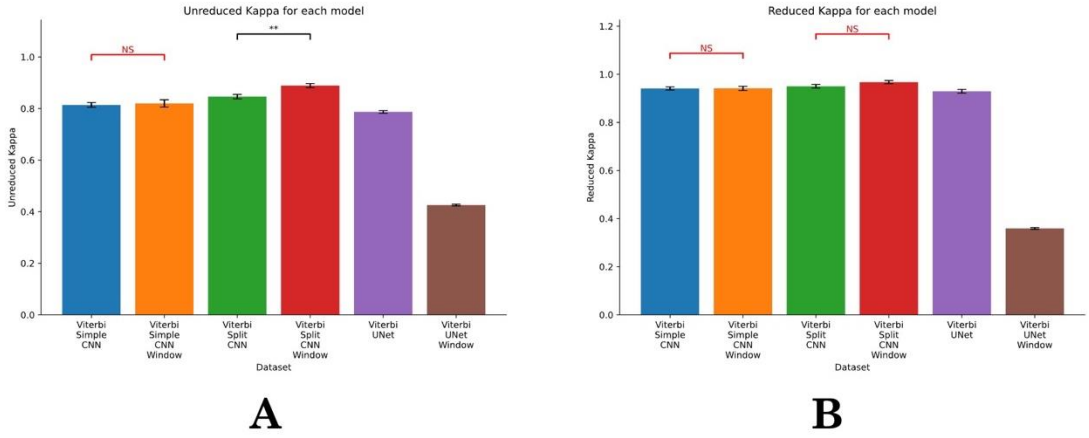


Figure 7.8: Model Training Metrics for "Perturbed Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

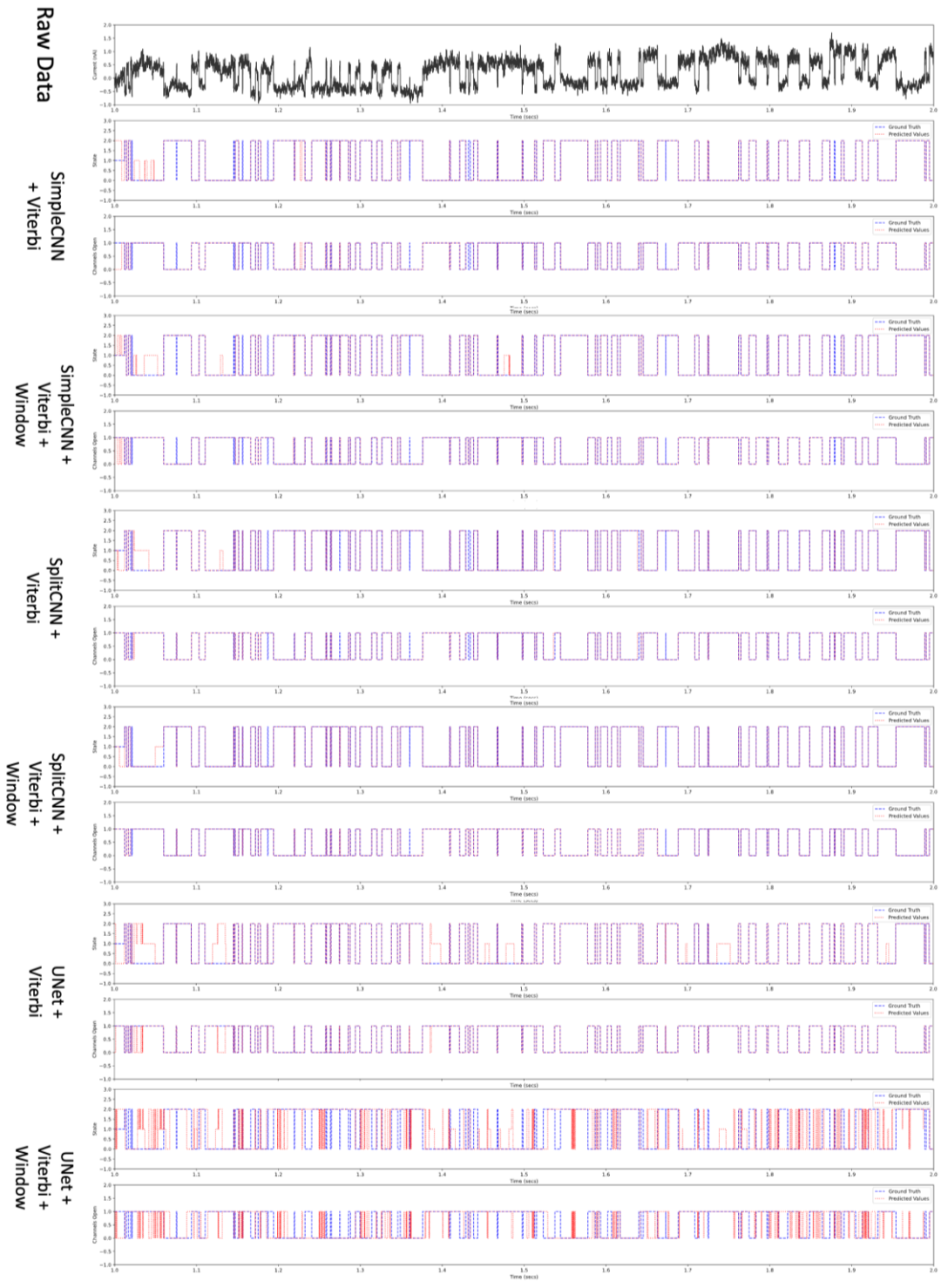


Figure 7.9: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Perturbed Three State Model with Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

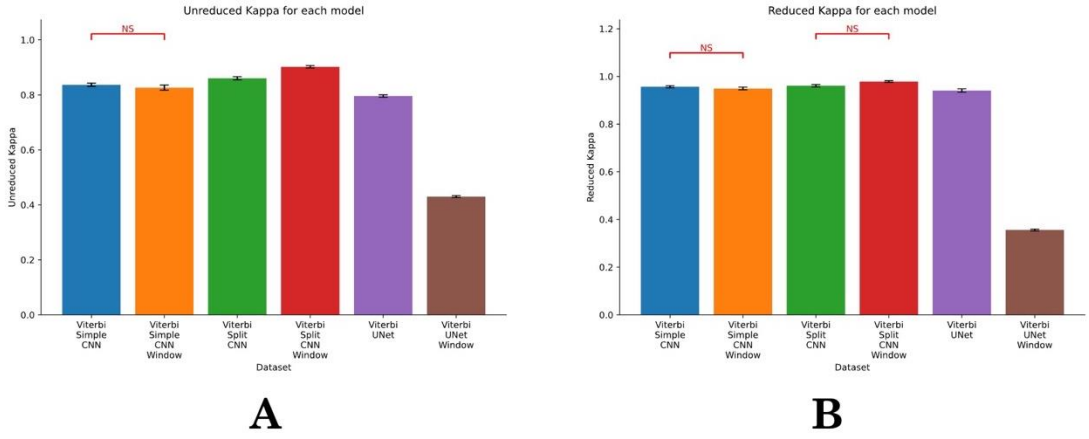


Figure 7.10: Model Training Metrics for "Perturbed Three State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

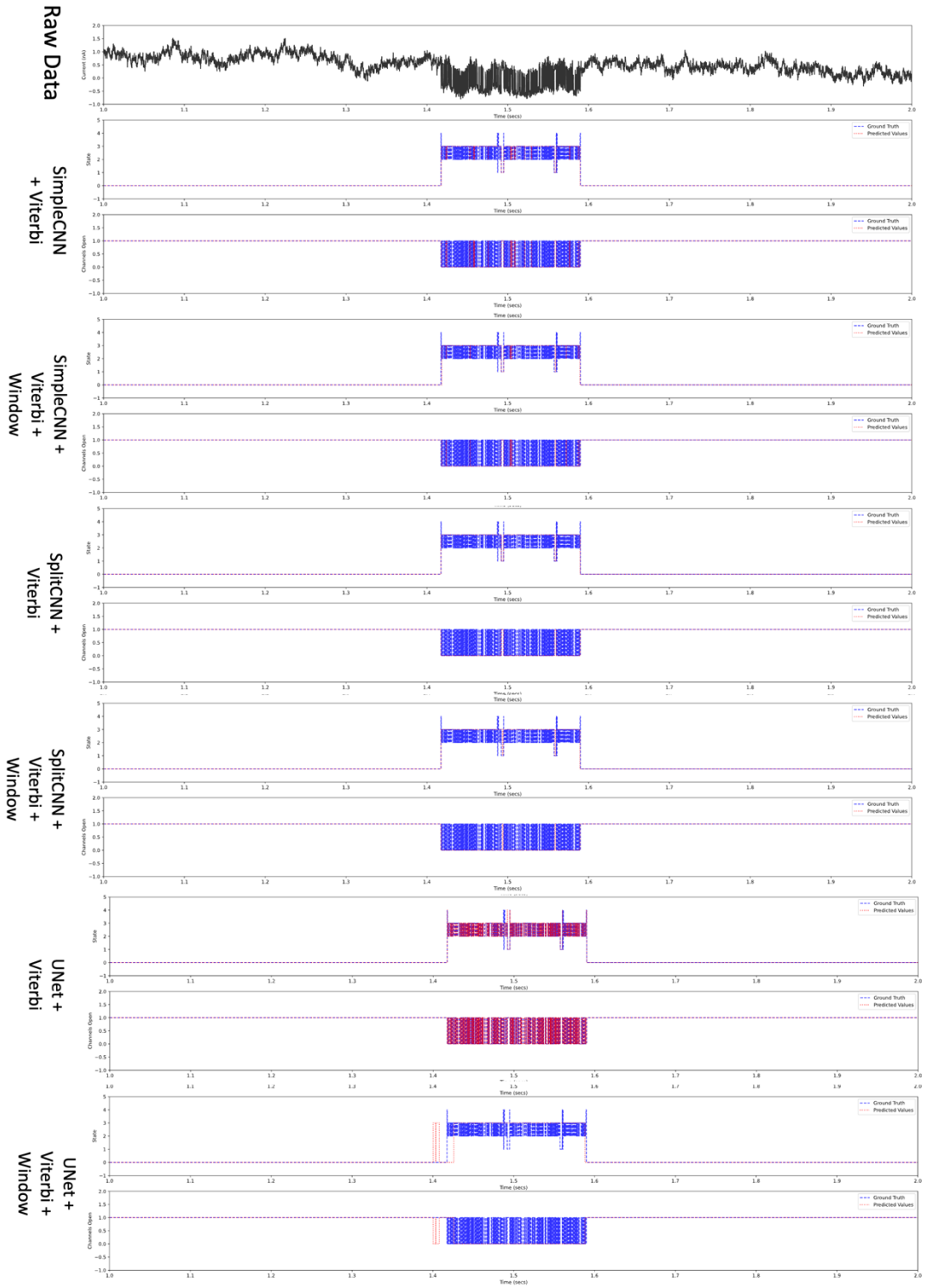


Figure 7.11: Sample Data Traces (15) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Static Five State Model without Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

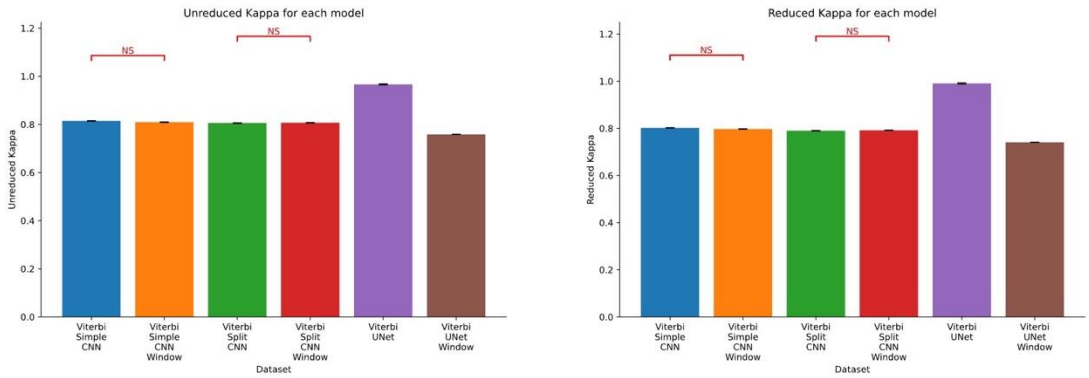


Figure 7.12: Model Training Metrics "Static Five State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

7.3.2 Static Five State Model without Drift

As in previous chapters this Markovian model produces flickery bursts which is a challenging idealisation task. Visual inspection of the raw data in Figure 7.11 shows that there are distinct differences with the progressive windowing U-Net; apparently false events detected outside of the clear burst, but apparently better performance within the bursts. However, in contrast, quantitative analysis by Cohen's Kappa reveals that the model is significantly worse for both Markovian recovery over all (Figure 7.12) and open/closed state recovery (Cohen's Kappa, reduced, Figure 7.12).

7.3.3 Static Five State Model with Drift

Interestingly, here, as with the equivalent data in Chapter 6, U-Net failed entirely as assessed by Cohen's Kappa (Figures 7.13). There is no statistically significant improvement of Cohen's Kappa with this adaptation applied to the SimpleCNN or SplitCNN either. However, visual inspection of the raw records, for SplitCNN with and without windowing, for example, does appear to show large areas of the record for which prediction has improved (Figure 7.14), but clearly this is surpassed by the additional errors in other areas.

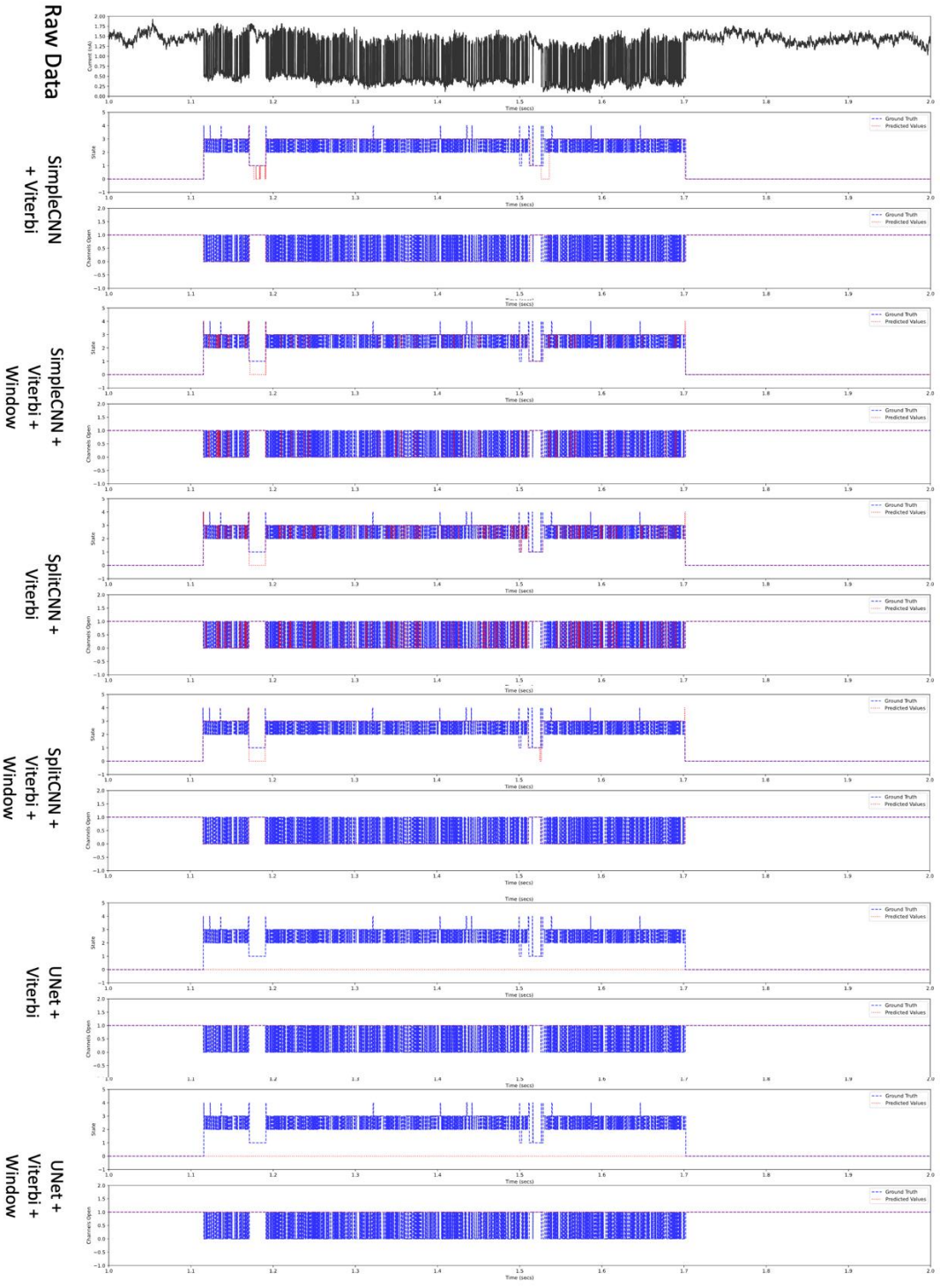


Figure 7.13: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Static Five State Model with Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

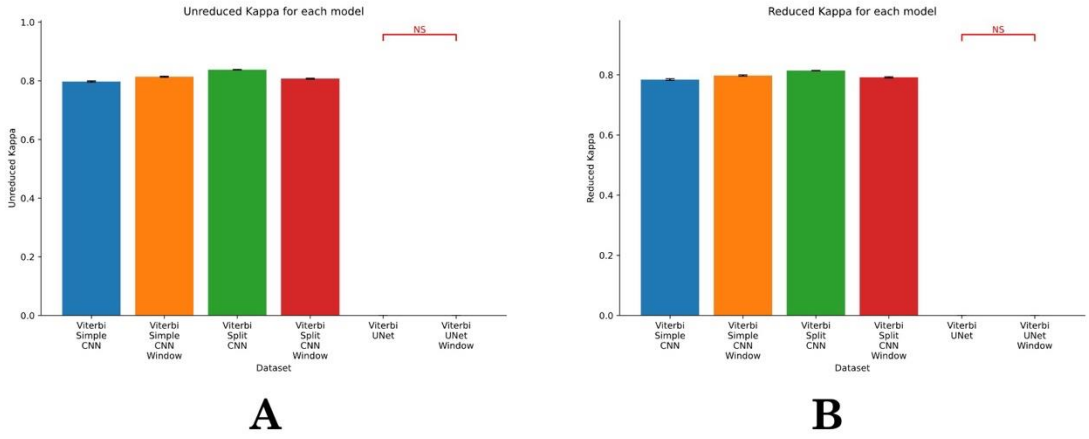


Figure 7.14: Model Training Metrics for "Static Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

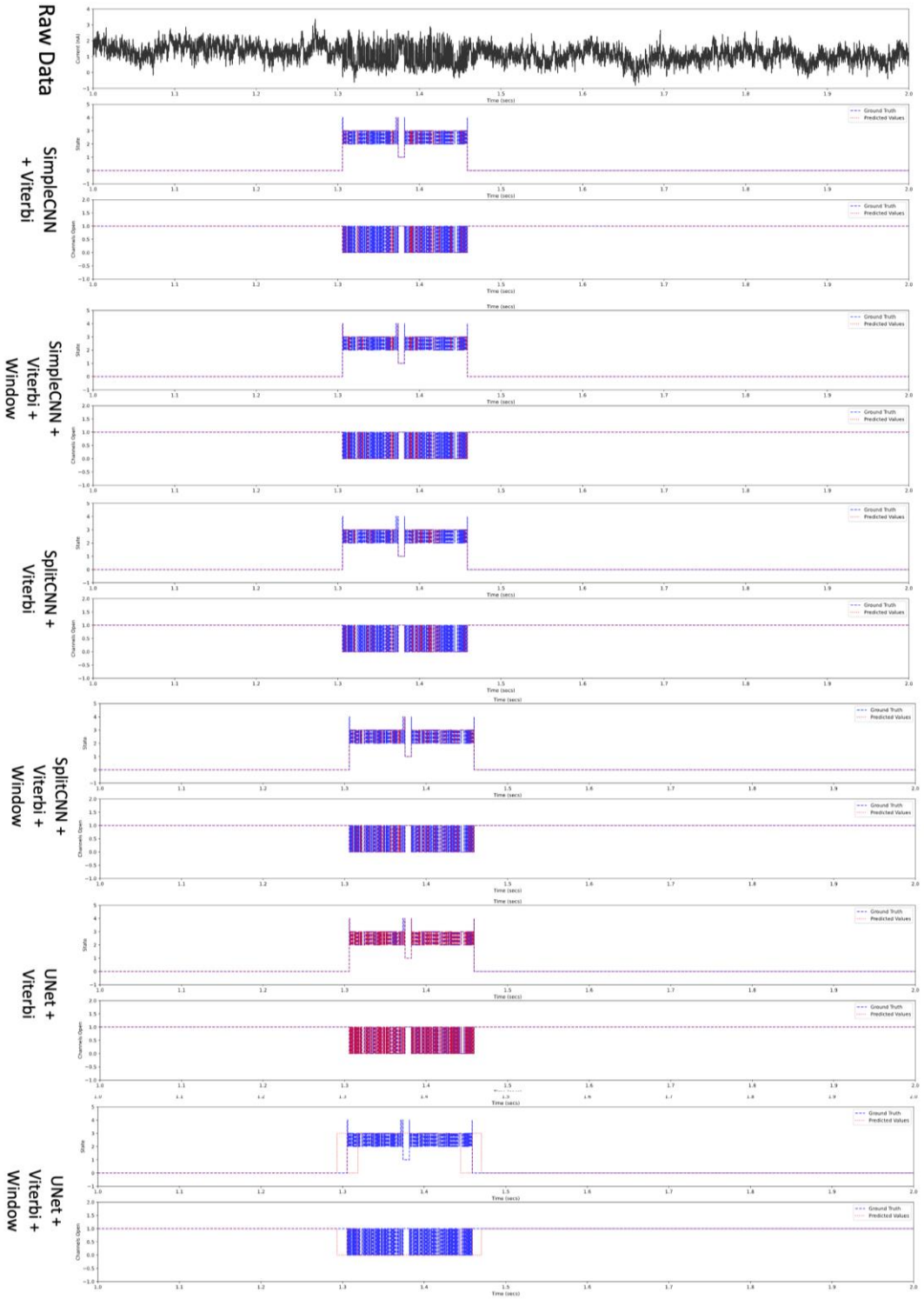


Figure 7.15: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Perturbed Five State Model without Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

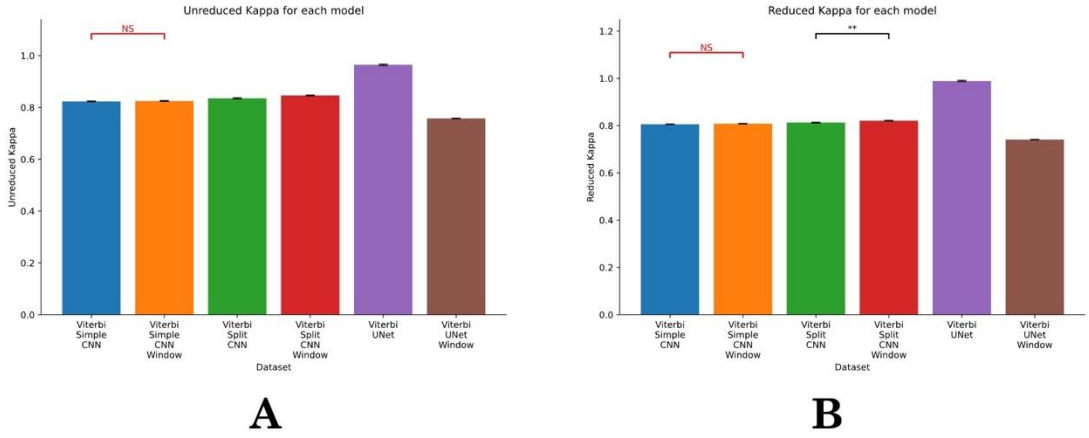


Figure 7.16: Model Training Metrics for "Perturbed Five State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

7.3.4 Perturbed Five State Model without Drift

Raw data shown in Figure 7.15. All the models performed reasonably well with this dataset with Cohen's kappa 0.8 or above (Figure 7.16 exception; windowed U-Net). Simple and SplitCNN models show a statistically significant improvement with windowing, but windowed U-Net is considerably worse (Figure 7.16). Note that despite the improvements gained by progressive windowing the simple and SplitCNN do not perform as well as the sequential (conventionally) windowed Viterbi labelled U-Net model first presented in Chapter 6 (on this dataset). The *Perturbed Five State Model With Drift* results (Figures 7.17, 7.18) are exactly analogous to those without drift above (Figure 7.16) except that here the deterioration of U-Net performance by progressive windowing is catastrophic with modal collapse, with performance no better than random chance.

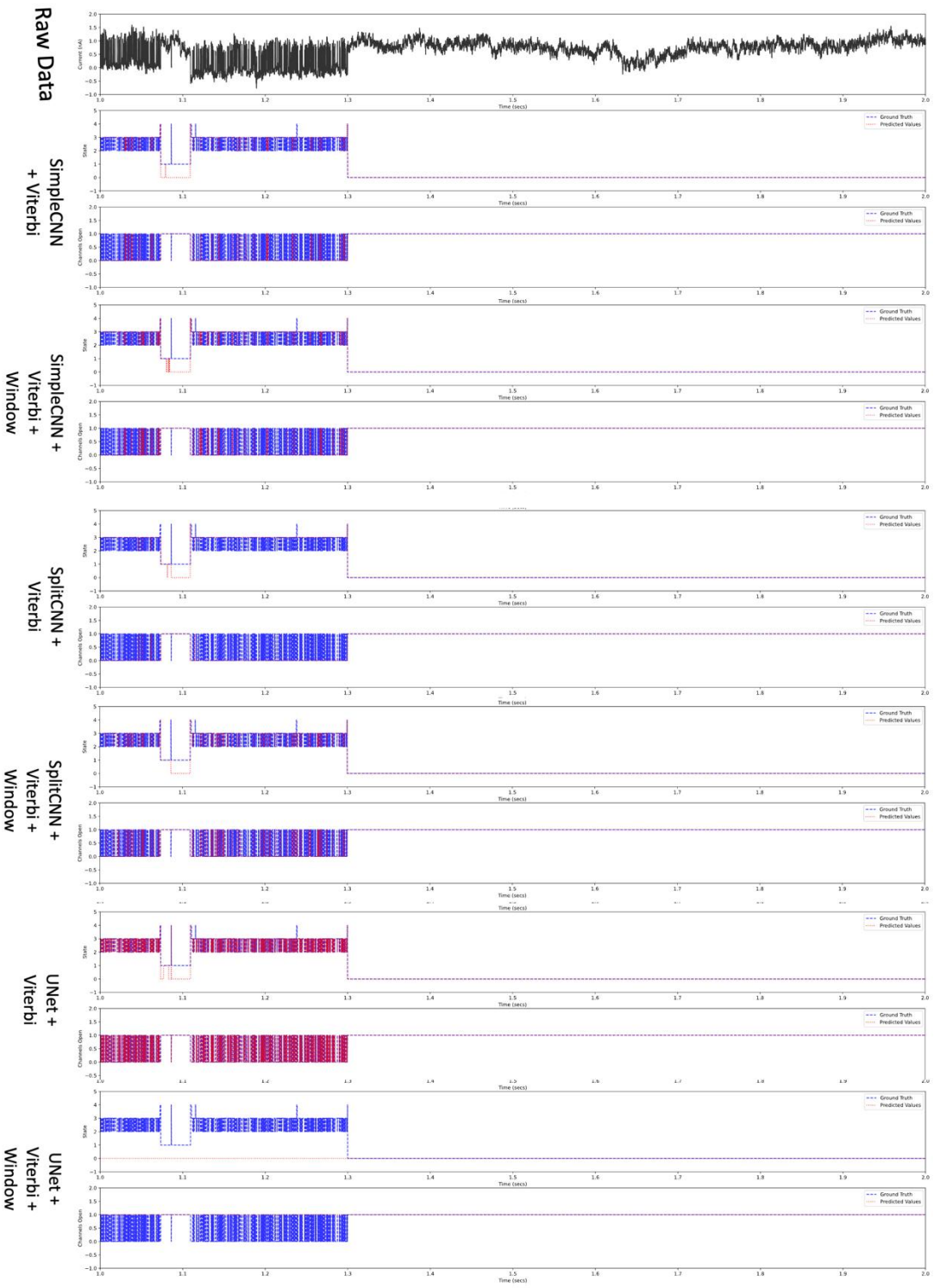


Figure 7.17: Sample Data Traces (1s) of an input simulated ion channel signal (leftmost, black), with the Markovian recovery prediction (left) and channel prediction (right) for each model; for the "Perturbed Five State Model with Drift" dataset.

Here we see the pairwise comparison of our top three models along with their "progressive windowing" pre-processed counterparts. We see subjectively that the "SimpleCNN" model performs slightly worse, introducing artefacts to the idealization, and the "SplitCNN" model slightly better, with slightly fewer artefacts. Perhaps most strikingly, the U-Net model is far worse, perhaps due to the progressive windowing preprocessing needing additional changes to the model architecture to work.

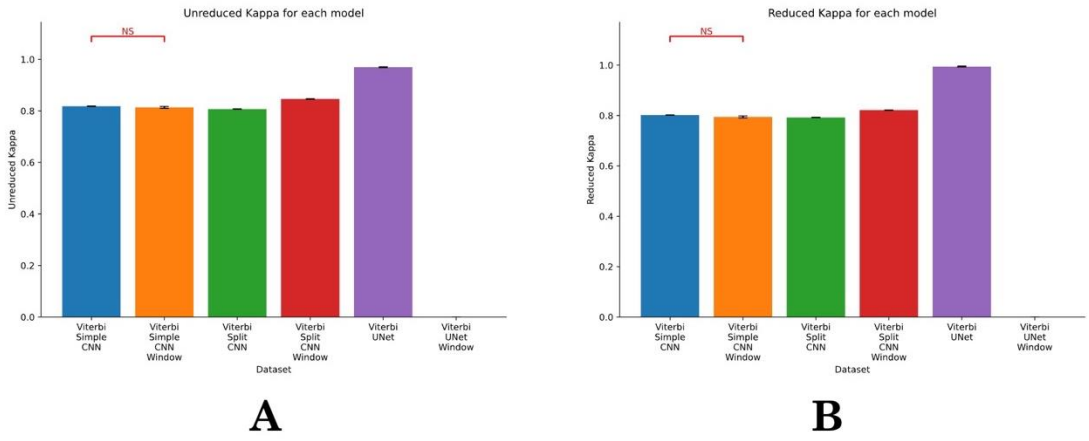


Figure 7.18: Model Training Metrics for "Perturbed Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

7.3.5 Training Progress

In terms of the training progress itself, training and validation metrics are presented in Figures 7.19, 7.20. We see some interesting trends; the progressive windowed SimpleCNN and SplitCNN models outperform their previous counterparts in *training*, but the validation sets are outperformed by the older SplitCNN model; suggesting that the smaller model (SimpleCNN) is overfitting the training data. The U-Net model does not perform well with the progressive windowing, with the performance significantly lower than the full-sized model at all times during training.

Training times (Figure 7.21) show that the window models are faster to train than the full sized models; however from the training progress we can see that the models train for approximately the same number of epochs before reaching a plateau in training and stopping. The reduction in training time is therefore likely caused by the smaller model size; since most of the parameters are in the final dense layer, reducing the size of the output window in the SplitCNN model on the 3-state datasets reduces the number of trainable parameters from 17 million to 9 million.

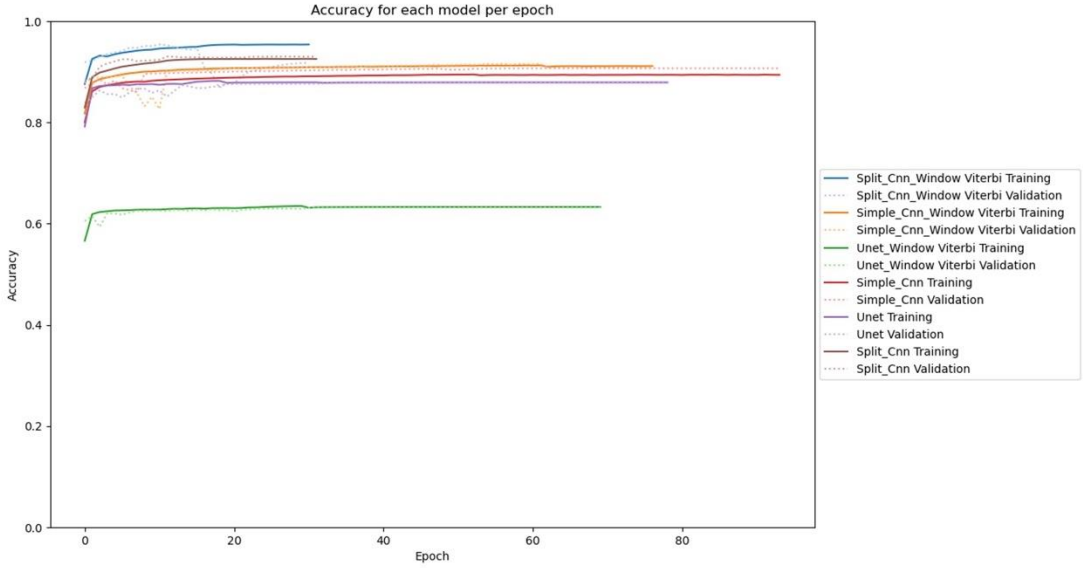


Figure 7.19: Comparison of the Accuracy for each epoch over the training duration.
In addition to training the models, we tracked their performance over time throughout the training process. In the Split_CNN and Simple_CNN case, the windowing shortens training time and increases model performance during training, however for the UNet model the opposite is observed.

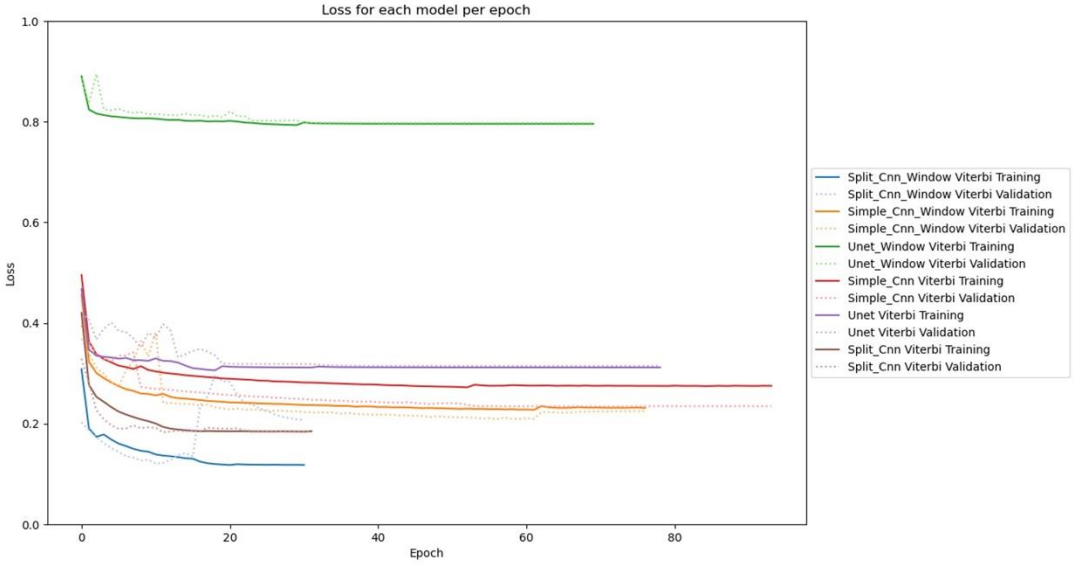


Figure 7.20: Comparison of the Loss for each epoch over the training duration.
We also track the categorical cross-entroopy loss for each epoch. In the Split_CNN and Simple_CNN case, the windowing shortens training time and increases model performance during training, however for the UNet model the opposite is observed.

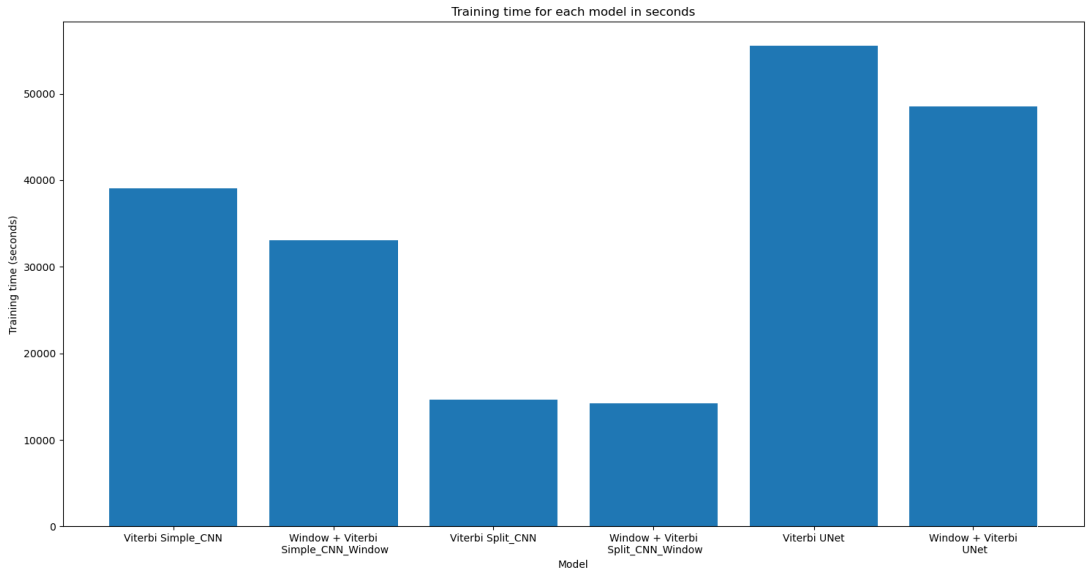


Figure 7.21: Training times for windowed and non-windowed models in seconds
Here we see that there is a slight improvement of the training times of the windowed models compared to the non windowed models. As the number of epochs is approximately the same for each pair of models, we can conclude this is likely caused by the reduced number of parameters from the smaller output size.

7.4 Discussion

In this chapter, we focussed on the theoretical and observed performance drop-off at detection window edges, perhaps resulting from lack of contextual (Markov state) information available for models to infer Markovian state. We adapted the models to take "whole" windows as input, but output only window centres as predictions. Assembling full annotations by using two passes through the datasets. This resulted in much faster detection times, and a small, but significant improvement in performance of some models, but notably worse performance of others (U-Net in particular).

Perhaps the biggest point of discussion is whether the performance gains we saw here with the progressive windowing justify taking these models forward in future. On the one-hand the models are smaller (useful in terms of future deployment) and run faster due to the reduced parameter numbers. Also, the law of marginal gains means that even small improvement at each stage of an analysis pipeline could end-up being valuable over-all. But against this, the most successful model to date has been the U-Net achieving an

impressive 0.9663 Kappa score on Markovian recovery in some tests, and 0.9941 on the reduced condition of simple open/closed idealisation. Noting that Markovian recovery by deep learning is entirely new and it is the idealisation task that is currently the primary tool of single channel analysis (Celik et al., 2020; Davies et al., 2010; Feetham et al., 2015; Nicolai & Sachs, 2013; Numata, Sato-Numata, & Yoshino, 2021). However, U-Net was the least reliable here and so clearly the progressive windowed U-Net model itself would not be a useful model to take forward from this work.

Following on from this, an obvious question is *why did U-Net fail here so badly, in the face of otherwise improved performance?* The answer to this question is likely to relate to the obligatory changes made to adapt the model to our use and, so it expected. The nature of this model modification turned out to undermine the design philosophy behind U-Net itself, where the architecture had been specifically designed for their one-to-one segmentation scenario (Ronneberger et al., 2015). As described in the methods (Chapter 7.2) the original U-Net model includes inverse convolutional layers, the last of which had to be reduced in size to match the smaller prediction window. Clearly, it turns out that this layer was often critical to performance. This raises the final point as to whether there could be other approaches to the same problem. We tackled the window edge problem by a direct deep learning approach, adapting the models to smaller output predictions and using two passes of the data. An alternative method that could be investigated in the future would be an entirely post processing approach. One could use the identical models to those used in previous chapters and again use two passes of the data, staggered by half a window. The first pass (A) would output full predictions from windows starting at 0, 1024, 2048 *et cetera* and the second pass (B) would output full predictions from 512, 1536, 2560 *et cetera*. Post processing would then simply aggregate A:256-768, B:768-1280, A:1280-1972, B:1972-2304 *et cetera*. This would be less computationally efficient since models would still pass the data twice and be the slower (and memory hungry) *full-sized models*, but it would likely correct window edge errors without damaging the performance of the underlying models themselves.

Another way to take this work forward would be in training a bespoke (unique for the case) model for Markov model recovery when a Markov schema is known a priori, and the user wants seamless Markovian state recovery. The method detailed here within would allow a model to be quickly trained on synthetic data, and then used to predict the user data, with said model cached to speed up the process in the future. This would give a one-to-one relationship with the users initial labelling of states at the cost of training a new model for each Markovian schema. In this case, training time is the performance bottleneck and so reducing this cost is of particular interest.

In conclusion we found that adapting the Simple and SplitCNN convolutional models gave small improvements to detection accuracy, and training time, but reduced reliability for U-Net. Therefore the Window Simple and Window Split CNN models were carried forward to the final chapter, in addition to the previously tested UNet model without windowing.

8 Ion Channel Idealisation using Deep Learning Methods

8.1 Introduction

In the work preceding this chapter, ion channel record generation using deep learning, and ion channel Markovian state recovering using deep learning were examined. However, arguably the most important problem to be solved regarding ion channel analysis using deep learning is the problem of channel idealisation. In the Markovian work, we saw that on simulated data, models could accurately recover the Markovian state, and by extension, the number of channels open at a given time from the raw signal record; however this comes with a few caveats; firstly, the data used for both training *and* testing is synthetically generated via a Markovian simulation. This is necessary to obtain an accurate point-by-point ground truth for the underlying, stochastic Markovian state; it is simply not possible to obtain this ground truth for lab-recorded data (indeed; this problem still exists somewhat for channel idealisation however we can mitigate this problem in several ways, such as using transfer learning methods to use a small amount of hand labelled data to supplement a model previously trained on a large amount of synthetic data). Secondly, all the datasets used in training and testing for the Markovian work were single channel; it is not realistic to make this assumption when applying these models to real, lab recorded data; very often a patch will include multiple channels at multiple conductance levels, meaning a simple binary classifier is inappropriate. Therefore, in this chapter, we look at building multichannel classifiers focused on channel idealisation only.

Measuring the efficacy of these models is significantly difficult; the lack of ground truth poses a similar problem as in the Markovian problem; we simply cannot know for certain the ion channel's configuration at each time point. However, we know from existing work that ion channel behaviour can be influenced via changes in the conditions; so we can use

summary statistics from the model output along with existing knowledge about the effects of substances on a channel to infer if a model is accurately detecting these outcomes; for example, although we cannot get a point-for-point labelling of an ion channel recording when a channel inhibitor is applied, we know that overall the number of openings should decrease. It is known that substances can affect the channel size, opening probability and/or the dwell time distributions of a channel depending on the substance and channel involved. Therefore, if we select a substance with known ion channel modulating properties we can test our deep learning models to see if they identify these known actions. If our deep learning model can identify such effects with similar sensitivity to existing methods, then the model has been successful.

Another significant challenge of evaluating deep learning models on real ion channel data, as opposed to simulated, is that real data contains artefacts that may affect the deep learning process. For example, MinMax scaling (the scaling used in previous work), is extremely sensitive to outliers and anomalies in the data; something which is not uncommon in ion channel recordings and often manually removed by a researcher. Previous work into automatic ion channel idealisation (Celik et al., 2020; Gnanasambandam et al., 2017; Hotz et al., 2013; Qin, 2004) has not approached this problem, so a novel solution must be developed. Furthermore, in all this previous work, the number of channels visible in the test set is always known *a priori*; resulting in a similar channel size for each dataset. Due to the nature of patch-clamp electrophysiology, we cannot guarantee the number of channels recorded in the patch clamp process; therefore a model trained on single channel data may be given data with several channels to idealise and fail. A novel approach is yet to be developed for this problem as well, and needs to be approached for models to work on real data.

For this chapter we use calcium activated potassium channels (KCa1.1, or BK for "big potassium") stably expressed in the HEK cell line since it is a well-studied system with commercial availability. BK channels are characterised by their conductance (1100-300pS) (Lee & Cui, 2010), far greater than other potassium channels, but smaller than

those used in similar work for automatic ion channel idealisation (Gnanasambandam et al., 2017), so provide a difficulty for deep learning models in line with previous work. Like all ion channels, BK have fundamentally important biological functions, for example, dysregulation of BK channels has shown to be a factor in a number of conditions from hypertension (Brenner et al., 2000) to epilepsy (Du et al., 2005) and autism (Laumonnier et al., 2006).

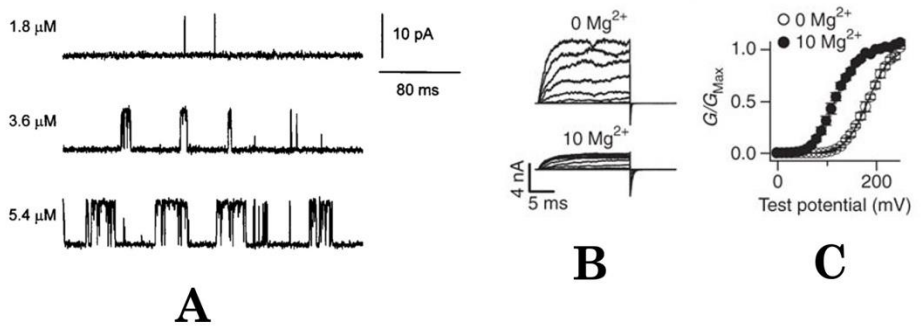


Figure 8.1: BK Channel sensitivity to changes in ion concentration, adapted from (Nimigean & Magleby, 1999; Yang et al., 2008)
(A) shows three samples of recordings from a BK channel in HEK293 cells at different calcium concentrations, recorded at 30V. Ca2+ is shown to increase Po, but has no effect on channel size. B and C show differences in channel size when the same channel is exposed to changes in Mg2+ concentration; B shows representative examples of the channel at 0 and 10mM of Mg2+ and C shows the voltage amplitude curve for the same set of conditions.

Regulation of the BK channel is complex involving, organic ligands, phosphorylation, voltage and its defining physiological feature of sensitivity to intracellular calcium ions (Horrigan & Aldrich, 2002). Elevation of calcium ion concentrations at the cytosolic surface leads to increasing open probability (Figure 8.1). The effect of individual ions is not restricted to open probability, for example the presence of magnesium ions shifts the activation voltage of BK channels to the left (Figure 8.1) (Zamoyski, Serebryakov, & Schubert, 1989) . In addition to individual ions, many more substances can effect BK channel activity. Interestingly it has been shown that carbon monoxide increases BK

channel size (Bae, Kim, & Lim, 2021) causing arrythmia, and myocardial cell deaths, potentially leading to cardiac fibrosis.

A well-known example of pharmacological modulation of BK is that of the injectable anaesthetic compound ketamine. In addition to its interaction with calcium ion channels it also reduces the open probability the BK channel (Denson, Duchatelle, & Eaton, 1994); BK channel inhibition is known to modulate neurotransmitter release (Denson et al., 1994; Denson & Eaton, 1994; Yamakura et al., 2000), causing anaesthetic effects.

Usefully there are also a range of commercially available ligands which can be routinely used as tools to probe the functions of BK. For example, iberiotoxin, the toxin from the scorpion *Buthus tamulus* that inhibits the current through BK channels, causing both slowing of the mean dwell times and reduction of the open probability of the channel (Candia et al., 1992) quite specifically and at very low concentrations. Of particular pertinence to this chapter are two further drugs; Penitrem A, a commonly used toxin produced by the spore *Aspergillus Claviceps* for the study of BK channels as it is widely available and is a BK antagonist in both intra- and extracellular applications (unlike iberiotoxin which only works extracellularly) (Asano et al., 2012). The well-known nature of Penitrem A's effect on BK channels allows us a baseline to compare against when evaluating models; a robust model should be able to detect the blocking action of the toxin on a BK channel across a number of conditions (Figure 8.2).

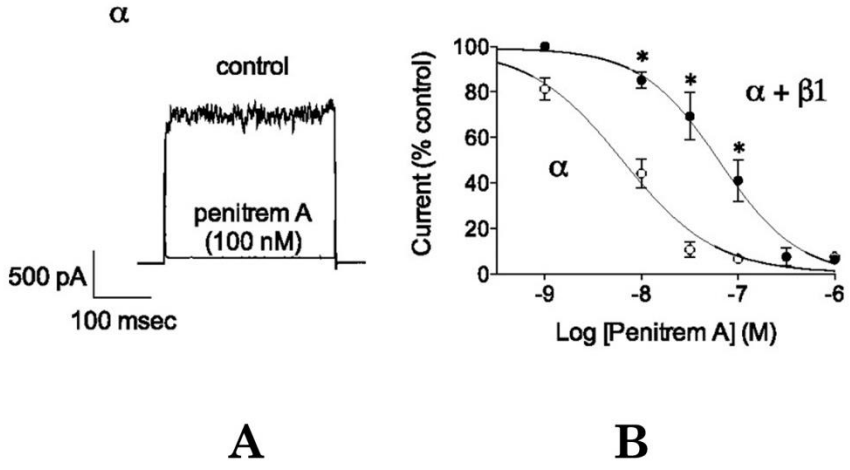


Figure 8.2: Response of BK channel to Penitrem A, Adapted from (Asano et al., 2012).
Penitrem is known to reduce channel activity for BK channels, and is more versatile and available than its analogues such as iberiotoxin. A shows a representative example of the signal with and without Penitrem A, and B shows the response of the current reading as the Penitrem A concentration is increased for two different phenotypes of the channel (α, $\alpha + \beta 1$).

The second drug; Vernakalant Hydrochloride, is of interest to us as while the effects on other, similar potassium channels are well known (Burashnikov et al., 2012; Naccarelli et al., 2008; Seyler et al., 2014), there has been little study on the effects of the drug on BK channels specifically. The drug is typically used in the treatment of atrial fibrillation; a condition of which BK channels are known to play a part (Jakob et al., 2021), so studying the effects of this drug on BK channels is a worthwhile in its own right in addition to model validation.

8.2 Aims and Objectives

This Chapter encapsulates 3 stand-alone experiments. Each one was recorded by a different experimenter (including one by myself) on a different set of equipment, with different acquisition hardware and different mode of patch clamp recording to contribute to generalisability of our results.

8.2.1 Ca2+ sensitivity of BK channels.

In this experiment, we start from a knowledge that calcium activated potassium channels (BK channels) exhibit an increase of open probability as the concentration of extracellular

calcium increases. We idealise a number of different ion channel recordings at different calcium concentrations, and from these idealisations calculate the open probabilities. A successful deep learning model should be able to generate a set of open probabilities that fits known open probability dose response curves.

8.2.1.1 Specific objectives:

A. Characterise BK channels, and their sensitivity to voltage and Ca2+ ions with a deep-learning based approach, with new, novel models trained on relevant datasets.

B. Compare traditional analysis via computer assisted idealisation through QuB of the inside out patch data with analysis using a deep learning approach, as well as secondary analysis from these idealisations such as dwell time histogram analysis.

C. Determine whether our deep-learning approach can detect Ca2+ activation with greater sensitivity to a traditional idealisation approach.

8.2.2 Penitrem A sensitivity of BK channels.

In this experiment, we again start from knowledge that BK channels are inhibited by the concentration of extracellular Penitrem A. In this case, an independent analysis and idealisation of outside-out patch clamp recordings was performed; a deep learning model should not only detect a decrease in open probability as the concentration of Penitrem A increases, but ideally have a stronger discrimination than the manual analysis.

8.2.2.1 Specific objectives:

D. Measure BK channels' sensitivity to Penitrem A using a deep learning approach,

E. Compare traditional (idealisation) analysis of outside-out patch data with a deep-learning approach by considering the differences and similarities between each approach's resulting analysis for open probabilities and dwell times.

F. Determine whether our deep-learning approach can detect Penitrem inhibition of BK with greater sensitivity to a traditional idealisation approach.

8.2.3 Vernakalant Hydrochloride sensitivity of BK channels

In this experiment, we start from the knowledge that cardiac domain potassium channels are inhibited by the concentration of extracellular vernakalant hydrochloride, however the effect of the drug on BK channels specifically is unknown. The previous two experiments act as controls for showing the deep learning models can detect a known drug effect, whereas this experiment attempts the inverse; detecting an unknown drug effect assuming the deep learning model works successfully. This approach yields both a new method for ion channel idealisation, as well as the discovery of a previously unknown effect of a drug on an ion channel.

8.2.3.1 Specific objectives:

G: Test whether vernakalant modulates BK channels via outside-out patch clamp recordings, and if so calculate the magnitude and EC50 of the effect using a deep-learning approach.

H: By comparing commonly sought after physiological analysis of ion channel function (dwell time, open probability analysis), compare the performance of the deep learning and traditional, semi-automatic idealisation approach to idealisation of vernakalant BK channel modulation data.

8.3 Methods

Three datasets were used in this work; the “Calcium” and “Penitrem” datasets were generated by members of the group for the purpose of model testing (*see acknowledgements*) and the “Vernakalant” dataset recorded by myself specifically for this work.

8.3.1 Cell Culture

HEK-293 cells were chosen for all experiments due to a strong expression of the BK channel gene and robustness to changes in conditions. They were kindly provided initially by Dr Sean Brennan, who stored and cultured here in on site. For all three datasets, cell culture protocol was the same (Figure 8.3, Table 8.1); cells were stored in a media of Gibco DMEM High Glucose + GLUTAMAX [Thermo Fisher Scientific, USA] with 10% foetal bovine serum (FBS) [Sigma Aldrich, USA] and 1% gentamicin [Sigma Aldrich, USA] ("FBS+" media) in T25 flasks at 80% confluence before splitting and 38C and 5% CO2.

Table 8.1: Cell Culture Solutions.
Two solutions were used for cell culture; since patch clamp recording is significantly difficult in the presence of FBS, a solution without FBS or Gentamicin was used to suspend cells for patch clamping. This solution was also used for washing the flask between spins in the cell culture protocol.

Media Name	FBS-	FBS+
Gibco DMEM High Glucose + GLUTAMAX	100%	89%
FBS	0%	10%
Gentamicin	0%	1%

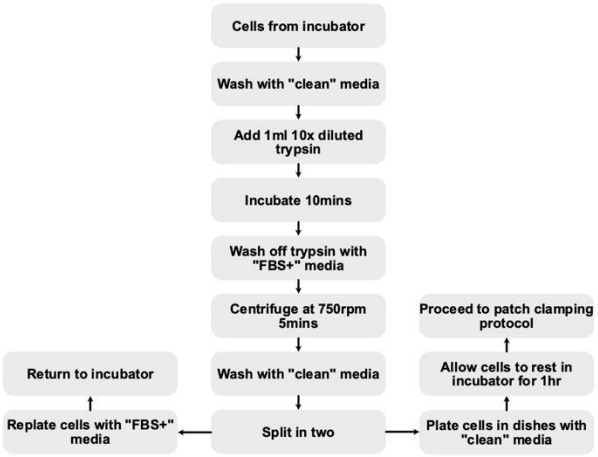


Figure 8.3: Cell Culture protocol.
BK cells were used to test the effects of different substances using our deep learning models. Cells were taken from an incubator at ~80% confluence (about every 2 days) and split using the above protocol. Cells were split into two groups at the end of splitting; one group would be resuspended for the following round of splitting, and the other would be used for experiments.

Cells were split every few days when confluence reached above 80% - this was done by removing media and washing a T25 flask [Thermo Fisher, USA] with 5ml clean media without FBS or gentamicin. 5ml 1% trypsin solution was then added to detach cells from the flask surface, incubating for 10-15mins at 38C 5% CO2 to allow this process to occur.

After cells had lifted from the bottom of the flask, the resulting solution was quenched with the 10ml of the FBS media, then spun down in a centrifuge at 750rpm for 5mins. The pellet was then washed again in the non-FBS media, and at this point two suspensions were created; in one case the pellet was resuspended in 10ml of the FBS media. Two 1ml measures of this suspensions were placed into clean T25 flasks and 5ml of additional FBS media was added, then flasks were placed into the incubator until splitting was required again. In the other case, the pellet was resuspended in non-FBS media, with 1ml of this solution placed into a 5cm glass bottomed petri dish [Matek, USA] and left for 10mins in the same incubation conditions as the other cells to recover ready for patching.

8.3.2 Freezing and Revitalising

Cells were split regularly (between 2-4 days), and a sample of cells were frozen each time to preserve cell stocks. To do so, after passing the cells through the centrifuge, sterile DMSO was added until a final concentration of 10% DMSO to 90% FBS media was achieved, and aliquoted into 1ml cryovials [Thermo Fisher, USA]. Samples were then transported to a -80C freezer for 48 hours before transferring into long term liquid nitrogen storage.

To revitalise the cells, cryovials were taken from the liquid nitrogen storage and immediately placed into a 37C water bath. After thawing, 10ml of pre-warmed FBS media was added into a centrifuge tube along with the previously frozen cells, and span down at 750rpm for 5mins. The FBS + DSMO solution was then decanted out and replaced with clean, FBS positive media into a T25 flask, and placed back into an incubator at 38C 5% CO2 until the desired confluence was required.

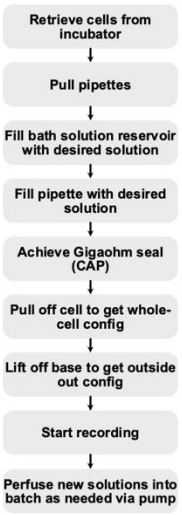


Figure 8.4: General Electrophysiology protocol.
Cells were recorded in the same way each time. Pipettes were pulled to the required resistance (5-8MΩ) and filled with a solution depending on the experiment and recording configuration. The patch clamping bath was filled with another solution, again depending on the experiment. A Gigaohm seal was then achieved followed by the required patch-clamp configuration (e.g. inside out or outside out). Then recording was started to obtain data samples for analysis, perfusing new solutions into the bath to measure drug response.

8.3.3 Recording Data

For patch-clamp recording, a similar protocol was used across all datasets (Figure 8.4); patch-pipettes were pulled using 1.5mm O.D 10cm length borosilicate glass capillary tubes [Intracel, England] and a two-step electrode puller [PP-830, Narishige, Japan] . Patch pipettes were pulled with an aim to achieve initial patch resistance of 5-8MΩ once filled with physiological solution.

8.3.3.1 Solutions

Solutions for each dataset can be seen in Table 8.2 and were used across all experiments; however whether each solution was extracellular or intracellular depended on the patch-clamp configuration being used and is given below. In addition, for the "Penitrem" dataset, an additional 300uM free calcium was added to the bath solution to induce activity. The recording configurations and solution locations can be seen in Table 8.2.

Table 8.2: Solutions used for patch-clamping.
Both solutions were made up using KOH to the desired pH.

Solution	A	B
K+	150mM	150mM
Cl-	154mM	161mM
Mg2+	1mM	2mM
Ca2+	2mM	2mM
HEPES (-1)	10mM	10mM
Na+	0mM	5mM
EGTA (-2)	3.05mM	0mM

Calculated osmolarity	316.05	337.2
pH (with KOH)	7.3	7.4

In the cases where a drug was used, it was always applied extracellularly; to do this, a portion of extracellular solution was aliquoted into a smaller vessel and the correct amount of drug was added to the solution to reach the desired concentration in a Falcon tube [Fisher Scientific, UK]. This solution was then perfused through the bath to achieve the change in conditions without losing the seal via a pump [Minipuls 3, Gilson USA].

8.3.3.2 Making a seal

Critical to the patch clamp technique is careful formation of the giga-ohm seal which allows for resolution of the pico-ampere current jumps exhibited by ion channel activity. This is achieved by a Faraday cage on an air table holding a microscope (for visual positioning of the pipette), micromanipulator [Narishige, Japan] (for precisely lowering the pipette onto the cell surface without rupturing it) and an electronic headstage for conducting the electrical current out of the cage and towards the amplifier. The Faraday cage and air table are crucial to the patch-clamp process, as a slight vibration during recording can cause extreme noise during the recording, or rupture the cell entirely.

When the desired solutions were in the pipette and dish, the pipette was lowered down onto a cell using a micromanipulator until an increase in resistance was observed, corresponding to the pipette tip touching the cell surface. Then, light suction was applied to the pipette to increase the resistance of the seal until no more suction was needed to continue the increase in resistance; at this point suction was completely removed, and resistance would climb steadily to the giga-ohm level.

At this point, the giga-seal had been created and current was being measured across the cell wall, however due to the experimental design, further steps needed to be taken to be able to modulate the extracellular solution without detaching the pipette from the cell membrane (since in this initial cell-attach patch configuration, the extracellular solution was in the pipette, and could not be changed).

For inside-out patch (IOP), the pipette was withdrawn from the cell to isolate a small patch of membrane with cytoplasmic face to the bath. However, for OOP additional suction was applied to the patch to rupture the cell membrane beneath the patch-pipette tip and achieve a whole-cell configuration. At this point, the pipette was withdrawn from the cell dish to pull off a patch of membrane clear of the cell. If a vesicle was apparent the pipette tip was quickly moved through the solution surface.

This then resulted in the pipette solution representing intracellular conditions, and the bath solution representing extracellular conditions, and a fluid pump could be used to perfuse new solutions through the bath, changing the extracellular conditions to be measured. For example, in the case of measuring the effects of calcium concentration, a series of solutions increasing in calcium were perfused into the bath, allowing for the same cell to be recorded under a set of different conditions.

8.3.3.3 Digitisation

Recording would be either be achieved through an amplifier, with either a gain of 200 or 500 times depending on noise and the number of channels present. The 4-pole in build basselfilter was set at either 1kHz or 2kHz depending on the sample rate, with an additional HumBug filter [Quest Scientific, Canada] to remove 50 cycle noise. This would then be passed into a digitiser to convert the signal from analogue to digital at a sample rate of either 10kHz or 250kHz depending on the dataset.

8.3.4 Real Data Datasets

If different voltage levels were desired, the voltage would be changed throughout the recording via the patch clamp software. In some cases a patch was robust enough to record over a series of both drug levels and voltage levels; in this case the voltages were cycled through first, then a drug applied. Data was recorded into WinEDR; the recording parameters for each of the datasets can be seen in Table 8.3.

Table 8.3: Dataset information for each dataset
Each dataset was recorded under slightly different conditions, examining a different drug or ion's effect on the BK channel.

Dataset Name	Sample Rate	Set-up	Variable Drug/Ion	Recording configuration
"Calcium"	10kHz	Microscope: Eclipse E600FN, Nikon, Japan Amplifier: Axopatch 200B, Axon Instruments, USA Headstage: CV 203BU, Axon Instruments, USA Digitiser: Digidata 1200, Axon Instruments, USA	Ca2+	IOP
"Penitrem"	250kHz	Microscope: Eclipse Ti, Nikon, Japan	Penitrem A	IOP

		Amplifier: Axopatch 200B, Axon Instruments, USA Headstage: CV 203BU, Axon Instruments, USA Digitiser: Digidata 1550, Axon Instruments, USA		
"Vernakalant"	10kHz	Microscope: CK2, Olympus. Japan Amplifier: Axopatch 200A, Axon Instruments, USA Headstage: CY 201A, Axon Instruments, USA Digitiser: CED MICRO3 1401, Cambridge Electronic Design, England	Vernakalant Hydrochloride	OOP

8.3.5 Data Pre-processing

Once the data had been digitised as above, into WinEDR binary format, it was converted to text comma separated value (CSV) format first split by condition such that each file only contained one voltage level and drug level. For the "Calcium" and "Vernakalant" datasets, QuB idealisations were performed to ascertain a baseline performance of what is currently possible with modern methods, and for the "Penitrem" dataset, idealisations and analysis were carried out by Dr Sean Brennan.

Several automatic processes were used to clean the data before being analysed with the deep learning models (Figure 8.5) to ensure the deep learning model could accurately idealise the data.

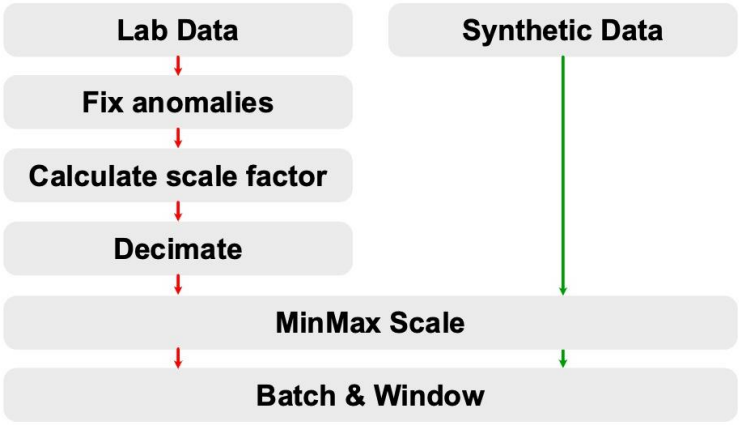


Figure 8.5: Preprocessing Pipelines.
The preprocessing pipelines were slightly different for the synthetic training data and the real, lab recorded data – due to the nature of the simulation, the synthetic data was far more ideal than seen in the lab; the synthetic data had no spiking anomalies and had a fixed number of channels present in each file, meaning the MinMax scaling was consistent. The synthetic data's events happened almost instantaneously, giving a quick change in current; whereas in the real data due to filtering and the realistic nature of the recording, the change in current for events was far slower.

8.3.5.1 Anomaly detection (deep-learning methods only).

Initially, an anomaly detector and remover was built to remove peak artefacts from the data (Figure 8.6). These artefacts could potentially be many times greater in amplitude

than a single channel event and interfere with any scaling operation used, and developing an intelligent method for handling these events is a crucial step, without which models may fail. This was achieved by first detecting anomalies; the data was split into 20 bins, and within each bin any point more than an arbitrary 4 standard deviations (0.0001% exclusion) from the mean would be considered anomalous. To increase the effectiveness of the anomaly detection, a simple propagation algorithm was used to include any datapoints part of a peak (for an upward peak artefact, any other points that were strictly increasing towards the peak's centre) that were under the 4 standard deviation threshold.

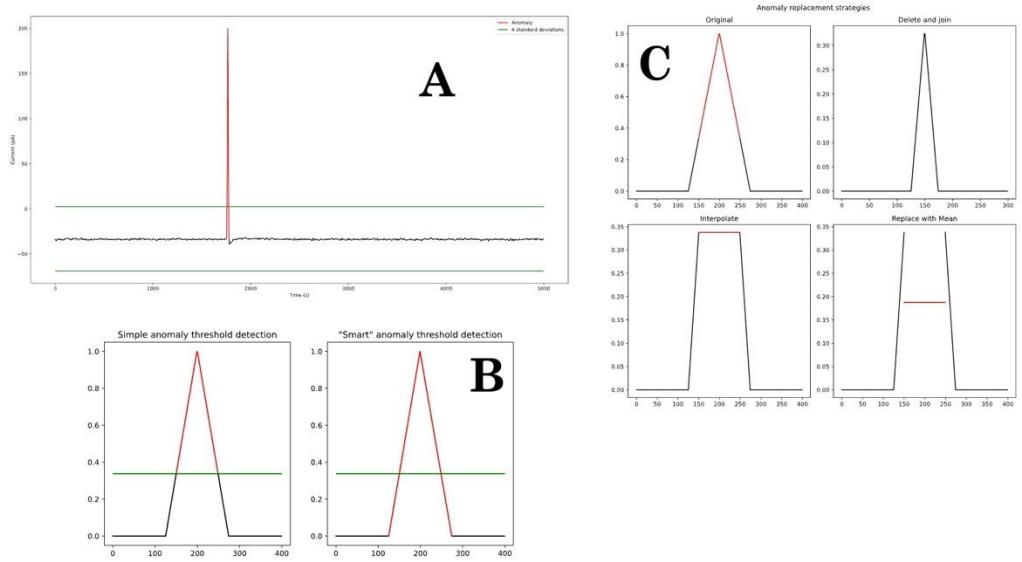


Figure 8.6: Anomaly Detection.
A shows an example of an anomaly along with the threshold (4 standard deviations) used in the preprocessing timeline. All points further than 4 standard deviations away from the mean were labelled as anomalous – B shows two different approaches to inclusion of anomalous points; the "simple" anomaly detection takes only the points above the threshold – however the "smart" algorithm includes all strictly decreasing points around the anomaly to label the base of the spike as well. In C we see the 3 different approaches to anomaly removal – we can either delete the anomaly completely and join the points together (this is what is used in the work); interpolate the missing data with a linear fit; or replace the anomaly with the mean. During training these latter two methods were found to lead to worse model idealizations so were avoided.

We tested several strategies to removing detected anomalies; simply deleting the anomaly, shortening the record; replacing the anomaly with the mean current value across the whole file; or interpolating the points before and after the anomaly linearly (Figure 8,6). In preliminary testing, it was found that simply deleting the anomaly was

the only feasible method, as the other two approaches introduced significant inaccuracies in the deep learning idealisation that resulted in poor performance for further analysis.

8.3.5.2 Timeseries Amplitude Normalisation via Estimation (TANE)

Due to the nature of ion channel activity and scaling functions such as *MinMax*, the size of the channels after scaling would depend on the number of channels present in the input signal. Since the training dataset consistently had five channels present, for lab-recorded signals with less than five channels present, a consideration had to be applied to scale the recording such that the mean channel size was equal to the channel size present in the traning dataset (Figure 8.7). In this work we use Timeseries Amplitude Normalisation via Estimation (TANE) where we estimate the number of channels present in the data by splitting it into 10 windows (to mitigate the factor of baseline drift) and construct amplitude histograms for each one. A simple peak-detector (scipy.signal) is used to compute the number of peaks in these histograms, which when averaged should correspond to the number of conductance levels in the data. Alternatively, this number can be entered manually by the user (Manual-TANE). We then divide the amplitude data pointwise by a factor inversely proportional to the number of channels observed; this would result in a fixed unitary conductance or channel size post scaling.

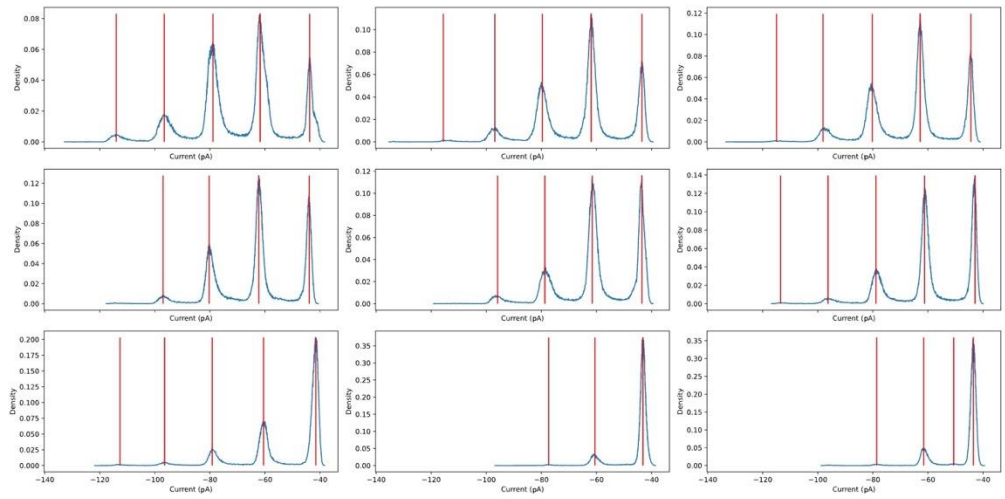


Figure 8.7: TANE Scale factor detection.
Since the model trained on data that always contained 5 channels, the minmaxing process would lead to a synthetic discrepancy between the mean size of channels between the training and real lab recorded dataset if the lab recorded data had less than 5 channels. Therefore, for every recording, the data was split into 20 windows and the amplitude histograms calculated (first 9 shown here) . The histograms were then run through a peak fitting algorithm to detect the number of channels in each segment – the mean of which across all windows would be taken as the detected number of channels, rounded up.

Without such a correction, idealisation would either fail completely, or return the incorrect number of channels, skewing downstream statistics. Application of this pre-processing step appeared to correct this issue, with the option of manual supervision to ensure the detected number of channels was correct by giving the user the predicted number of observed channels and allowing a manual override if the estimation was deemed incorrect.

8.3.5.3 Time-domain scaling

In the final step, a basic decimation algorithm was used on the signals; this algorithm simply took each *n*th point, where *n* was the decimation parameter (Figure 8.8) and discarded the rest. This was in part to sharpen event edges to increase model performance,

and reduce input data size, and offered an additional control to bring the number of points in an event more in line with that seen in the training data.

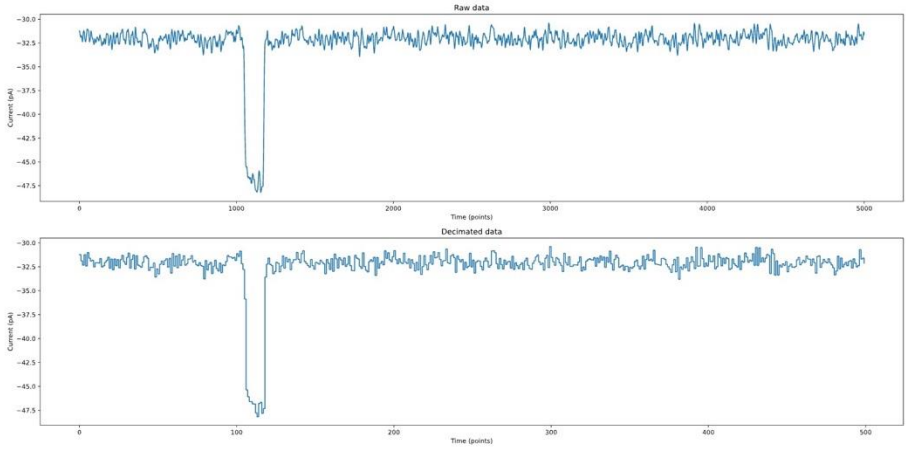


Figure 8.8: Data decimation (downsampling).
A simple decimation process was applied to the recording data where each nth point was taken of the signal, with n being the decimation factor (here 10). This was important for 3 reasons, particularly for the "Penitrem" dataset. Firstly, the "Penitrem" dataset had an extremely high sample rate, meaning that the events (whilst having a similar size by time) had vastly more points than the training set, causing the inputs to the model to appear squashed on the time axis. By decimating this, we corrected this problem. Secondly – it improved model performance on high sample rate data – by using the decimation process we effectively cut the prediction time by a factor of 10. Lastly, and perhaps most importantly, is that due to the filtering of the data through the hardware - the events had a much "softer edges" (by number of points); decimation "sharpens these edges by "speeding up" the change in current.

These pre-processing steps were in addition to the pre-processing steps seen in Chapter 7, with the above steps added where appropriate.

8.3.6 Deep Learning

8.3.6.1 Data simulation

The work on Markovian models has been limited to single-channel data, so new models had to be re-trained for the idealisation of multichannel data. Previous work, such as the original DeepChannel (Celik et al., 2020) models were already tested on multichannel data, so forms a baseline for a state-of-the-art comparison. To train multichannel models, the same model architectures from Chapter 7 were used, but with a new dataset, extending the previous simulations to allow for multi-channel simulation.

8.3.6.2 Deep Learning Models

The initial models tested for multi-channel idealisations were the three best models from the previous chapter: a Split CNN model modelled from similar work (Supratak et al., 2017) with a window truncation method detailed in Chapter 7; a "Simple" CNN model with a window truncation method, serving as a low-parameter baseline to compare similar models against; and the state-of-the-art UNet model applied to signal segmentation rather than image segmentation (also with a window truncation method). These models had identical optimiser functions (Adam) and loss functions (categorical crosscross entropy) as in Chapter 7, but the target labels were the number of channels open at a given point in the signal rather than the Markovian state as in Chapter 7.

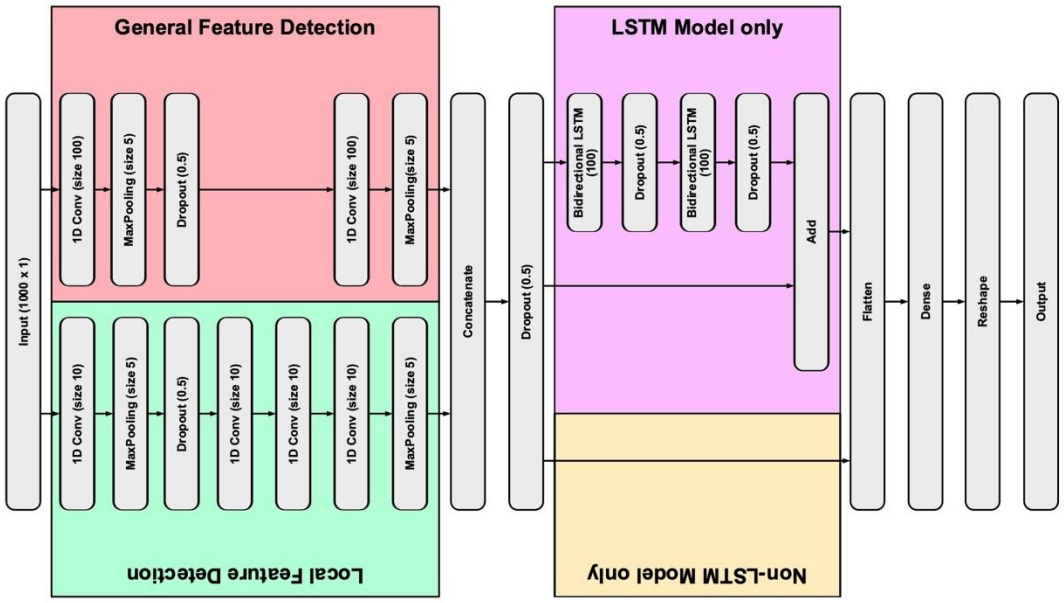


Figure 8.9: New SplitCNN Model Architecture adding an LSTM route in the second stage.
In the original DeepSleepNet paper which inspired the SplitCNN design (Supratak et al., 2017), an LSTM route is used in the second stage of the model to allow for longer-term feature detection. This new model is added to the models we test on multichannel data.

In addition to these two models, DeepChannel (Celik et al., 2020) was re-added into the model consideration as it has been shown to perform multi-channel idealisation accurately in previous work. In addition a new model was added: similar to the Split CNN model (Figure 8.9) but with an additional LSTM step after the convolutional blocks to better

encode historical data, as this model has also seen high performance in similar tasks (Supratak et al., 2017).

8.3.6.3 Data Training

For the synthetic training data in this chapter, multiple independent Markovian processes were simulated and added together, and noise added in post to simulate ambient conditions in the lab. In total, 8 different datasets were trained (drift/non-drift datasets both for 1, 3, 5 and 10 channels). Each of these consisted of 24 files of 10mins of data at 10kHz for training, totalling 144 million input datapoints, and 281250 training samples after the windowing process. A further 24 equivalent files were used for testing each model, and 12 files more for mid-training validation. In addition to the 1/f noise sampling to replicate realistic noise, a digital patch-clamp style 8-pole Bessel filter was applied to the resulting data at a cut off frequency of 2kHz.

The models were then trained in a similar method to Chapter 7, with automatic learning rate scheduling and early stopping based on validation performance. The training progress for each model, and evaluation on the testing datasets were noted. These models, trained on the synthetic data were then used to idealise the lab-recorded data to gain a point-by-point classification of the raw signal.

8.3.7 Data Postprocessing

After obtaining the automated model idealisations for the patch-clamp (real) data files, some postprocessing was applied to the data to improve model results.

8.3.7.1 Time Domain Rescaling

The idealisations were unscaled by the timepoint (correcting decimation (see section 8.3.5.3), and then passed through a dead-time filter to remove events smaller than or equal to 10 points long to remove flickering.

In the "Calcium" dataset, three candidate models from previous work were used to analyse the data, however in the "Vernakalant" and "Penitrem" datasets only the best of these models was used as performance was similar across all the models in testing. For the "Vernakalant" and "Calcium" datasets, an additional QuB idealisation was performed as a baseline to compare against. This idealisation would act as a ground truth for the following processing.

From the idealisations; a baseline correction was made to set the baseline in the data to the baseline in the idealisation to avoid biasing the *nPo*. The *nPo*s were then calculated by using the following formula:

$$nPo = \sum_{k=0}^{m} kn_k$$

Where k is the conductance level, n_k the number of points at the conductance level, and m the maximum conductance level observed.

For cases where a drug was applied, the relative *nPo*s were calculated by dividing the *nPo*s by the *nPo* observed when no drug was used.

In all cases, dwell-time histograms for the two most common conductance levels were generated fitted using the standard method of log-binning the dwell times and using a square root axes for the frequencies. Curves with the equation:

$$F(x) = \sum a_i e \cdot e^{x-\tau_i - e^{x-\tau_i}}$$

Were fitted to the dwell time histograms. This equation is slightly different to the typical equation used to fit dwell time histograms (Sigworth & Sine, 1987), however the inclusion of the extra constant e is helpful for fitting as each a_i now directly corresponds to the peak of each exponential function.

The areas (a_i) and centres (τ_i) of each of these fits were then compared to check for changes in the underlying Markovian process of the channel, along with the relative nPos (for

checking if a change in conditions increased or decreased channel activity) and gaussian fitted parameters for the amplitude histograms (for testing changes in channel size).

8.4 Results

8.4.1 Multichannel Model Training on Synthetic Data

For idealisation of synthetic data, the micro F1score and Cohen's Kappa score were recorded for each model trained on each dataset (Figures 8.10-8.25;); in total 40 models were trained and given a code name moving forward; the Simple CNN models were designated A, Split CNN +LSTM B, and UNet C, Split CNN without LSTM D, and DeepChannel E. The number of channels the dataset was generated to have would make the next part of the codename, then whether or not the training data had added drift would make up the final part (D or ND for drift/no-drift). Therefore the 5 channel Split CNN model with LSTM with drift added would be represented by Model B5D. The models showed mixed results, with significant difference between each model ($p < 0.001$, $n=24$) but no model consistently outperformed the others. For example in the single channel dataset the CNN models significantly outperformed DeepChannel ($p < 0.001$, $n=24$) but on the 5 channel dataset without drift, DeepChannel was far superior ($p < 0.001$, $n=24$) for all models bar the Simple CNN model 5AND)

Following additional assessment of apparent errors produced with preliminary testing of sections of real data, the 5 channel drift version of the LSTM-SplitCNN model (seen in section 8.3.6.2) was chosen to be taken forward for idealisation of real data in the following analysis.

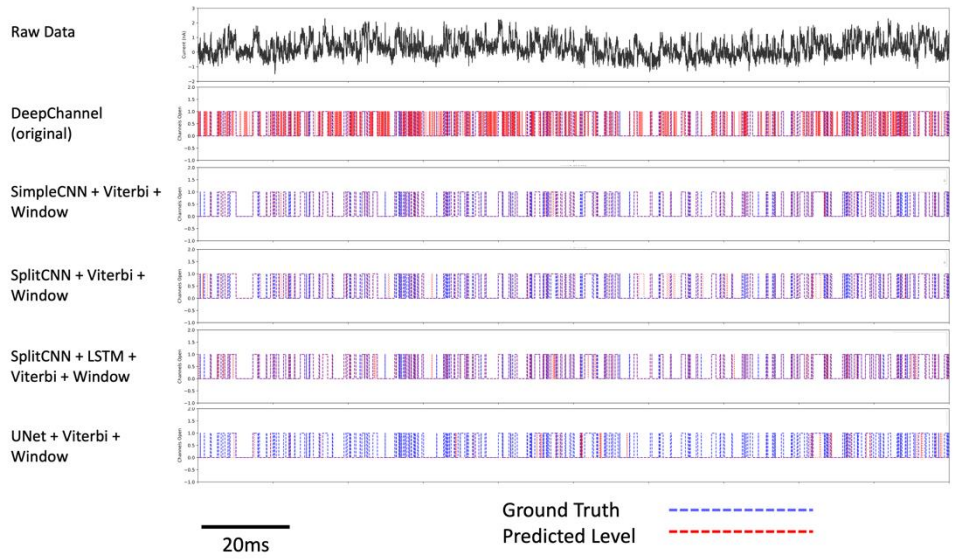


Figure 8.10: Model Training Traces for "One Channel, No Drift" Synthetic Dataset
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

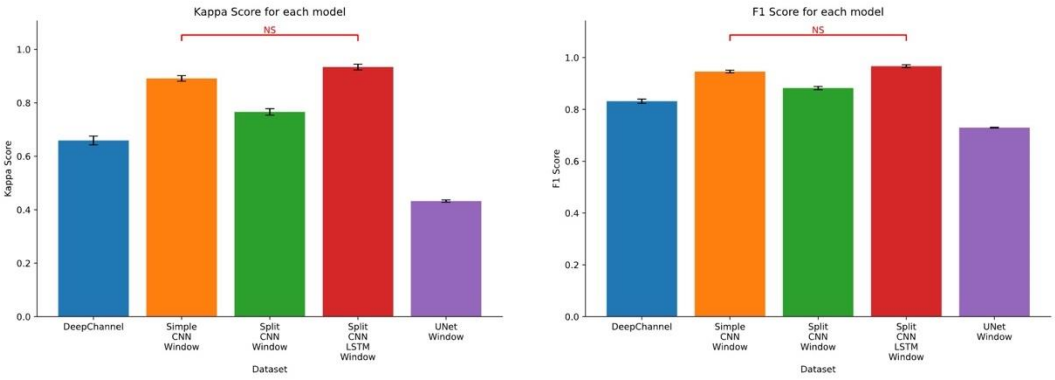


Figure 8.11: Model Training Metrics for "One Channel, No Drift" Synthetic Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

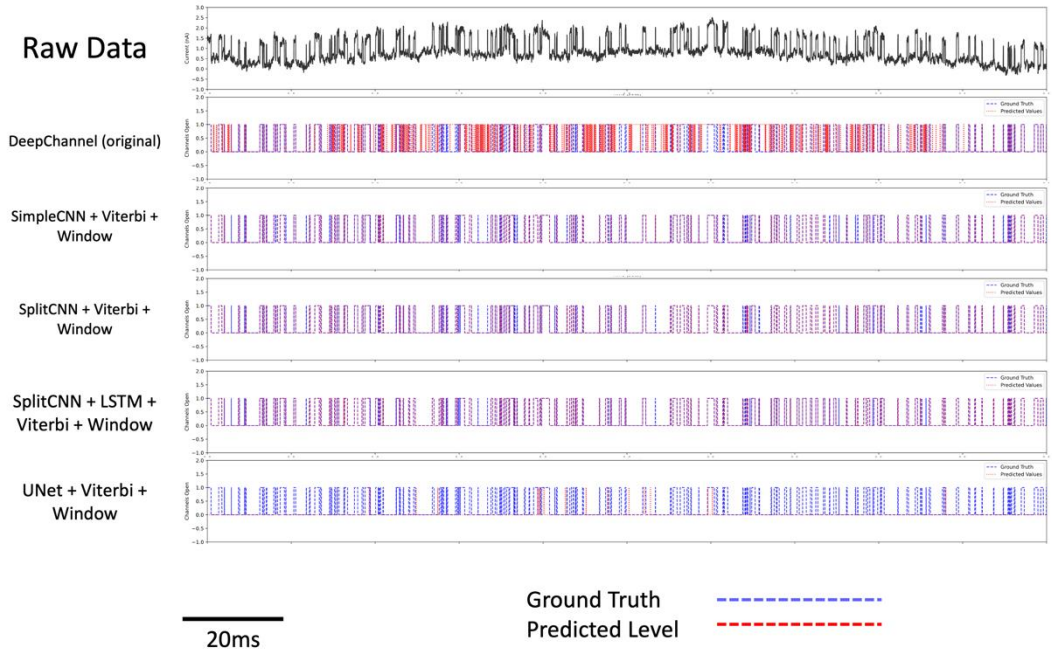


Figure 8.12: Model Training Traces for "One Channel, Drift" Synthetic Dataset.
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

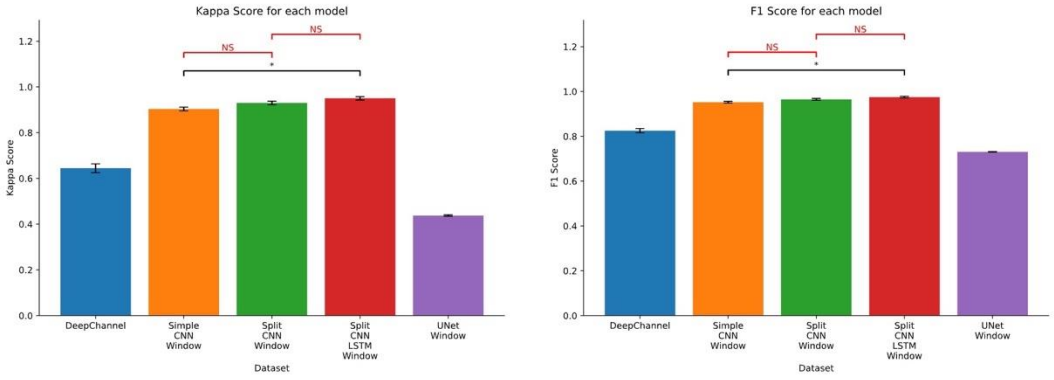


Figure 8.13: Model Training Metrics for "One Channel, No Drift" Synthetic Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

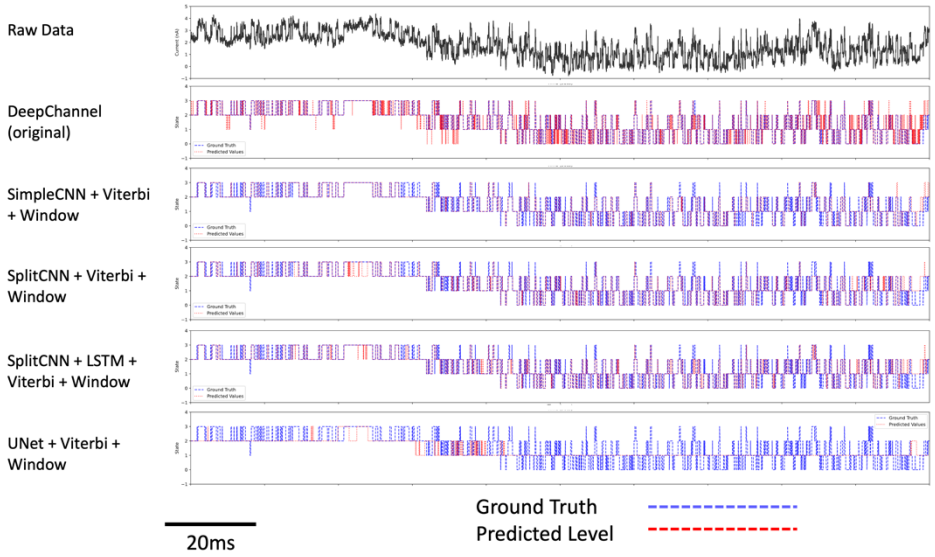


Figure 8.14: Model Training Traces for "Three Channels, No Drift" Synthetic Dataset
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

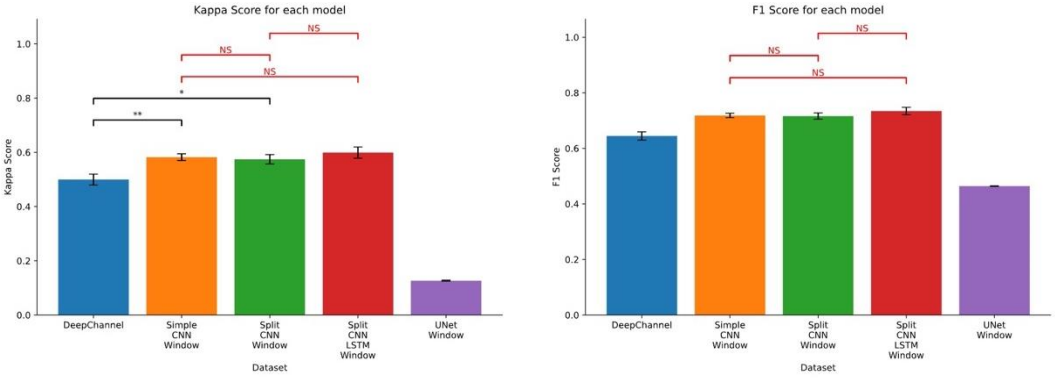


Figure 8.15: Model Training Metrics for "Three Channel, No Drift" Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

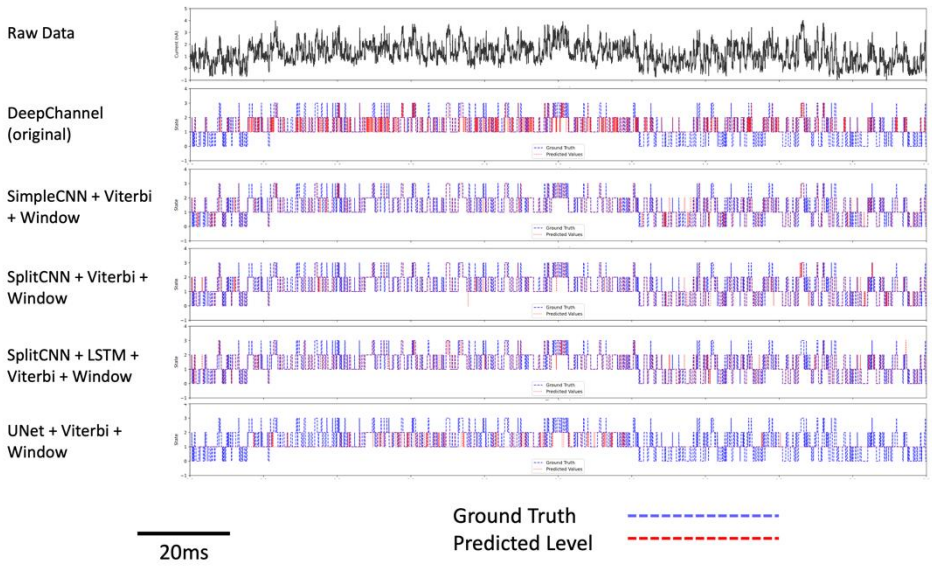


Figure 8.16: Model Training Traces for "Three Channels, Drift" Synthetic Dataset
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

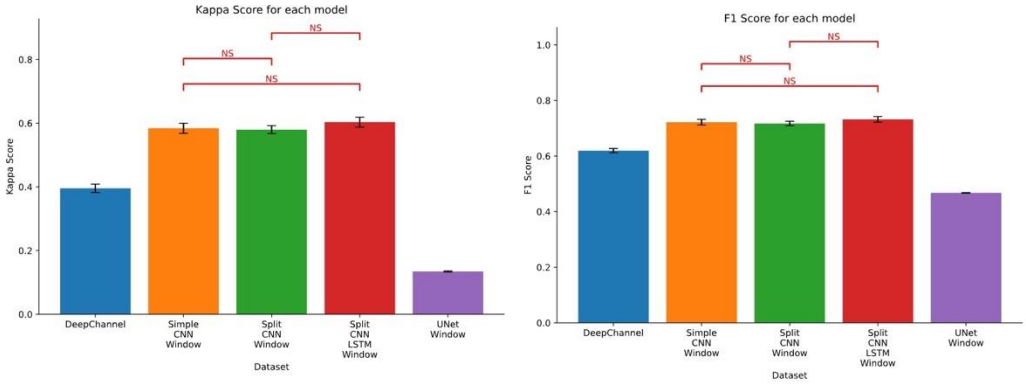


Figure 8.17: Model Training Metrics for "Three Channel, Drift" Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

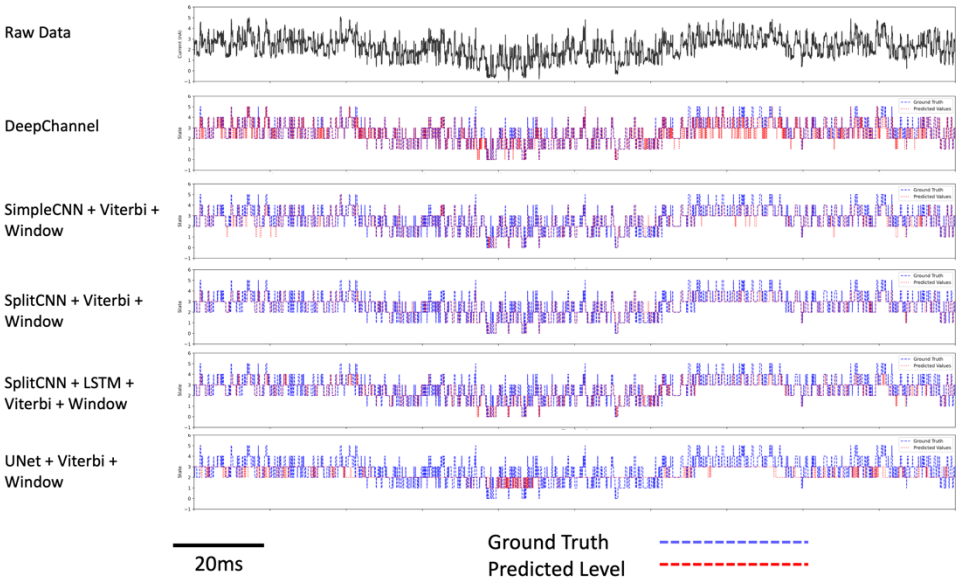


Figure 8.18: Model Training Traces for "Five Channels, No Drift" Synthetic Dataset.
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

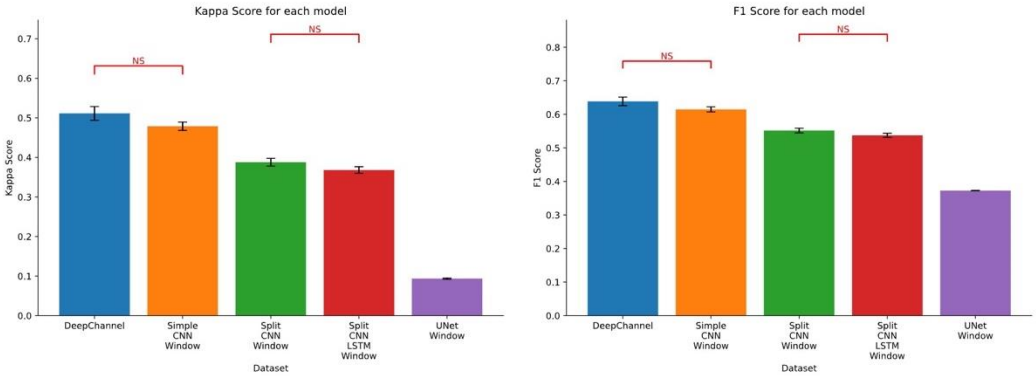


Figure 8.19: Model Training Metrics for "Five Channel, No Drift" Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

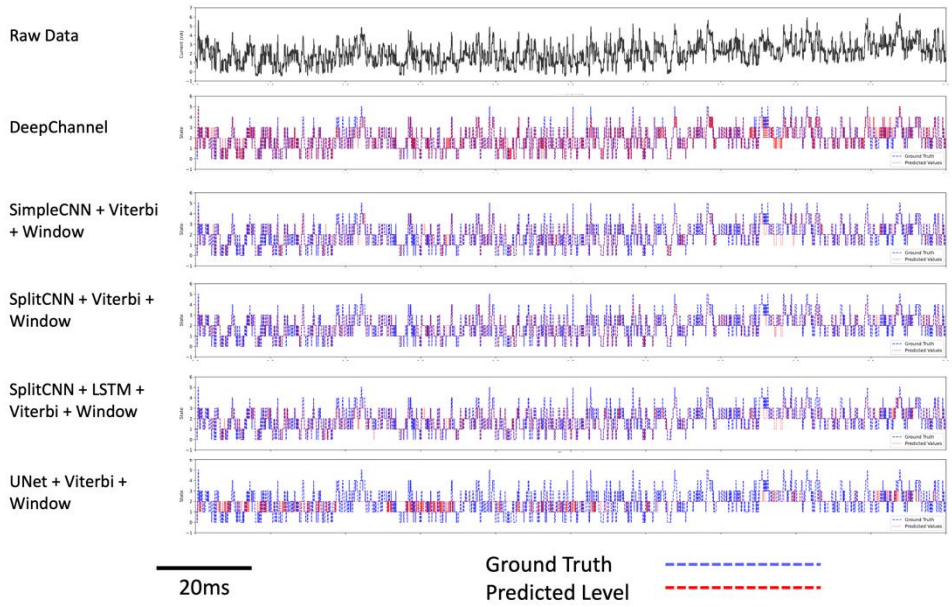


Figure 8.20: Model Training Traces for "Five Channels, Drift" Synthetic Dataset.
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

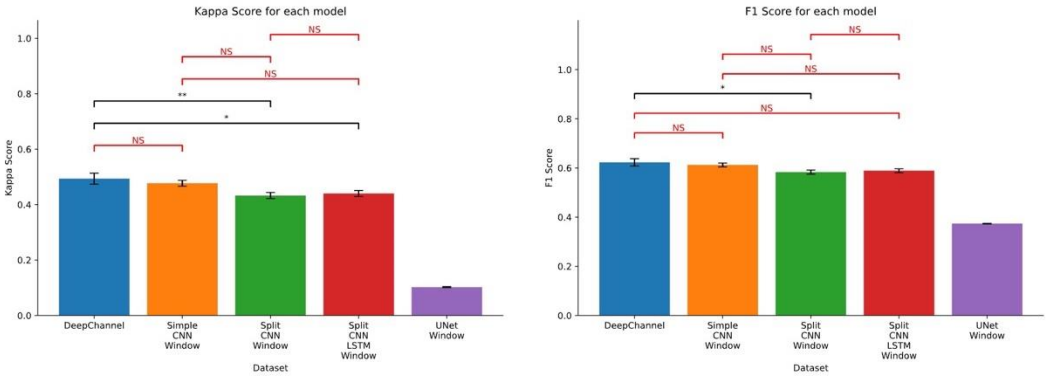


Figure 8.21: Model Training Metrics for "Five Channel, Drift" Dataset
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

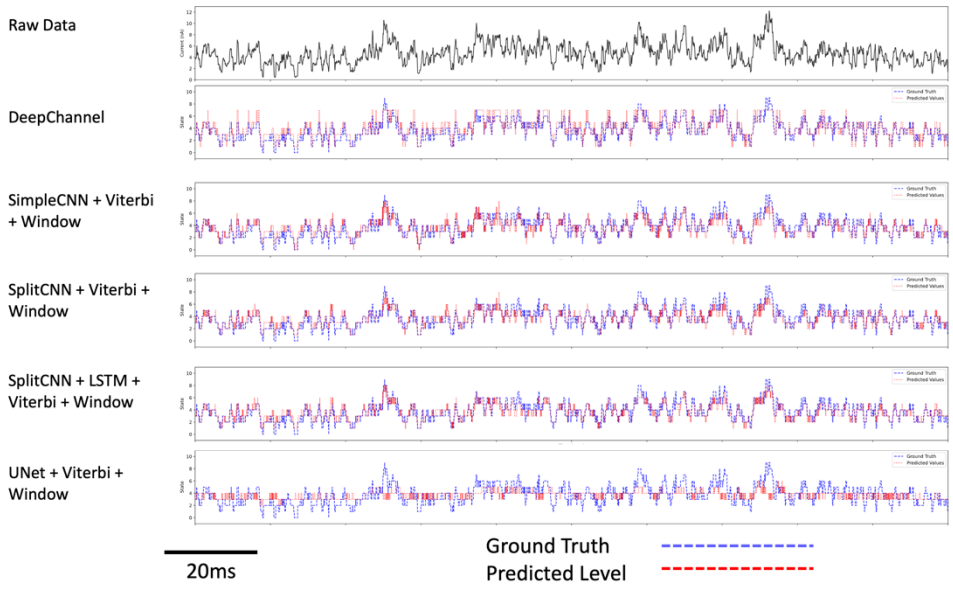


Figure 8.22: Model Training Traces for "Ten Channels,No Drift" Synthetic Dataset.
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

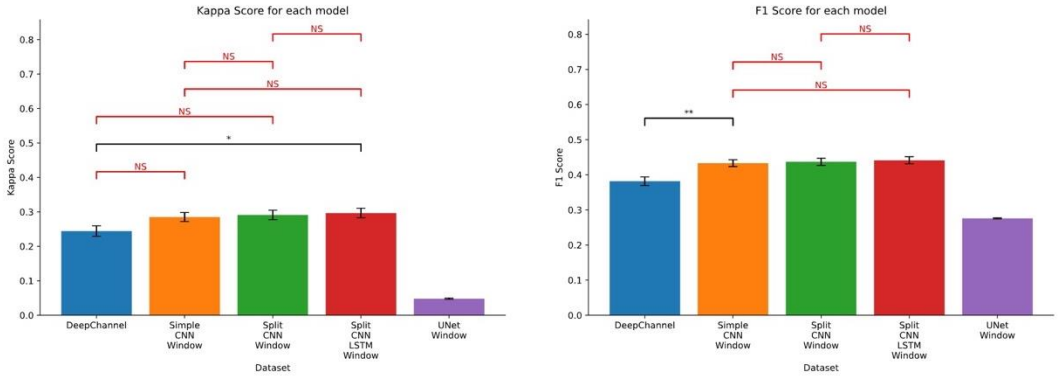


Figure 8.23: Model Training Metrics for "Ten Channel, No Drift" Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

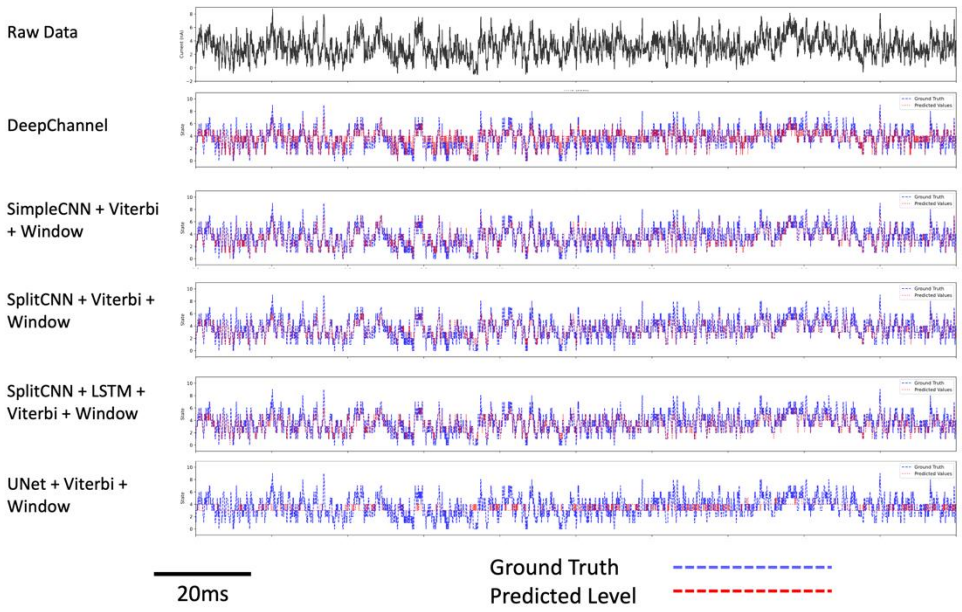


Figure 8.24: Model Training Traces for "Ten Channel, Drift" Synthetic Dataset.
Representative traces for each model are seen above, with the ground truth in blue and model idealization in red. Here, DeepChannel shows oversensitivity to events and UNet under sensitivity.

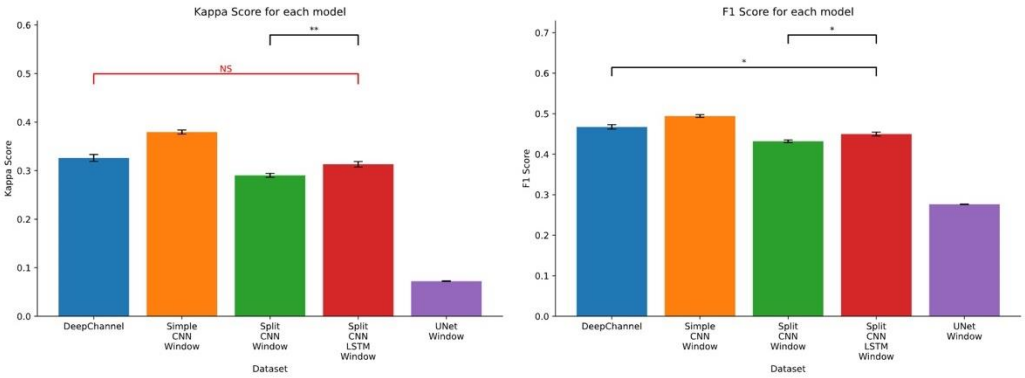


Figure 8.25: Model Training Metrics for "Ten Channel, Drift" Dataset.
Two way ANOVA showed significant differences of the models for both F1 score and Cohen's Kappa score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

8.4.2 "Calcium" Dataset

The "Calcium" dataset contained recordings for a number of BK channel IOP patch-clamp experiments with different holding potentials and Ca^{2+} concentrations (at the cytoplasmic

face of the membrane). These data were split in to two groups: (i) Recordings at either -40mV and +40mV with *changing Ca^{2+} concentration*, and (ii) recordings at a Ca^{2+} concentration of 17.7μM with *differing holding potentials* (-100mV to 100mV stepped by 10mV).

Sample idealisations of files of the "Calcium" dataset can be seen in Figures 8.26 and 8.27. Qualitatively, we found that the idealisation process was mostly successful (Figure 8.26) with some errors resulting in a blank idealisation (Figure 8.27). This could be caused either by model under/overfitting, or a problem in the pre-processing pipeline (such a TANE malfunction, see discussion). Formal model validation with this dataset, however, requires comparison of the physiological phenotype with the well-established data in the literature, by way of comparing further physiological analysis with current methodologies, in addition to how successfully the model detects drug action known to occur.

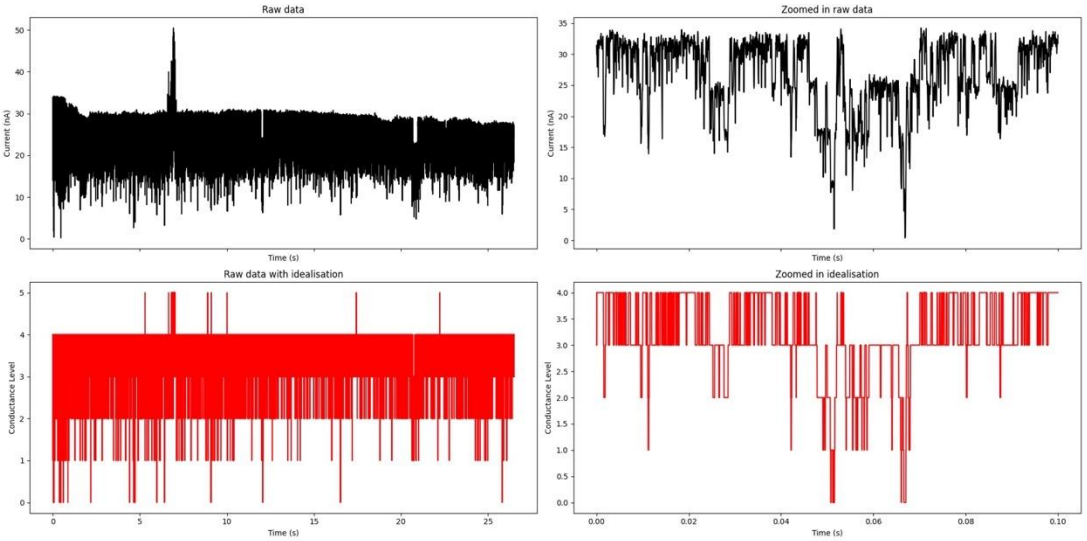


Figure 8.26: Successful idealisation from the "B5D" model for "Calcium" dataset.
Here we show a raw data trace from the "Calcium" dataset, and the corresponding idealization – we see that the model successfully idealises the record, successfully converting the raw noisy signal into a square wave.

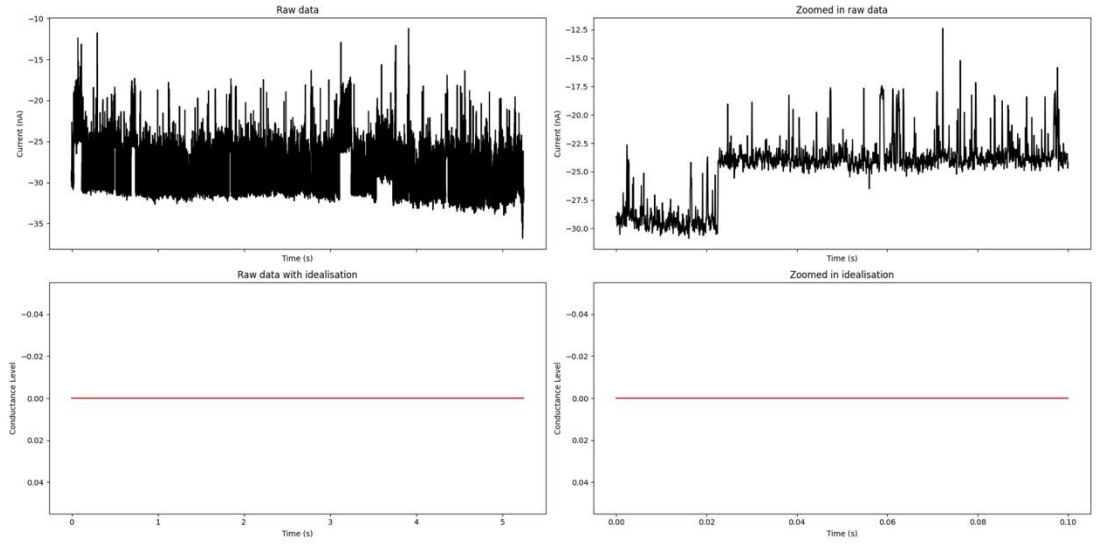


Figure 8.27: Unsuccessful idealisation from the "B5D" model for "Calcium" dataset.
Here we show an example where irrespective of metrics it is clear that the model has not idealized correctly. This can be caused by a number of issues, (see full text); but is likely fixed by further work into improving the quality and range of the training data, and the pre-processing pipeline.

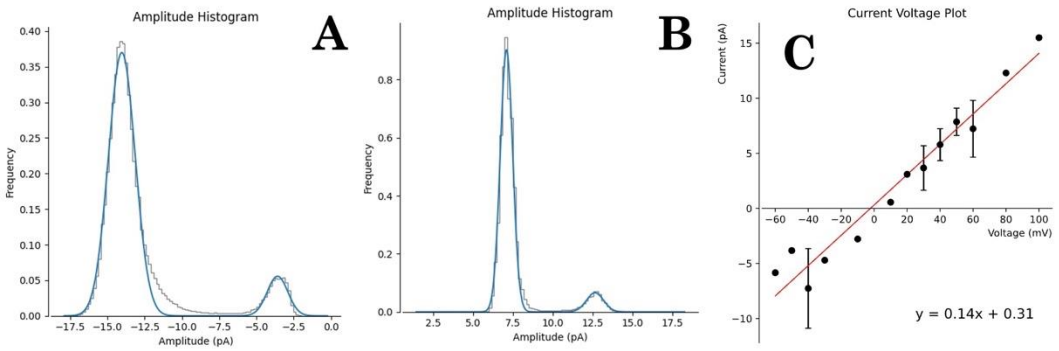


Figure 8.28: Amplitude histograms and unitary conductance plot for "Calcium" dataset.
A and B show representative examples of current histograms for -40mV and 40mV recordings respectively, fitted with Gaussian curves. C shows unitary conductance plot of voltage vs channel size (current). A linear regression (n=42) showed a significant relation between the two variables ($p < 0.001$). The unitary channel conductance, measured as slope of unitary current against voltage (di/dV) corresponded to 137.7±29.8pS (n= 42), consistent with the literature on BK channel conductance of 100-300pS.

8.4.2.1 Single Channel Amplitude Histograms

Single-channel amplitude histogram analysis of the calcium dataset (Figure 8.28) showed a significant, approximately Ohmic relationship between single-channel amplitude and voltage for the channel ($p < 0.001$), as expected. The unitary channel conductance, measured as slope of unitary current against voltage (di/dV) corresponded to 140±31pS (n= 42). This assumes that the current was non-rectifying over the recording range. It should be noted, however, that this analysis is done prior to idealisation and is separate from model performance.

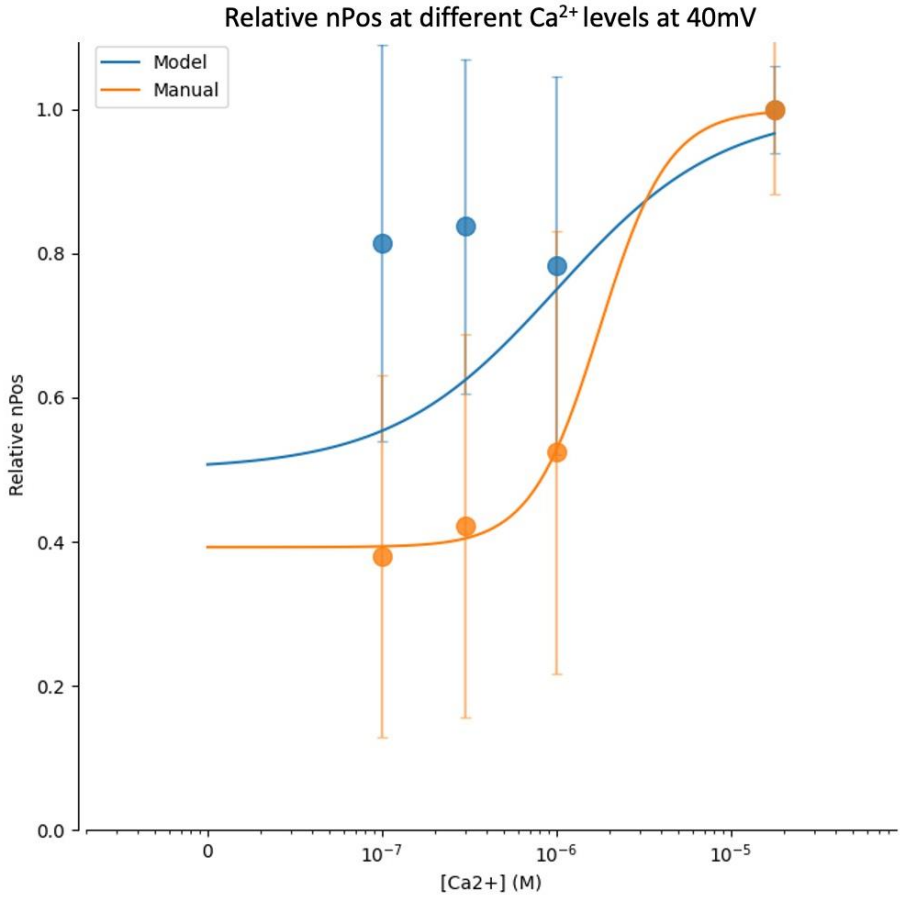


Figure 8.29: Dose Response Curves Relative (to 17.7uM) nPo at different levels of Calcium for both B5D model idealisations (Model) and manual idealisations (Manual).
2 way ANOVA for the effects of the concentration of calcium showed a small significance of the amount of calcium applied to the channel to the open probability (p = 0.029, n=2) and no significance for a difference in the model idealisations versus the manual qub idealization. DRC were fit with 3-parameter logistic models, Slope (B5D, Slope: -6.5e+00±9.4e+01, Min: 8.1e-01±2.2e-02, EC50: 5.2e-06±1.2e-04 Mand QuB, Slope: -1.9±6.4e-01, Min: 3.9e-01±2.0-02, EC50: 1.9e-06±4.0e-07 M, n=2).

8.4.2.2 Calcium Dose Response Curves

For dose response curve analysis we used *nPos* (the mean number of channels open across the trace, see methods) rather than unitary amplitudes. Following idealisation with either a manual, QuB method (see section 8.3.5) or our deep learning model (B5D) we derived two independent dose response curves for *nPos* for the activation of BK with Ca^{2+} (Figure 8.29) at 40mV membrane potential. Both models revealed the expected increase of open probability (nPos) with increasing [Ca^{2+}] with a small significant difference for *nPos* between Ca^{2+} concentrations ($p = 0.029$, $n = 42$, 2-way ANOVA). EC50s calculated for the Ca^{2+} activation effect were approximately 5 and 2uM for the QuB our model "B5D" respectively (Figure 8.29).

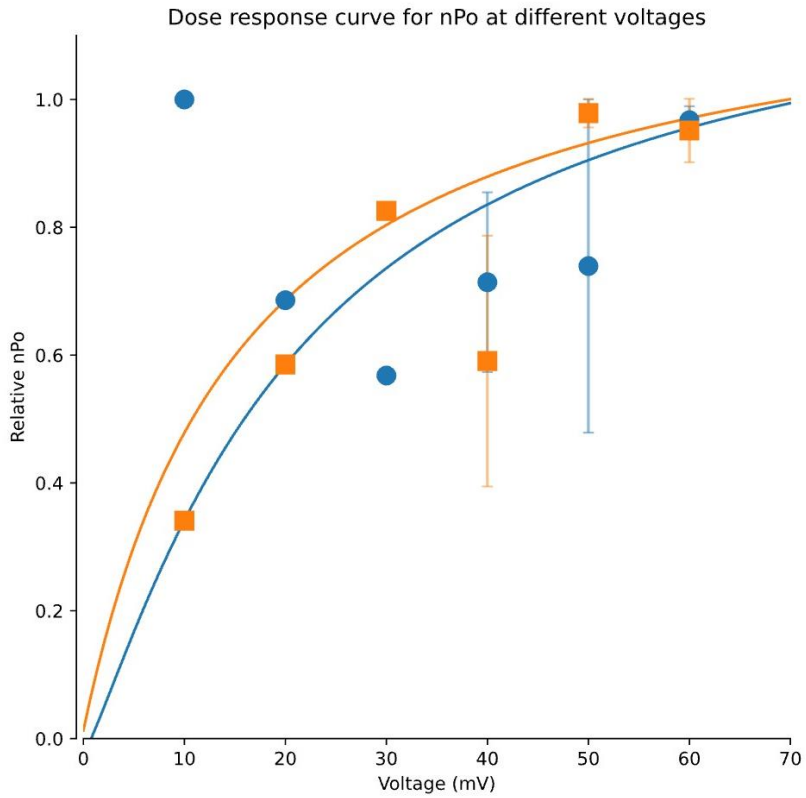


Figure 8.30: Relative nPo at different Voltages for both model idealisations (Model) and manual idealisations (Manual).
For current voltage fits:, the V1/2 was 21.3±0.02mV (p = 1e-6) and that with B5D was 16.07±3.1 mV (p=5e-5, n=1). In this case, the maximum nPo for each cell was used as the benchmark for relativity. Minimum and maximum were approximately 0 and 1.0 in both cases, since this was normalised data.

8.4.2.3 Current (nPos) -voltage Analyses

One of the BK channels defining properties is activation by voltage therefore I used the same approach as above (*Ca^{2+} dose response curves*) to measure this property with both the QuB method and B5D in the presence of a fixed intracellular Ca^{2+} intracellular concentration. Figure 8.30. Fits were limited to positive membrane potentials and one concentration of Ca^{2+} due to data availability (17μM). Fit to the QuB dataset gave a midpoint for voltage activation of 21.3±0.02mV (p = 1e-6) and that with B5D was 16.07±3.1 mV (p=5e-5).

8.4.2.4 Kinetic Analyses

Dwell time histograms were calculated (Figure 8.31) using a 3 exponential curve fit on both the model and QuB idealisations to compare kinetic analysis between the manual and automatic idealisations. I compared these fits systematically with 2-way ANOVA. The dwell time parameters (Taus and Areas 1, 2 and 3 for both open and closed dwell times) for both the model and manual idealisations are shown in Figure 8.31 and 8.32. For each set of parameters (Closed or Open Taus and Areas), a 2-way ANOVA was carried out to test for the effect of Ca^{2+} concentration and results can be seen in Figures 8.33 to 8.39 and Table 8.4. In addition, a 2-way ANOVA showed significant differences in the means of the overall tau and area parameters between the manual and model approaches (p = 0.0319, n= 34), and deep learning model showed a lower variance in every parameter(p =7= 7.3960e-7, Fisher Exact testtest, n = 1212). The lack of change of kinetics is in line with the current literature that BK channels show no change in Markovian structure with Ca^{2+} concentration (Geng & Magleby, 2015; Nimigean & Magleby, 2000) .

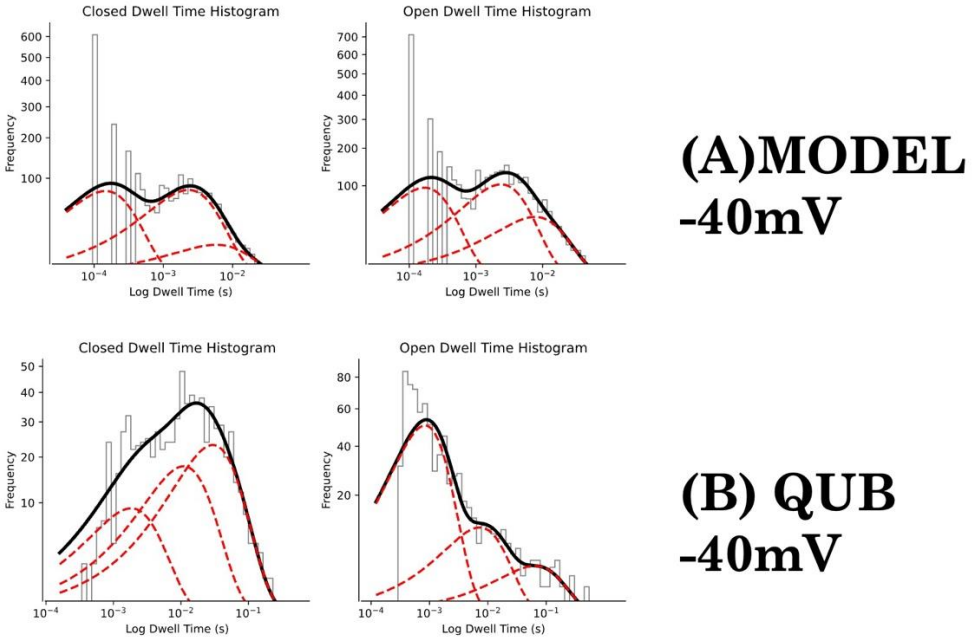


Figure 8.31: Representative dwell time histograms for both the model (A) and QuB (B) idealizations at -40mV. *Dwell times for representative example data recorded at -40mV can be seen above for both the model idealisation and the manual QuB idealisation. All idealisation dwell times were fitted with 3 exponential plots, noting the relative areas and centres (taus) for each curve. Summarised fit parameters can be seen in Figures 8.33 to 8.39, along with the supplementary figures.*

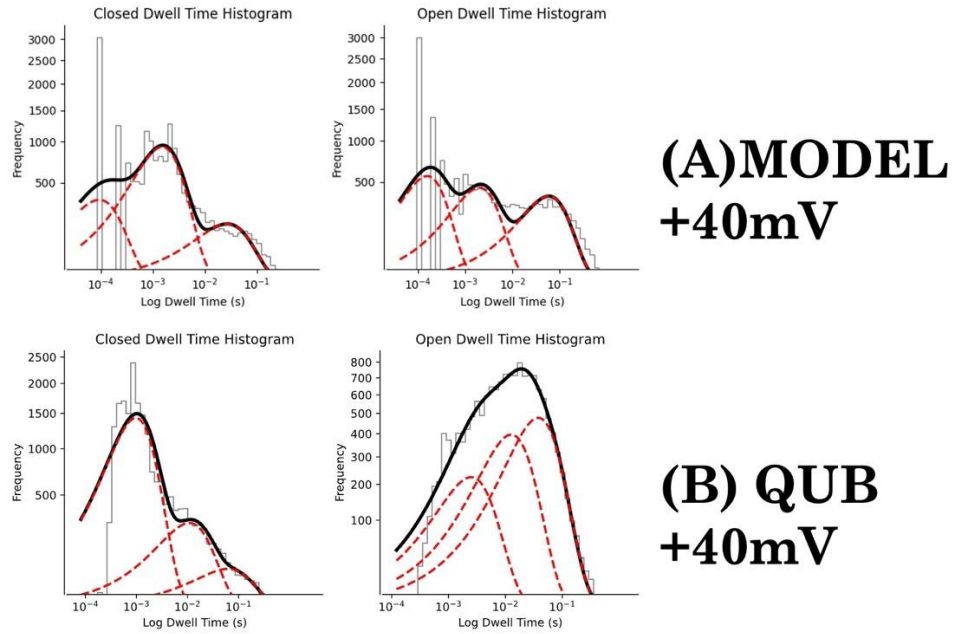


Figure 8.32: Representative dwell time histograms for both the model (A) and QuB (B) idealizations at +40mV. *Dwell times for representative example data recorded at 40mV can be seen above for both the model idealisation and the manual QuB idealisation. All idealisation dwell times were fitted with 3 exponential plots, noting the relative*

areas and centres (taus) for each curve. Summarised fit parameters can be seen in Figures 8.33 to 8.39, along with the supplementary figures.

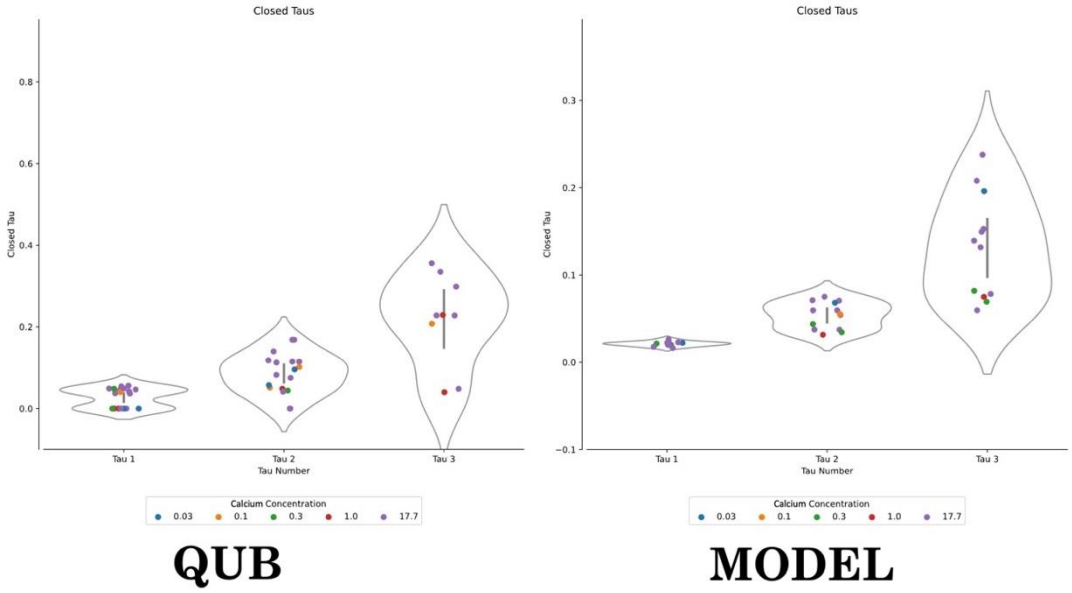


Figure 8.33: Dwell Time Parameter Plots for Closed Tau Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these closed tau parameters for both idealisations. A two way ANOVA showed a significant effect of the drug on the closed taus for either the model idealisations (p = 0.0478, n = 34), and also for the QuB idealisations (p = 0.0350, n = 34). Furthermore, while the model exhibits a tighter clustering of parameters. Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 34); however this does not necessarily imply one is better than the other.

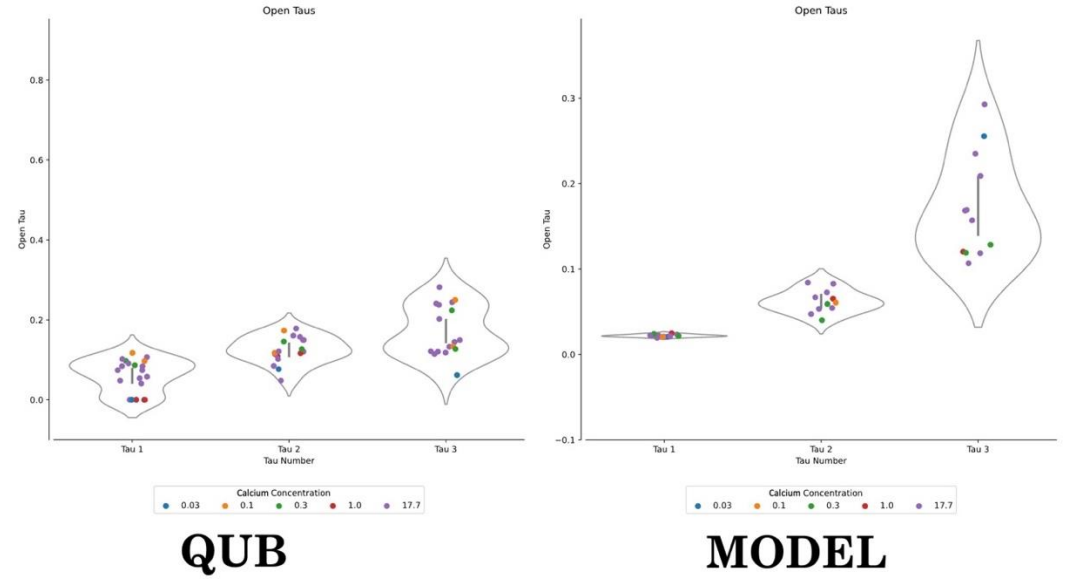
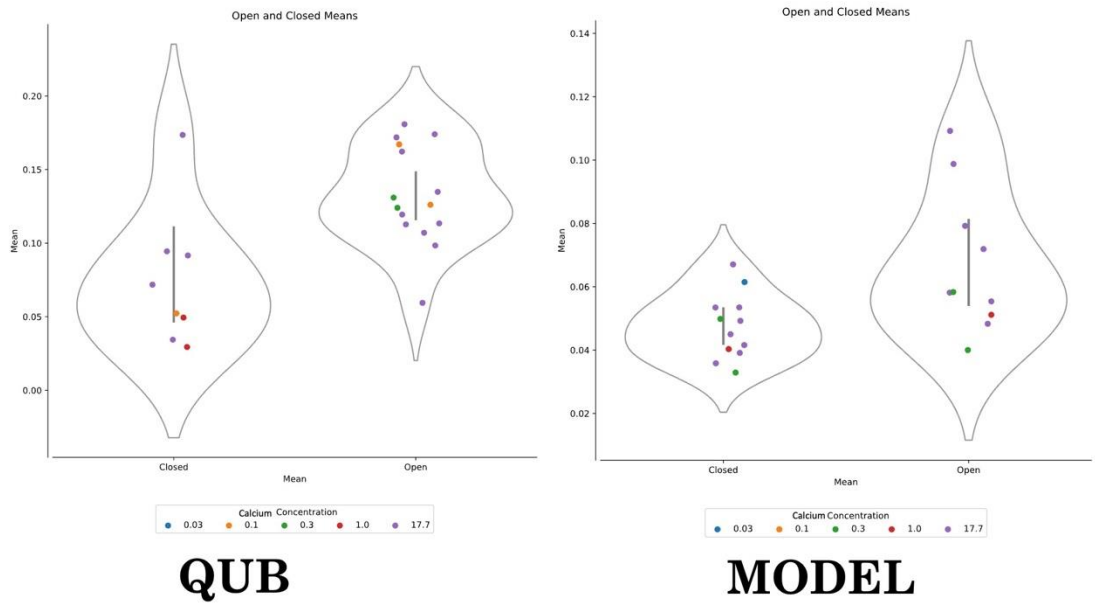


Figure 8.34: Dwell Time Parameter Plots for Open Tau Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

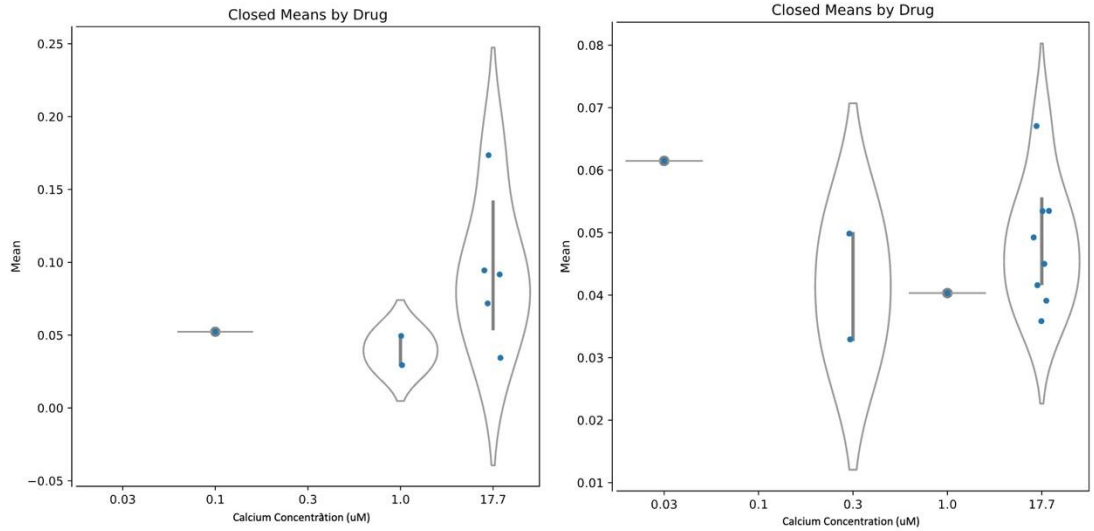
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these open tau parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the open taus for either the QuB or model idealisations (p > 0.05, n = 34). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test (p > 0.05, n= 34). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 34); however this does not necessarily imply one is better than the other.



QUB **MODEL**

Figure 8.35: Dwell Time Parameter Plots for Open and Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus). A two way ANOVA showed a significant effect of the drug on the open areas for the QuB idealisations (p < 0.001 respectively, n = 34) but not the model idealisations (p > 0.05, n = 34). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test (p > 0.05, n= 34). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 34); however this does not necessarily imply one is better than the other.



QUB **MODEL**

Figure 8.36:Dwell Time Parameter Plots for Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by calcium concentration at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the closed mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by calcium concentration. A two way ANOVA showed a significant effect of the drug on the open areas for the QuB idealisations (p < 0.001 respectively, n = 34) but not the model idealisations. Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test (p > 0.05, n= 34). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 34); however this does not necessarily imply one is better than the other.

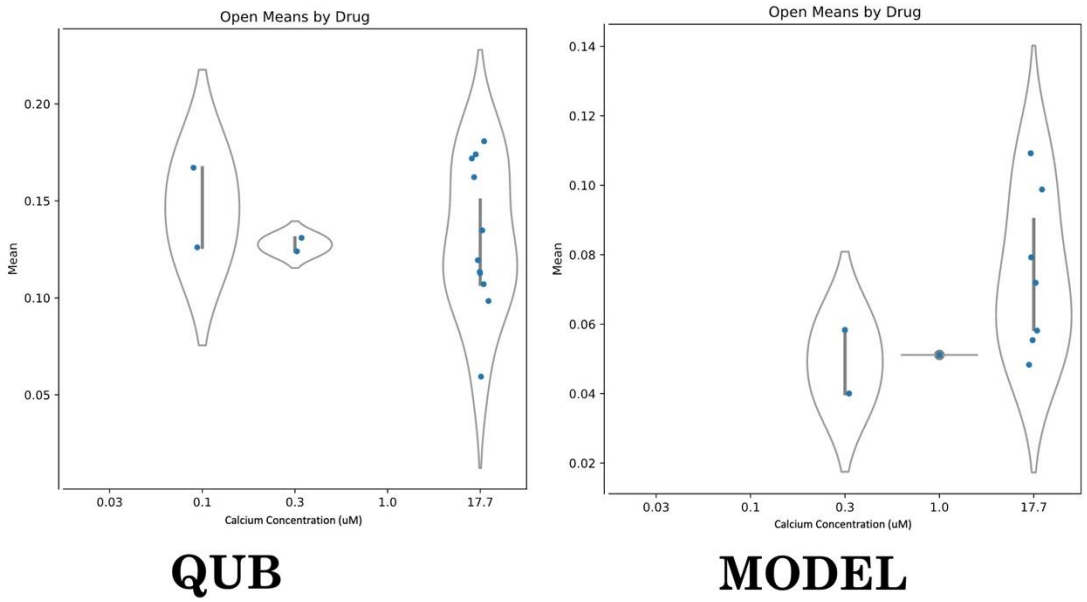


Figure 8.37: Dwell Time Parameter Plots for Open Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by calcium concentration at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the open mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by calcium concentration. A two way ANOVA showed a significant effect of the drug on the open areas for the QuB idealisations (p < 0.001 respectively, n = 34) but not the model idealisations. Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test (p > 0.05, n= 34). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 34); however this does not necessarily imply one is better than the other.

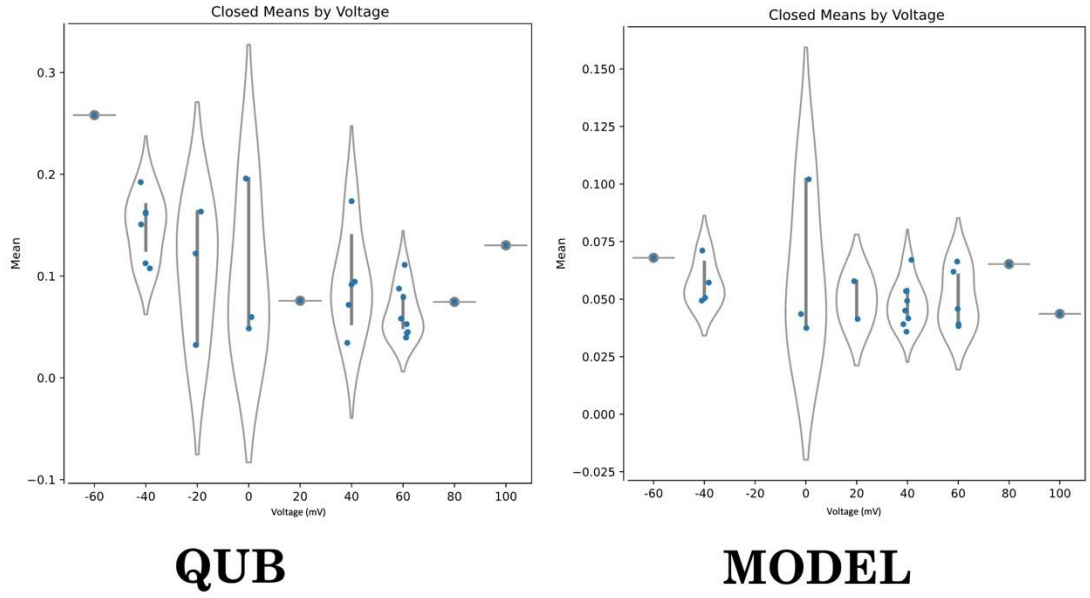


Figure 8.38: Dwell Time Parameter Plots for Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by voltage at 17.7uM.
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the closed mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by voltage.

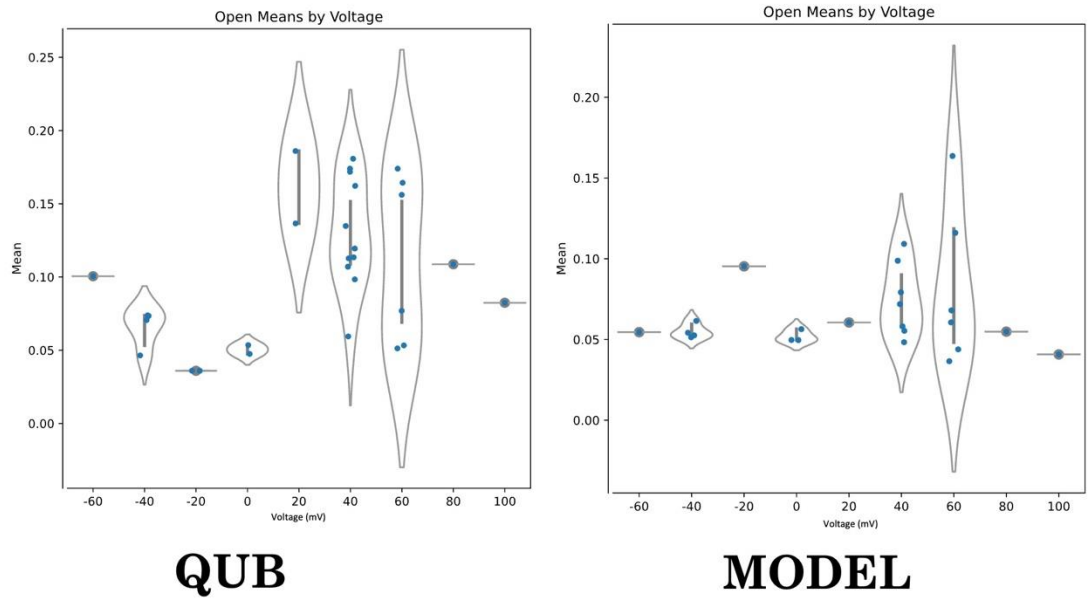


Figure 8.39: Dwell Time Parameter Plots for Open Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by voltage at 17.7uM.
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the open mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by voltage.

Table 8.4: Kinetic Sensitivity to Ca2+ Statistical tests for each group of dwell time parameters at 40mV
Each group of dwell time parameters (open taus, closed taus, open areas, closed areas) were tested via 2 way ANOVA (n = 20) for the effect of change in Ca2+ concentration. Means were tested via 1 way ANOVA for the same effect.

Parameter	QuB Ca^{2+} Effect ANOVA	Model Ca^{2+} Effect ANOVA
Closed Taus	P = 0.0350	P = 0.0478
Open Taus	P = 0.0027	P > 0.05
Closed Areas	P > 0.05	P > 0.05
Open Areas	P > 0.05	P > 0.05
Closed Means	P > 0.05	P > 0.05
Open Means	P > 0.05	P > 0.05

8.4.3 "Penitrem" Dataset

The "Penitrem" dataset contained multiple recordings of BK activity with different concentrations of a known BK channel inhibitor, Penitrem A, recorded using IOP, mostly at -60mV (membrane potential). For this dataset, 3 models were compared: (i) A manual idealisation performed with pClamp (see Chapter 8.3.5, provided along with the raw data), (ii) an idealisation using manual TANE and the B5D model; and (iii) an idealisation using automatic TANE and B5D model.

Figure 8.40 and Figure 8.41 show representative examples of the "Penitrem" dataset along with idealisation with the automatic TANE B5D model. As with the previous dataset, errors are apparent; sometimes the model would insert events where there were none, or miss events entirely.

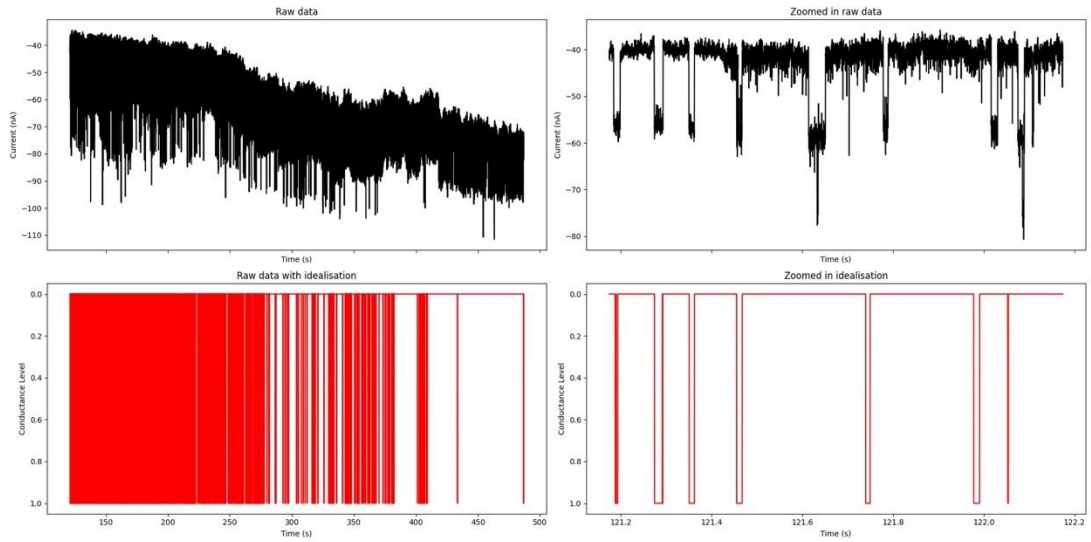


Figure 8.40: Successful idealisations from the "B5D" model for "Penitrem" dataset.
This challenging dataset exhibits significant baseline drift, a type of noise avoided in similar work. Here our model sees good performance, missing a few events but overall achieving good accuracy.

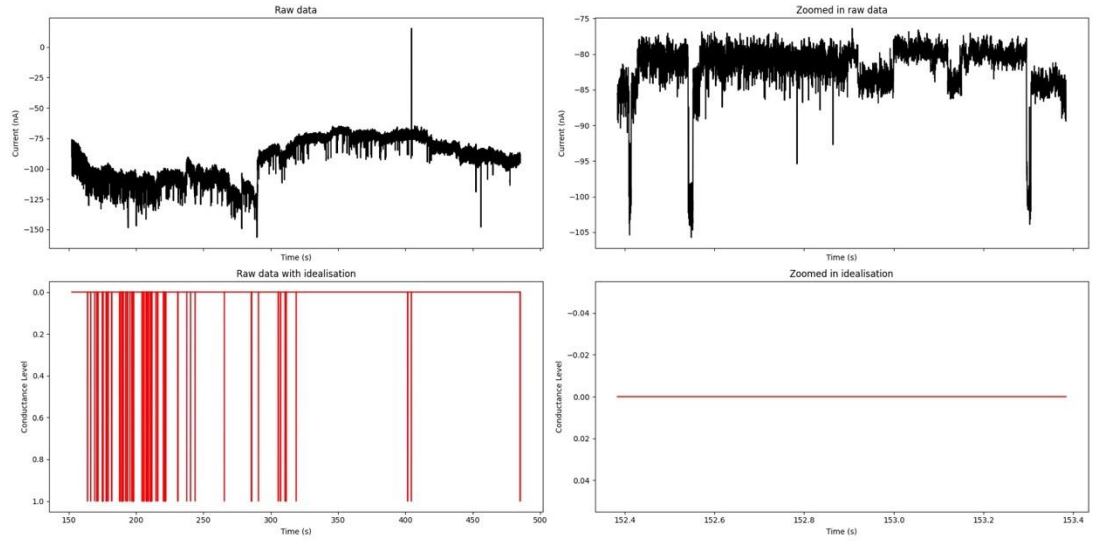


Figure 8.41: Unsuccessful idealisations from the "B5D" model for "Penitrem" dataset.
Sometimes it is clear, from observation rather than metrics that idealization has failed. This can be a result of a number of issues, including poor recording conditions, or incorrect parameters in the pre-processing steps. This problem can be resolved by further improvements in the pre-processing algorithms, or increasing the quality of the training datasets.

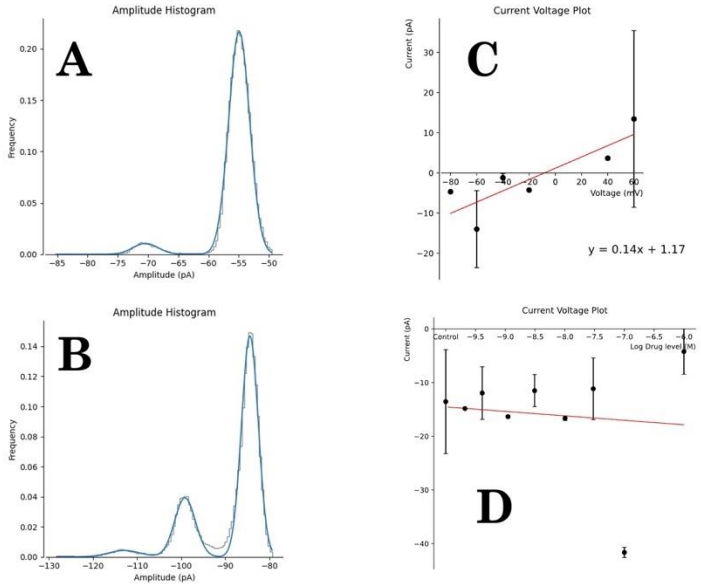


Figure 8.42: Example amplitude histograms, unitary conductance by voltage, and current dose response plots for "Penitrem" dataset.
A and B show representative examples of two current histograms at for data recorded at -60mV, fitted with Gaussian curves. C shows the unitary conductange/channel sizes (adjusted to negative when channels open downwards) versus voltage. D shows this same relationship but split by the amount of drug instead of voltage (i.e. a dose response curve with respect to current). In both C & D points further than 2 standard deviations from the mean were filtered to remove outliers. Statistical analysis of the linear regression showed a linear relationship between voltage and current (p < 0.05, R^2 = 0.675, n=117) but not dependency between drug level and channel size (p > 0.05, n=117).Again, the unitary channel conductance, measured as slope of unitary current against voltage (di/dV) corresponded to 140.6±105pS (n= 117), consistent with the literature on BK channel conductance of 100-300pS.

8.4.3.1 Amplitude histograms

The amplitude histogram analyses (Figure 8.42) found a significant relation between voltage and channel size as expected ($p < 0.05$, $R^2 = 0.675$, $n=117$), but no significant association between the channel size to the drug concentration ($p > 0.05$). The unitary slope conductance, calculated as above was 140.6±105pS (n= 117)

8.4.3.2 Penitrem Dose Response Curves

For dose response curve analysis, we used *nPo* as well as unitary conductance. These were measure at different Penitrem concentrations at a membrane potential of -60mV (Figure 8.43).

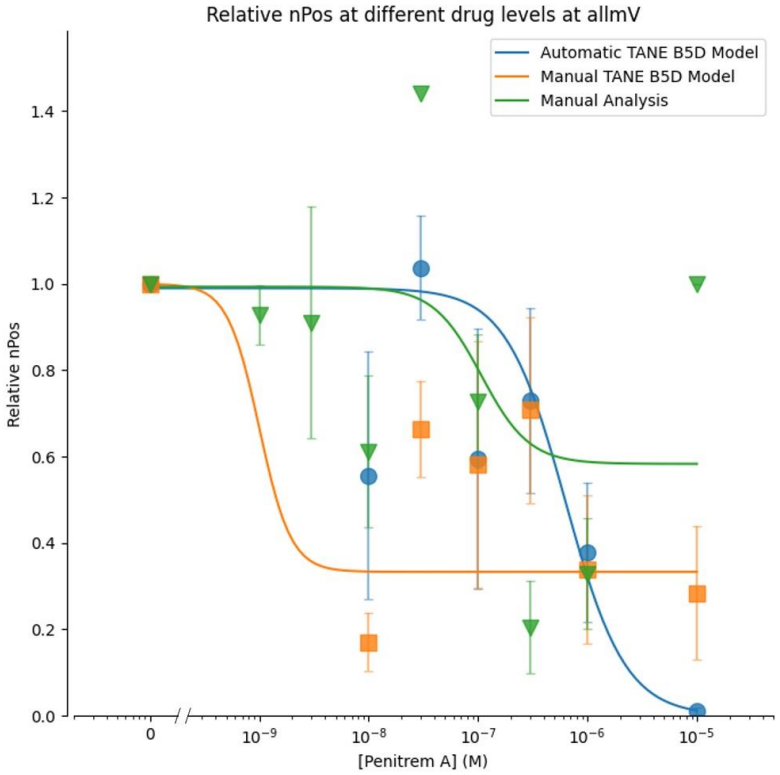


Figure 8.43: Relative nPos for different Penitrem levels, by different analysis methods.
Here we show 3 different analyses methods used for analyzing the Penitrem dataset – this dataset came along with analyses information, plotted here as "Fully Manual Analysis". Then the B5D model was used twice, once with automatic TANE and once with manual TANE. In all 3 cases we see a general downward trend in channel activation, with the trend being more visible in the model idealizations. DRC were fit with 4-parameter logistic relationships (Automatic TANE, plus B5D, Slope: -5.1±12.9, EC50: 7.8±7.1 (μM). Manual TANE, plus B5D, Slope:-3.8±0.4,, EC50:11.7±1.6 (μM)., Full manual (collaborator), Slope:-0.06±0.07, EC50: 5.3±642.6 (μM))

There was no significant effect of the drug on unitary conductance ($p > 0.05$, $n = 117$), and no significant differences in the relative *nPos* for each of the 3 models ($p > 0.05$) . Manual analysis with pClamp detected a relation between drug and *nPo* however neither of the deep learning models did (1-way ANOVA had $p > 0.05$ for both model methods and $p = 0.00478$ for the manual idealisation). Nonetheless, we attempted to fit dose response curves to nPos ~ concentration for each case. Figure 8.43 shows dose response curve analyses for the Penitrem action on BK (at membrane potential, Vm -60mV) using three different methods: (i) automatic TANE B5D model, (ii) manual TANE B5D model, and (iii) the original pClamp idealisation provided by the experimenter. The manual TANE method (ii) was unfittable in this way, but both methods (i) and (iii) were successfully fitted. Fitted dose response curves for the automatic TANE B5D model and pClamp gave

IC50s of 7.8±7.1(μM) and 5.3±642.6(μM) respectively. Interestingly the automatic TANE B5D model predicted a minimum nPos of near zero (at 10μM Penitrem) whereas the pClamp idealisation left a significant current present (approximately 0.5).

8.4.3.3 Kinetic analysis

For kinetic analysis I used the automatic TANE B5D model as it was the model that correctly identified a drug effect in the dose response curve. Since *nPo* was calculated as part of previous work on this dataset, the raw idealisations were not available and therefore kinetic analysis could not be completed.

As before the idealisations were used to construct dwell times histograms for open and closed states. Representative examples of these graphs are shown in Figure 8.44 for Automatic TANE B5D model at both 3 and 300nM Penitrem. As the data was analysed prior to the experiment taking place, the raw idealisations were not available and therefore dwell times could not be calculated for the manual idealisation method.

Examination of the dwell time histograms for the automatic TANE B5D model (Figures 8.45 to 8.47 and Table 8.5) shows a higher variance than the other datasets, however this is to be expected as the data is more challenging than the other two datasets, with more extreme noise present. Table 8.55 shows the statistical 2-way ANOVA tests for drug effect on the dwell time histogram parameters.

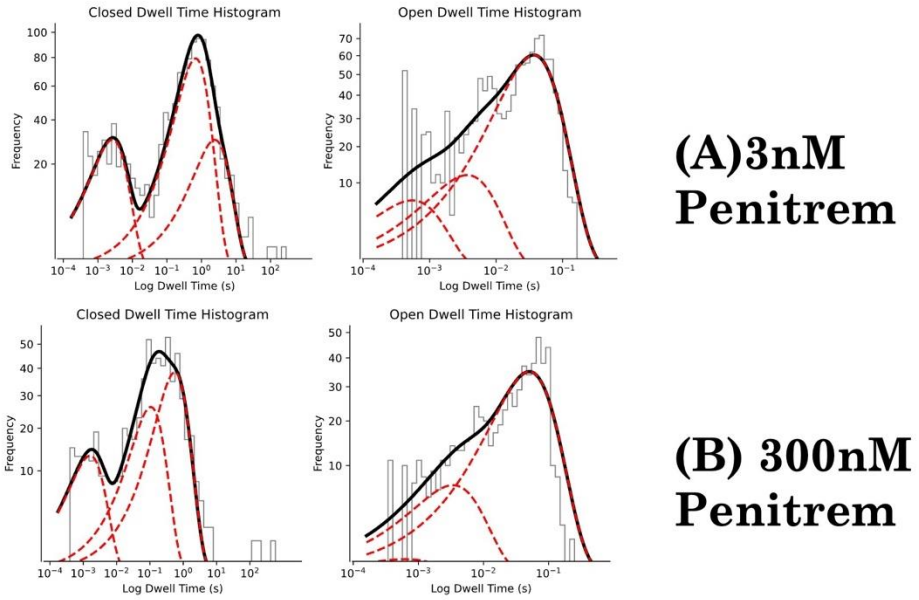


Figure 8.44: Representative dwell time histograms for both the model idealization at 3nM Penitrem A (A) and 300nM Penitrem A (B).
Showed is a representative model example for a recording at -60mV at 0.3nM Penitrem A (top) and -60mV at 300nM (bottom) fitted with 3 exponential curves. For curve fitting, both the area and tau (centre) parameters were recorded and summarized in the supplementary figure and figures 8.45 to 8.47 to examine if the drug had an effect on the underlying Markovian structure of the ion channel.

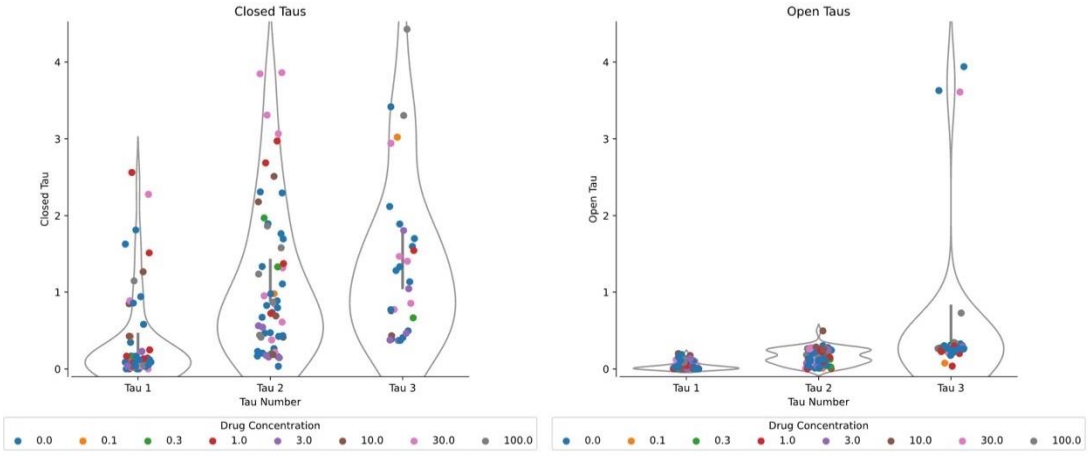


Figure 8.45: Dwell Time Parameter Plots for Closed and Open Tau Parameters for the model (Auto TANE B5D) idealisations at -60mV.
For the idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these tau parameters for the model idealisations. A two way ANOVA showed a significant effect of the drug on the closed taus ($p = 0.003$, $n=73$), but not the open taus ($p > 0.05$, $n=73$).

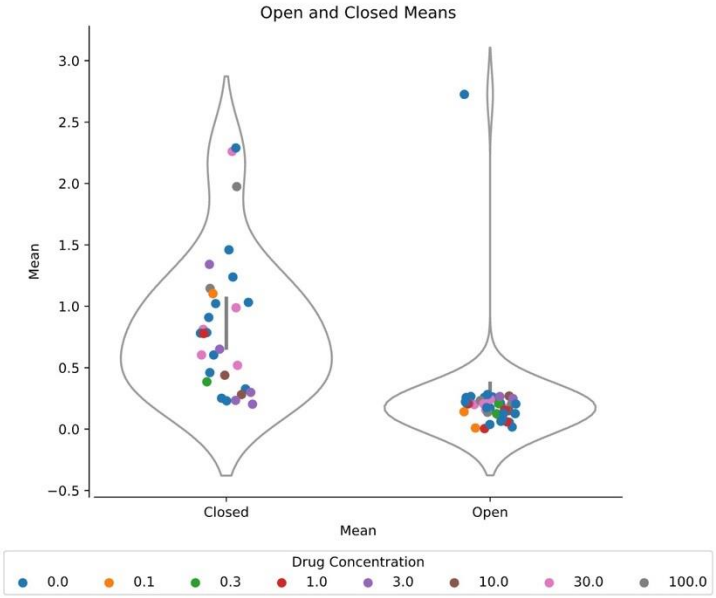


Figure 8.46: Dwell Time Parameter Plots for Open and Closed Mean Parameters for the model (Auto TANE B5D) idealisation at -60mV.

For the idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus). A two way ANOVA showed no significant effect of the drug on the means for the model idealization ($p > 0.05$, $n = 73$).

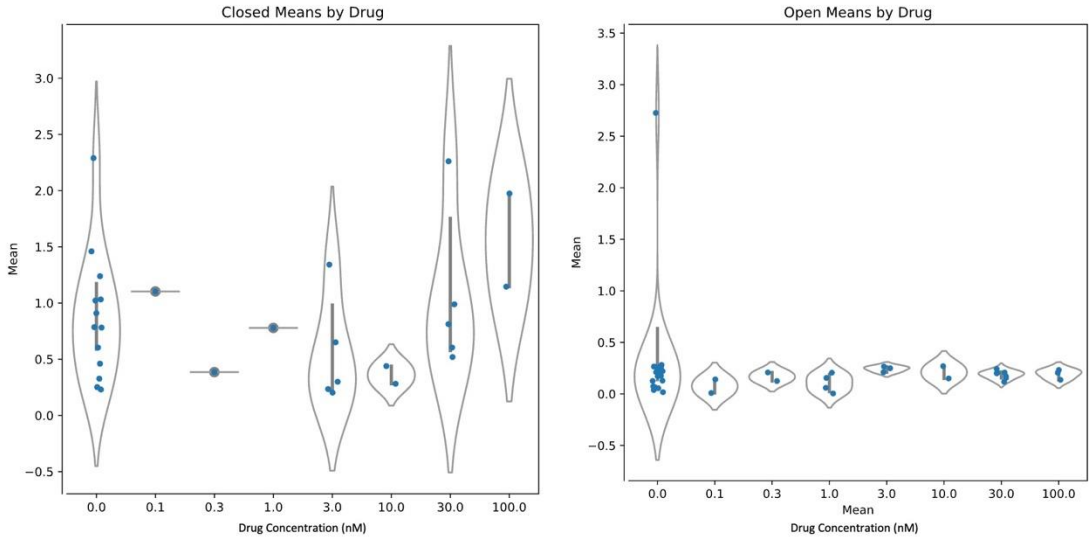


Figure 8.47: Dwell Time Parameter Plots for Mean Parameters for the model (Auto TANE B5D) idealisations by drug concentration at -60mV.

For the idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the mean parameters for the idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by voltage.

Table 8.5:Statistical tests for each group of dwell time parameters.
Each group of dwell time parameters (open taus, closed taus, open areas, closed areas) was tested via 2-way ANOVA (n =7272) for the effect of change in Penitrem concentration. Means were tested via 1 way ANOVA for the same effect.

Parameter	Model Penitrem Effect ANOVA
Closed Taus	P = 0.00307
Open Taus	P > 0.05
Closed Areas	P > 0.05
Open Areas	P > 0.05
Closed Means	P > 0.05
Open Means	P > 0.05

8.4.4 "Vernakalant" Dataset

I recorded the "Vernakalant" dataset using my protocols described in the methods. The dataset contained recordings of BK channels in control and 3 different concentrations of vernakalant hydrochloride at different membrane potentials (-40 to 40mV in 20mV intervals), under the more challenging OOP patch-clamp configuration. This allows direct exposure of drug to the extracellular face of the membrane/ion channels.

Data in the "Vernakalant" dataset was idealised using Automatic TANE and A5D, B5D and C5D models. Figures 8.48 and 8.49 show representative examples of subjectively good and bad idealisations with the B5D model. As with the "Calcium" dataset we found that while most of the data idealised correctly, there were clear cases of both false positives and false negatives. This was especially the case at 0mV data where the channel size was especially small, and the data was scaled larger through minmax scaling.

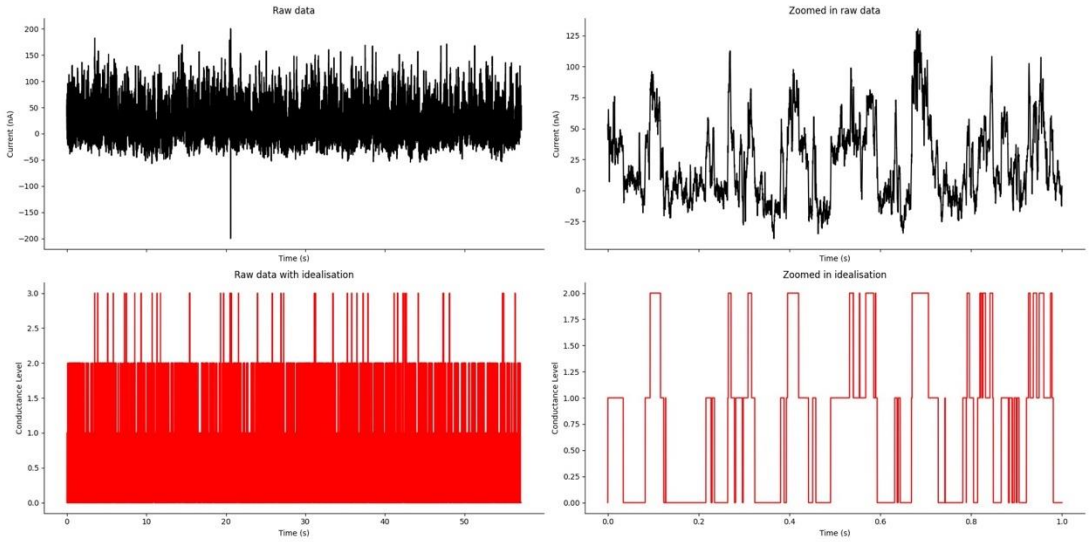


Figure 8.48: Successful idealisations from the "B5D" model for "Vernakalant" dataset.
Here we show a raw data trace from the "Calcium" dataset, and the corresponding idealization – we see that the model successfully idealises the record, successfully converting the raw noisy signal into a square wave.

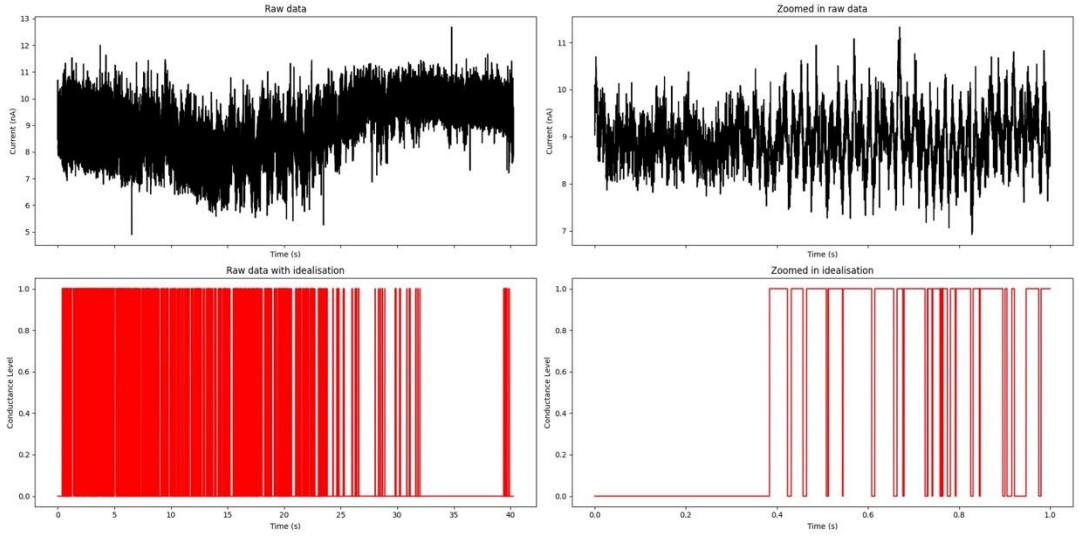


Figure 8.49: Unsuccessful idealisation from the "B5D"model for "Vernakalant" dataset.
Here we show an example where irrespective of metrics it is clear that the model has not idealized correctly. This can be caused by a number of issues, (see full text); but is likely fixed by further work into improving the quality and range of the training data, and the pre-processing pipeline.

8.4.4.1 Amplitude Histograms

I again created amplitude histograms and current-voltage measurements for each membrane potential (with and without the presence of vernakalant Figure 8.50). In

terms of voltage, there was a linear relationship between voltage and unitary amplitude ($p < 0.001$, $R^2 = 0.996$, n=70) and the slope conductance was 210+/-64pS (n=70). This was paralleled at each concentration of vernakalant used (230+/-73, 210+/-40 and 210+/-41 for 1,10 and 100nM respectively) and there was no significant difference in these (2-way ANOVA).

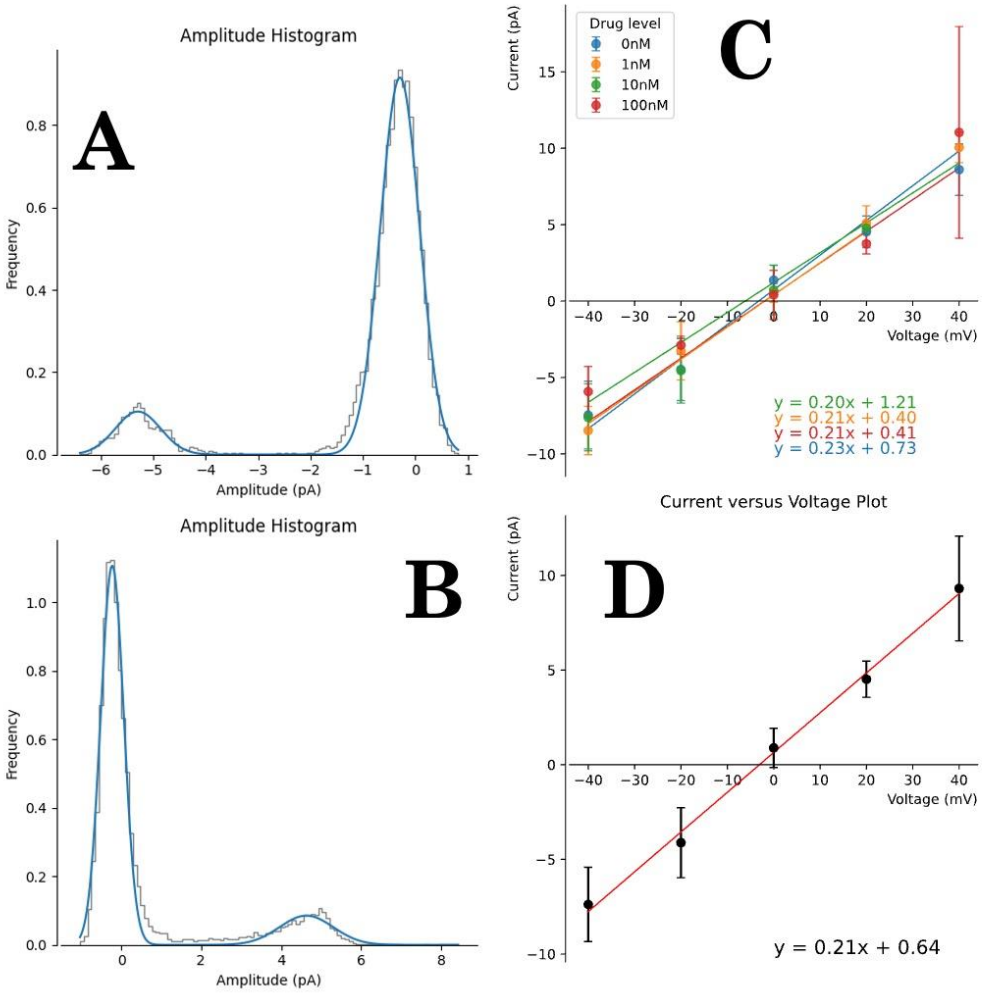


Figure 8.50: Example amplitude histograms, unitary conductance by voltage and unitary conducatnace by voltage, by drug

A and B show representative examples of current histograms for -20mV at 10nM of Vernakalant (A) and 20mV at 0nM (B), fitted with Gaussian curves. C shows the relationship of unitary conductance/channel sizes (adjusted to negative when channels open downwards) by the concentration of drug. ANOVA showed no significant difference between the drug groups effect on the channel size ($p > 0.05$, $n = 67$), however by aggregating all the drug levels and considering just the unitary conductance versus voltage relationship, we saw a statistically significant relationship ($p < 0.001$, R^2 0.996, $n = 67$).

8.4.4.2 Dose Response Curves

Dose response curves were then constructed for vernakalant on *nPos* for each of our 5 different membrane potentials with four different idealisation methods (i) The A5D model with automatic TANE, (ii) The B5D model with automatic TANE, (iii) The C5D model with automatic TANE and (iv) semi-automated idealisation with QuB. We found that all three deep learning models broadly agreed with the QuB idealisation, with no significant difference of *nPos* across the drug concentrations at any voltage (2 way ANOVA, $p > 0.05$). Hill (dose response) curves were fitted for each voltage and model (Figures 8.51, 8.52, 8.53 and Table 8.6). A significant inhibition was observed at -20mV by the UNet model ($p = 0.0088$), and at 0mV by the A5D and B5D model ($p = 0.0071$ and 0.0048 respectively). These effects were not noticeable at other potentials and manual analysis detecting no drug effect at any potential.

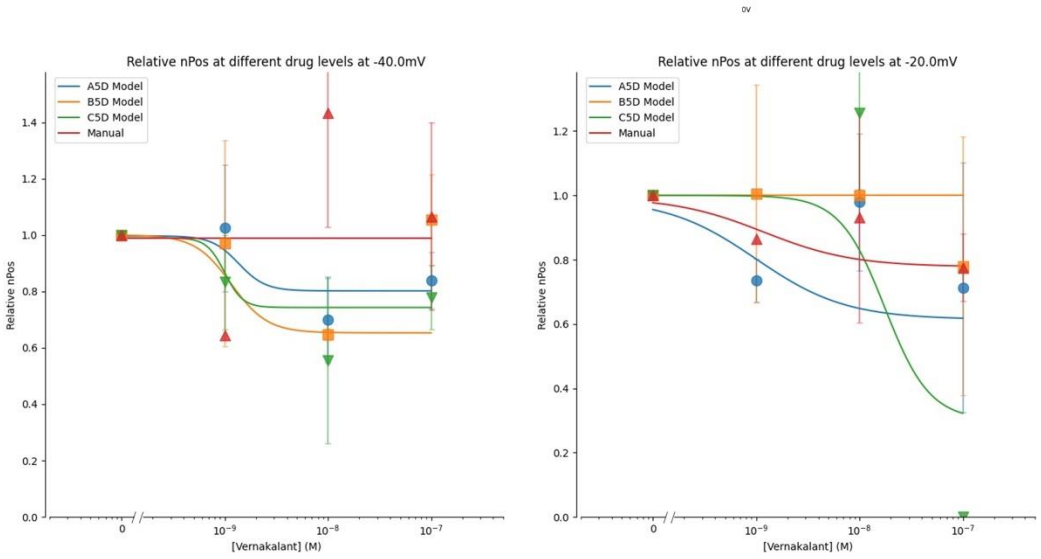


Figure 8.51: Relative nPo versus drug concentration as predicted by a number of different model architectures at -40mV and -20mV.
Here we show the different nPo analysis for the output of three different model architectures on the Vernakalant dataset, as well as a manual analysis using current methods. Across the 5 different voltage conditions, we only see three scenarios where a model detects a significant effect of Vernakalant via ANOVA with the B5D and A5D models at 0mV ($p = 0.007082$ and 0.004759 respectively, n=3), and the C5D model at +40mV ($p = 0.008759$, n=3). Overall however, a two-way ANOVA detected no significant effect of either the drug or the method used to idealise the data across all voltages ($p > 0.05$, n=3). For Hill Curve Analysis, EC50s for -40mV were 1.0±0.03uM, 6.3±1.5uM, 2.5±6.0e-02uM and 2.2±1.1 uM for the A5D, B5D, C5D and QuB analysis respectively, and for -20mV 3.5±94.34uM, 6.3±1.5uM, 2.5±0.1uM, 25.1±1.69uM for A5D, B5D, C5D and QuB respectively.

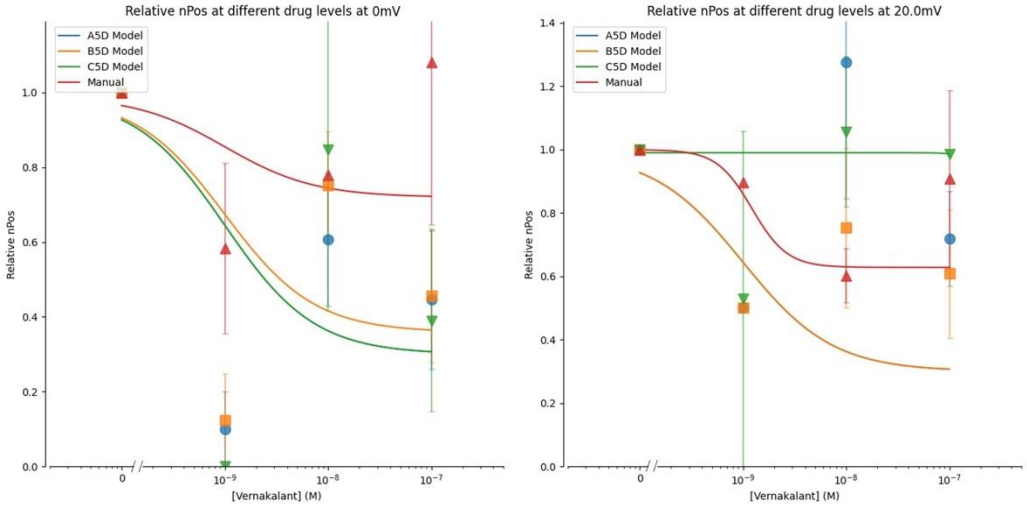


Figure 8.52: Relative nPo versus drug concentration as predicted by a number of different model architectures at 0 and 20mV.

Here we show the different nPo analysis for the output of three different model architectures on the Vernakalant dataset, as well as a manual analysis using current methods. Across the 5 different voltage conditions, we only see three scenarios where a model detects a significant effect of Vernakalant via ANOVA with the B5D and A5D models at 0mV ($p = 0.007082$ and 0.004759 respectively, $n=3$), and the C5D model at +40mV ($p = 0.008759$, $n=3$). Overall however, a two-way ANOVA detected no significant effect of either the drug or the method used to idealise the data across all voltages ($p > 0.05$, $n=3$). For Hill Curve Analysis, EC50s for 0mV were 4.2±1.6e-4uM, 7.0±1.5e-3uM, 6.6±7.5e-5uM and 2.1±8.6e-2 uM A5D, B5D, C5D and QuB analysis respectively, and for 20mV 4.5±0.32 uM, 7.0±1.5e-3 uM, 9.5±6.9e-3 uM and 2.4±1.8uM for A5D, B5D, C5D and QuB analysis respectively.

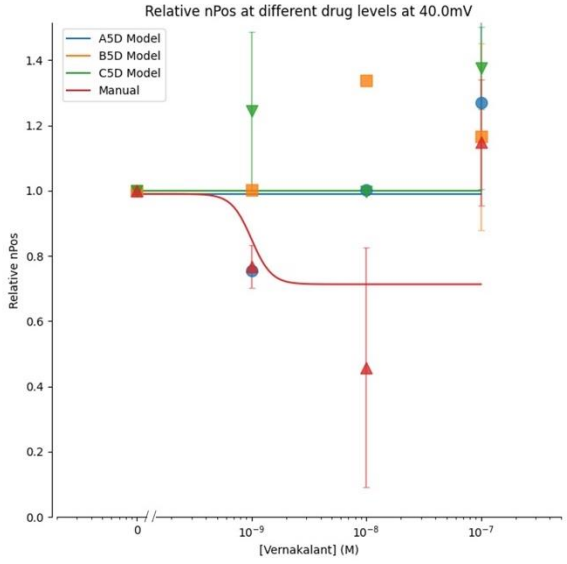


Figure 8.53: Relative nPo versus drug concentration as predicted by a number of different model architectures at 40mV.

Here we show the different nPo analysis for the output of three different model architectures on the Vernakalant dataset, as well as a manual analysis using current methods. Across the 5 different voltage conditions, we only see three scenarios where a model detects a significant effect of Vernakalant via ANOVA with the B5D and A5D models at 0mV ($p = 0.007082$ and 0.004759 respectively, $n=3$), and the C5D model at +40mV ($p = 0.008759$, $n=3$). Overall however, a two-way ANOVA detected no significant effect of either the drug or the method used to idealise the data

across all voltages (p > 0.05, n=3). For Hill Curve Analysis, EC50s for 40mV were 0.5±3.1e-5uM, 26.6±2.19e3uM, 1.1±0.015uM and 16.6±3.2e-2uM for A5D, B5D, C5D and QuB models respectively.

Table 8.6: EC50s calculated from dose response curve fits to each data idealised by each method at each membrane potential.
pVals are those for the null hypothesis that the EC50 intercept differed from zero. -- means it was not statistically significant.

Vm	**Model**	**EC_{50}±SD (μM)**	**pVal**
-40	C5D	2.5±6.0e-02	--
-40	A5D	1.0±0.03	--
-40	QUB	2.2±1.1	0.07
-40	B5D	6.3±1.5	--
-20	C5D	2.5±0.1	--
-20	A5D	3.5±94.34	--
-20	QUB	25.1±1.69	--
-20	B5D	35±3.9	4-e6
0	C5D	6.6±7.5e-5	2e-16
0	A5D	4.2±1.6e-4	2e-16
0	QUB	2.1±8.6e-2	4e-10
0	B5D	7.0±1.5e-3	2e-16
20	C5D	9.5±6.9e-3	2e-16

20	A5D	4.5±0.32	8e-8
20	QUB	2.4±1.18	--
20	B5D	7.2±1.4e-3	2.2e-16
40	C5D	1.1±0.015	5e-14
40	A5D	0.5±3.1e-5	2.2e-16
40	QUB	16.6±3.2e-2	--
40	B5D	26.6±2.19e3	--

8.4.4.3 Kinetic Analyses

From the automatic TANE B5D model and QuB idealisations, we took the dwell time histograms and fitted exponential curves as before, with representative examples shown in Figures 8.54 and 8.55. We assumed a three-state stochastic model again and the resulting open and closed taus (1,2,3 each) and respective areas are plotted in Figures 8.56 to 8.62. Table 8.7 shows a summary of the 2 way ANOVA tests completed on each group of parameters (open taus, closed taus, open areas, closed areas),

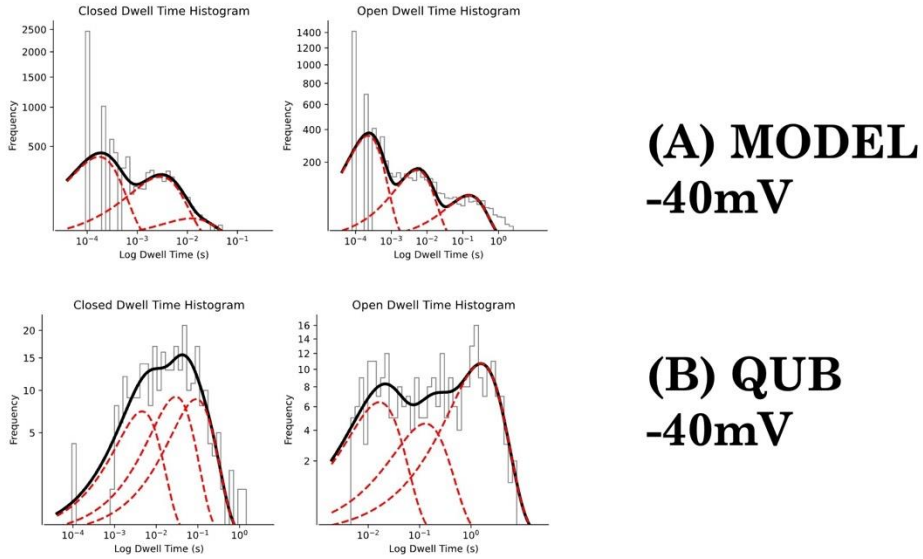


Figure 8.54: Representative dwell time histograms for both the model (A) and QuB (B) idealizations at -40mV.
Here we show a pair of representative examples of dwell time histograms at -40mV for both the model idealization and QuB idealisation. For all files, three exponential curves were fit to the dwell time histograms and their areas and centres (taus) recorded. Summarised fit parameters can be seen in the supplementary figures and figure 8.56 to 8.62

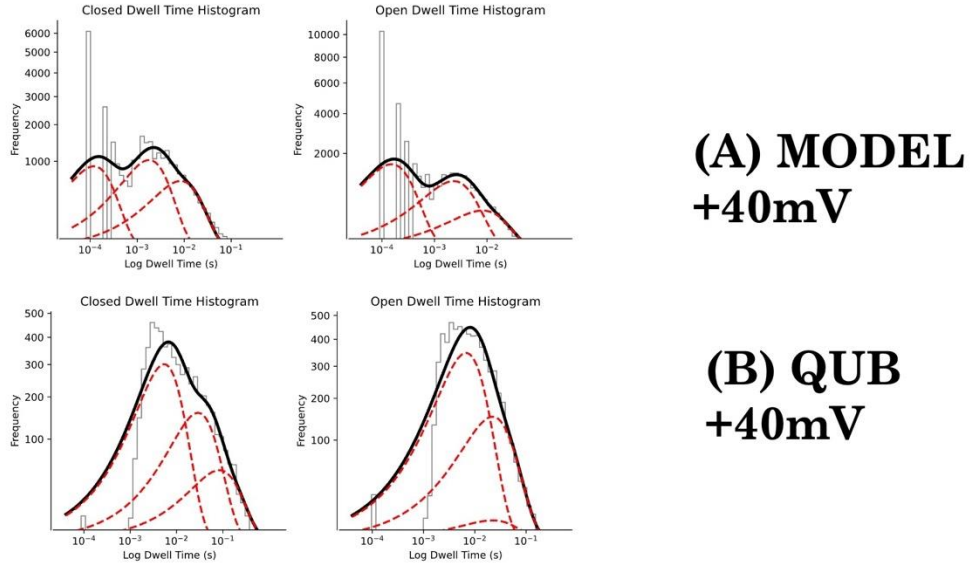


Figure 8.55: Representative dwell time histograms for both the model (A) and QuB (B) idealizations at +40mV.
Here we show a pair of representative examples of dwell time histograms at 40mV for both the model idealization and QuB idealisation. For all files, three exponential curves were fit to the dwell time histograms and their areas and centres (taus) recorded. Summarised fit parameters can be seen in the supplementary figures 8.56 and 8.62

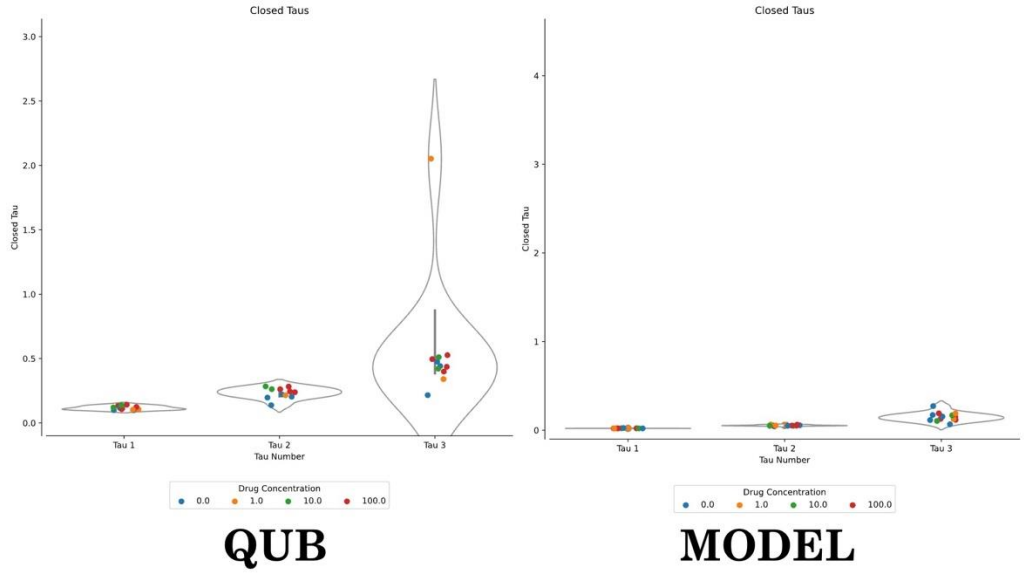


Figure 8.56: Dwell Time Parameter Plots for Closed Tau Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these closed tau parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the closed taus for either the model idealisations or QuB idealisations ($p > 0.05$, $n = 70$). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test ($p > 0.05$, $n= 70$). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters ($p < 0.001$, $n = 70$); however this does not necessarily imply one is better than the other.

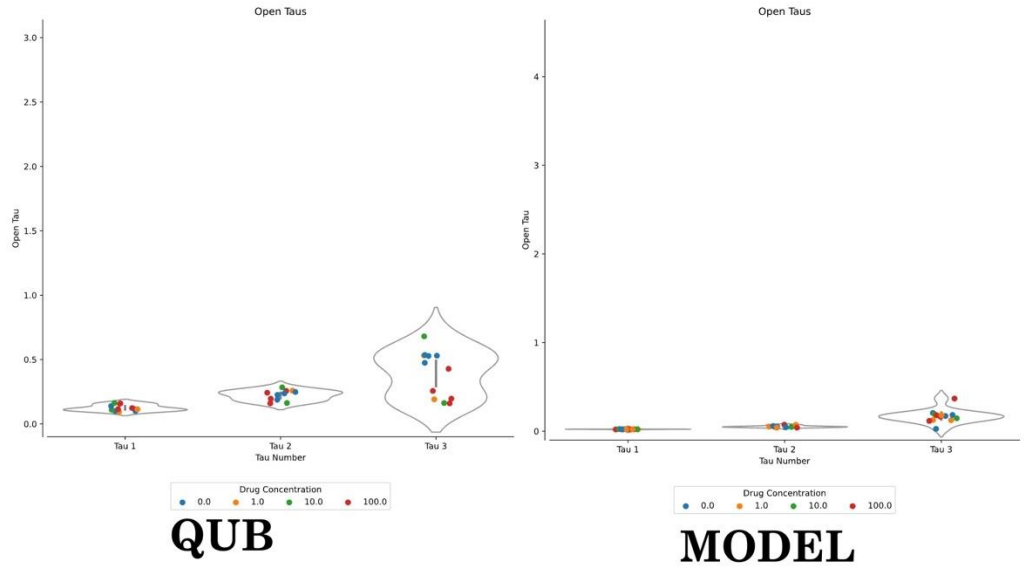
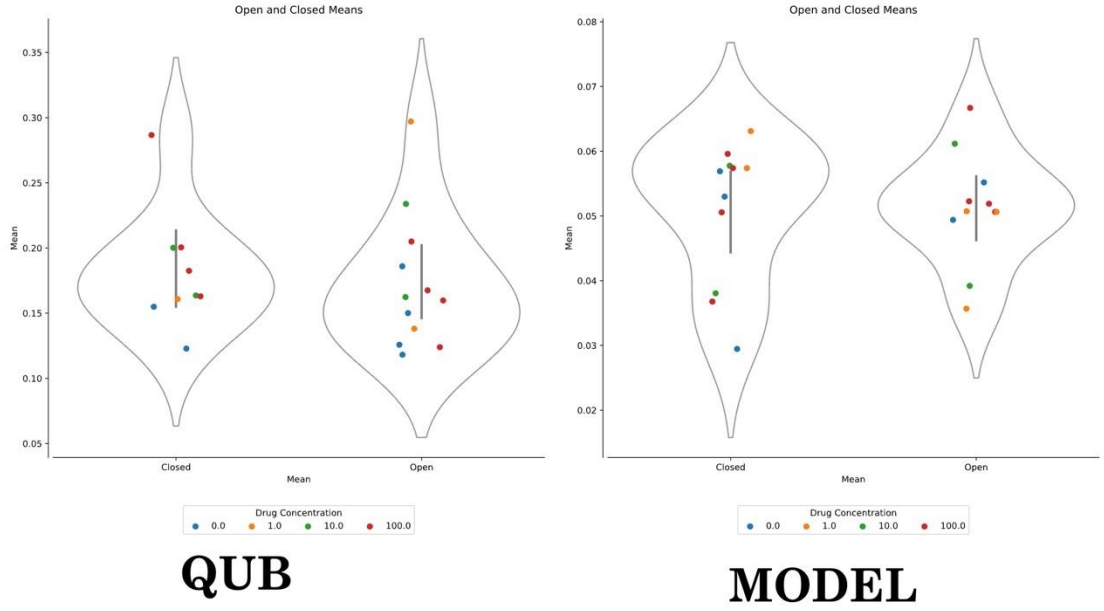


Figure 8.57: Dwell Time Parameter Plots for Open Tau Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these open tau parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the closed taus for either the model idealisations ($p > 0.05$, $n = 70$), but does show a significant difference for the QuB idealisations ($p = 0.0297$, $n = 70$). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via

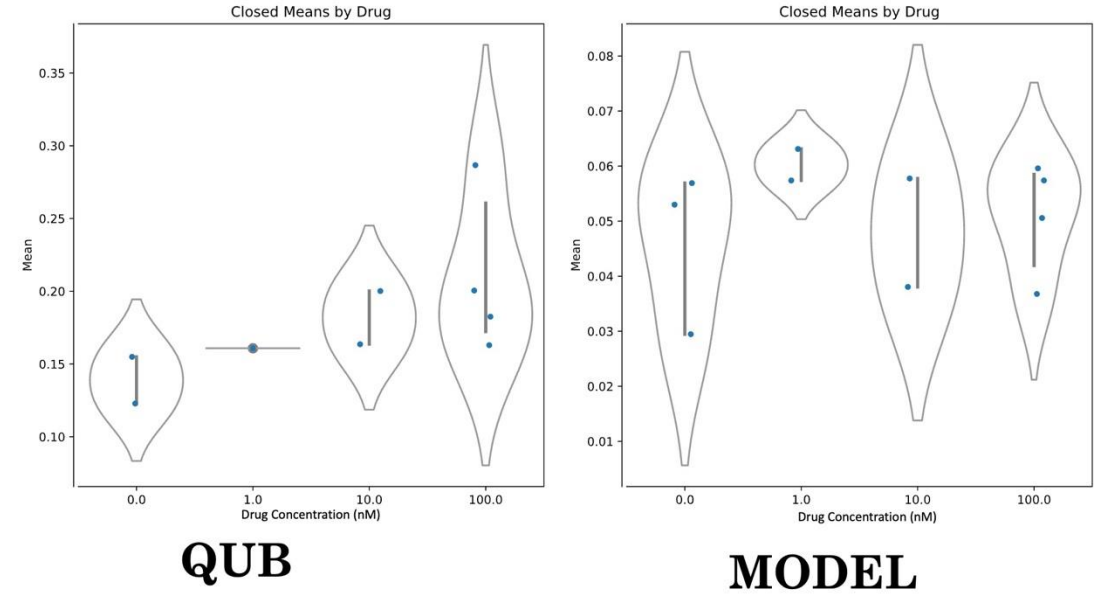
an F-Test (p > 0.05, n= 70). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p < 0.001, n = 70); however this does not necessarily imply one is better than the other.



QUB **MODEL**

Figure 8.58: Dwell Time Parameter Plots for Open and Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

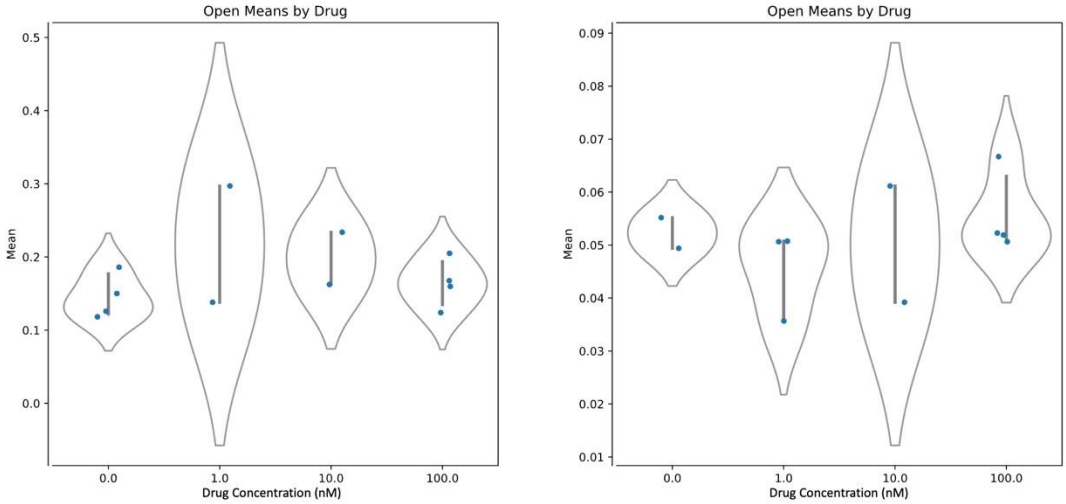
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus). A two way ANOVA showed no significant effect of the drug on the means for either the QuB or model idealisations (p > 0.05. n= 70). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test (p > 0.05, n= 70). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters (p = 0.0319, n = 70); however this does not necessarily imply one is better than the other.



QUB **MODEL**

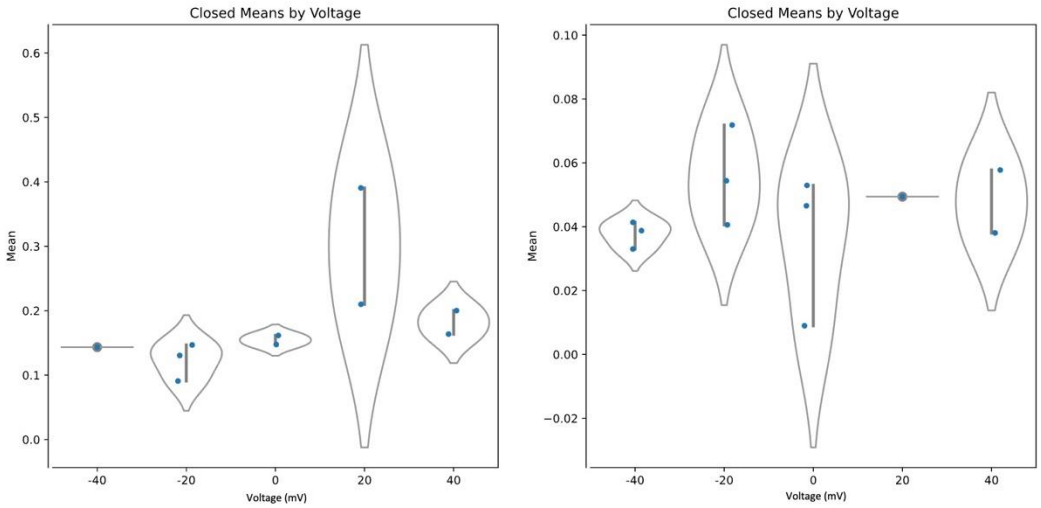
Figure 8.59: Dwell Time Parameter Plots for Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by Vernakalant concentration at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the closed mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by vernakalant concentration.



QUB **MODEL**

Figure 8.60: Dwell Time Parameter Plots for Open Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by Vernakalant concentration at 40mV
For both idealisations, the open and closed dwell times were recorded, and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the open mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by vernakalant concentration.



QUB **MODEL**

Figure 8.61: Dwell Time Parameter Plots for Closed Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by Voltage at 10nM Vernakalant.
For both idealisations, the open and closed dwell times were recorded, and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the closed mean

parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by vernakalant concentration.

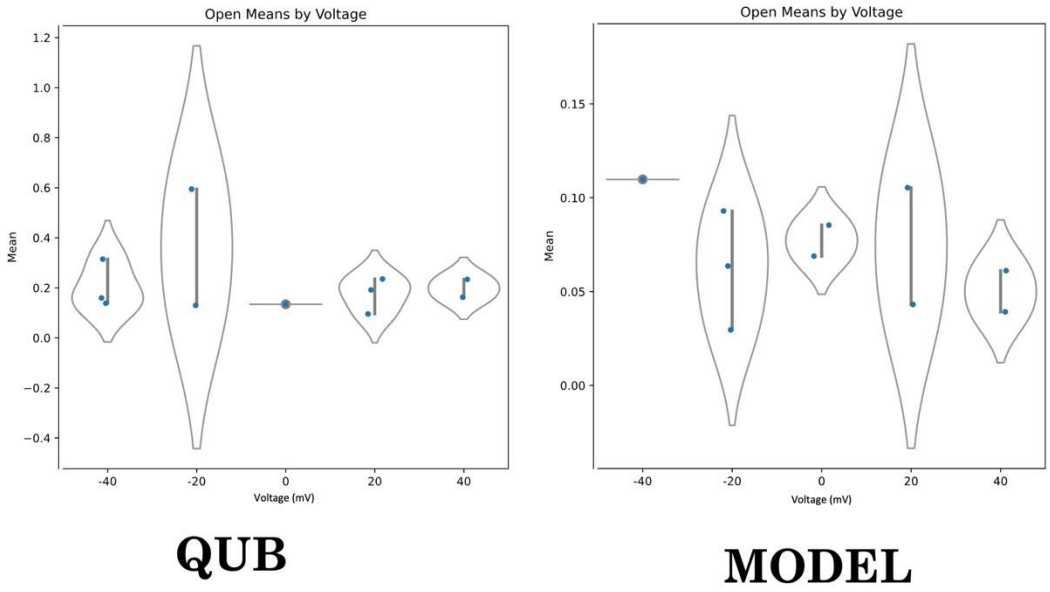


Figure 8.62: Dwell Time Parameter Plots for Open Mean Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations by Voltage at 10nM Vernakalant.
For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of the open mean parameters for both idealisations (that is, the sum of all areas multiplied by the corresponding taus) split by vernakalant concentration.

Table 8.7: Statistical tests for each group of dwell time parameters.
Each group of dwell time parameters (open taus, closed taus, open areas, closed areas) was tested via 2 way ANOVA (n = 1414) for the effect of change in Vernakalant concentration. Means were tested via 1 way ANOVa for the same effect

Parameter	QuB Vernakalant Effect ANOVA	Model Vernakalant Effect ANOVA
Closed Taus	P > 0.05	P > 0.05
Open Taus	P = 0.0297	P > 0.05
Closed Areas	P > 0.05	P > 0.05
Open Areas	P > 0.05	P > 0.05

Closed Means	P > 0.05	P > 0.05
Open Means	P > 0.05	P > 0.05

As with the calcium dataset, we tested the differences in the QuB and B5D model in the overall parameter means via a 2 way ANOVA, as well as the variances via an F-Test.

The 2-way ANOVA showed a significant difference between the QuB and B5D model dwell time histogram parameters ($p < 0.001$, $n = 70$), however as with the calcium dataset this does not necessarily mean one is better than the other. We also saw that in almost all parameters (apart from the "area 3" parameter for both open and closed dwell times), the B5D model showed a smaller variance, with a Fisher exact test (n=12) showing an overall lower variance ($p = 0.0033$)

8.5 Discussion

In this chapter, I measured the ability for a deep learning network to perform BK ion channel idealisation in three scenarios designed to represent a range of real-world analysis tasks. In each case results were compared it to the best current manual methods. Since there can be no genuine "ground truth" with such real analysis tasks, the first two datasets analysed ("Calcium" and "Penitrem") had functional "ground truth" outputs, in the sense that we have prior knowledge of how BK channels do behave in response to voltage, Ca^{2+} and Penitrem challenge. This discussion will begin by summarising the outcomes of each of the specific objectives given in the introduction 8.2, and then conclude by discussing the strengths and weaknesses observed for each method.

8.5.1 Calcium Dataset

A. Characterise BK channels, and their sensitivity to voltage and Ca^{2+} ions with a deep-learning based approach, with new, novel models trained on relevant datasets.

BK channels are a widely studied ion channel, this is why they were chosen for this study; furthermore in terms of size (unitary conductance) they are perhaps the "Goldilocks" ion channel in terms of signal to noise ratio. Ion channels can have a single channel conductance in the high femtosiemens (fS) range where even by eye in a clear record they are unlikely to be resolvable (certain, glutamate ligand-gated ion channels, resolved by noise analysis; (Howe, 1996)). Ion channels in the low pS range are quite common varying from for example 10pS to 50pS, for example L- and N- type Ca^{2+} ion channels (Snutch, Peloquin, Mathews, & McRory, 2013).. At the other extreme there are several ion channels in the nanosiemens (nS) range, such as bacterial mechanosensitive ion channels MscL (Haswell, Phillips, & Rees, 2011). The BK channel has widely published unitary conductances in the range of 100 to 300pS. In our study here we found in this initial dataset a single channel conductance of 137.7pS (Figure 8.28), which sits in the approximate average of this range. Previous work (Blatz & Magleby, 1987; Geng et al., 2020) shows the BK channel as both voltage and calcium sensitive analysed both of these properties with our novel model and analysed both of these properties with our novel model and using QuB. QuB was one of the most popular open-source packages available for single channel analysis, but is no longer maintained (we cloned this when it was available) and so a simple to use alternative would be useful, irrespective of any advantages new methods might bring.

In terms of Ca2+ sensitivity, both QuB and our B5D model showed a similar increase activity with increase in Ca2+ in the low micromolar range (Figure 8.29) . Data was limited (see caveats below), but nevertheless both models detected this with similar calculated EC50s in the low micromolar range (2uM and 5uM for B5D and QuB, respectively). This is entirely comparable to the literature with the dose response curve reported in Figure 6 of the ground breaking Barrett et al 1992 showing an EC50 of approximately 2uM, but a specific EC50 not quoted. (Barrett, Magleby, & Pallotta, 1982). Furthermore, Barrett et al show the dose response curve at 20mV where we have it at 40mV which will cause a small shift. Other work has reported greater sensitivity to Ca (ie lower EC50) that this, in the sub micromolar range (e.g. 31nM) (Numata et al., 2021))

indicating perhaps different behaviours between isoforms (Latorre et al., 2017). Although in a sense, the B5D model wins here reporting a slightly lower EC50 (showing a more sensitive analysis), paradoxically, the fall-off of BK channel activity is more evident after QuB analysis. It should, in theory (Barrett et al., 1982; Blatz & Magleby, 1987; Geng et al., 2020) all to near zero by 100nM (for an EC50 of 2uM), but B5D appears to be over estimating the residual current at low activity levels. This is broadly indicative of an issue with net false positives, or type I error and will be discussed further below.

In terms of voltage, the BK channel is closely related to the voltage-sensitive potassium ion channel family and retains profound voltage sensitivity. However, the pattern is complex with voltage sensitivity increasing with increased intracellular Ca2+. So for example in the original Barrett et al paper they show that with 1uM intracellular Ca, 50% activation of the channel occurs at 30 to 40mV, but with 10uM Ca2+ this rises to shifts leftward (more negative) by about 40mV. In terms of voltage, the BK channel is closely related to the voltage-sensitive potassium ion channel family (Kim & Nimigean, 2016) and retains profound voltage sensitivity (Barrett et al., 1982; Blatz & Magleby, 1987). However, the pattern is complex with voltage sensitivity increasing with increased intracellular Ca2+. So for example in the original Barrett et al paper they show that with 1uM intracellular Ca, 50% activation of the channel occurs at 30 to 40mV, but with 10uM Ca2+ this rises to shifts leftward (more negative) by about 40mV.

In terms of voltage sensitivity both idealisation methods detected the predicted activation at positive potentials (see Figure 8.30) and reported half-maximum activation potentials ($V_{1/2}$) in the region of +10mV, with 17μM present, showing agreement between the analysis methods. However; this relation lay to the right of that expected from the literature, where with >10μM present we may have expected half maxima ($V_{1/2}$) to be nearer -10mV (Blatz & Magleby, 1987; Geng et al., 2020). .This discrepancy could result from paucity of critical data meaning the fits were only possible at positive potentials and have been normalised; and represents a common theme throughout this work of unreliability of the manual analysis. There was not sufficient data available to produce

the 3D-dose response curve (x concentration, y, voltage and z is open probability) sometimes applied to the BK channel to show its complex calcium ion and voltage sensitivity. This is because whilst there were many recordings, one needs a combination of each condition to be present, together with a *maximum* or common condition to allow normalisation. An alternative approach would be to only use patches with one channel in, and then use absolute open probabilities. This was the approach used in the Barrett et al. study, but filtering in this way would have left a very small dataset, and partly defeated the purpose of this work which was automated analysis of *difficult* data.

We also conducted a kinetic analysis using the existing QuB method and novel B5D model. Channel kinetics are one of the critical ion channel properties that can only be derived using "single" channel patch-clamp recording and provides mechanistic insight (Sivilotti & Colquhoun, 2016) impossible with any other (current) technique. There were numerous kinetic analyses of BK channels early after its discovery and isolation (Cox, 2014; Lucchesi & Moczydlowski, 1991; McManus & Magleby, 1988; Yoshida et al., 1991) .This was extended to provide an understanding of the influence of different beta subunits (Contreras, Neely, Alvarez, Gonzalez, & Latorre, 2012) and more recently these data have been combined with structural data to give a detailed understanding of structure-function. In our own analysis we calculated dwell times for open and closed events with the two different methods QuB idealisation and B5D. We do this under exhaustive conditions and report Tau for a wide range of voltages and Ca^{2+} conditions. We found B5D to be more consistent across analysis, but this does not necessarily equate to more accurate. The shortest Tau were very tightly clustered in the B5D model and whilst there was not sufficient data for statistical significance there does appear to be a trend in the longer open time taus for larger values with higher concentrations of Ca^{2+} (Figures 8.29b, 8.29c).). One-way ANOVA was applied to look for the "functional ground truth" of kinetic sensitivity to Ca^{2+}/voltage exhibited in the literature and whilst both of our approaches largely missed this known effect, QuB did a little better (Table 8.4).

Probably the most established Markovian kinetic model is that of Magleby's group (Rothberg & Magleby, 1998; Shelley, Niu, Geng, & Magleby, 2010). This is highly complex set of 10 state models, however individual dwell time curves between these can our own can be visually compared. We did this at +40 and -40mV, at 18μM, whereas Shelly et al 20102010 show a range +70 and; 70mV (-70,-50,+50,+70mV) with 100μM or more Ca^{2+} and other more subtly different experimental conditions. Shelly is not specific about the exact idealisation protocol, but since this work follows on from Rotheberg & Magleby 1998 it is likely to be 50% threshold crossing (explicit in the latter). Starting with open time observations (Figure 8.63); Shelly et al see only one clearly distinguishable open time at any voltage despite the complex kinetic schema reported. They find however, that the mean of this peak increases with increasing voltage from around 0.1ms at -70mV to approximately 1ms at +70mV. Our data with B5D idealisation, however, whilst similarly fitted with multiple exponentials shows two clearly distinguishable peaks at -40mV; one at about 0.1ms another near 3ms (Figure 8.32), perhaps suggesting different recording conditions or behaviour between isoforms. By +40mV our (B5D) fitted data shows three clearly distinguishable peaks with Tau of around 0.1,1 and 100ms. Interestingly, QuB idealisation delivers a longer open states than either Shelly or B5D, with a criticised deadtime issue (Prof David Colquhoun personal discussions), where the nature of the of the output leads to all events being exact multiples of the sample interval. With QuB data, by +40mV there is a single clear open time peak, but it is evidently longer than that of the B5D model and some 10x that reported by Shelly (and note Shelly used higher Ca^{2+} which would be expected to increase open times). If Shelly et al and Rothberg et al are treated as functional ground truths, the B5D kinetic data do appear subjectively a little closer than the QuB data. Direct comparison of the closed times between our study and previous kinetic analyses are not as useful as comparisons of closed times, because closed times (especially longer ones) are strongly dependent on number of channels present in the patch. Nevertheless, again, B5D has detected the presence of a substantial short events missed by the QuB methods. Furthermore, I would direct readers to Figure 8.63, reproduced from Rothberg et al and Shelly and invite comparisons with our raw figure

data in Figures 8.26, 8.40 and 8.48. In these previous studies, the single objective was to record very short sections of very high-quality data; manually quality controlling, choosing only records with demonstrably only one channel present etc, all to optimise for single channel kinetics. In the present study we, by design, used all the data that was recorded without any more than the most rudimentary quality control. This was to test the software in non-ideal conditions, since it was felt the principle of deep learning idealisation had already been proven with high quality curated data already.

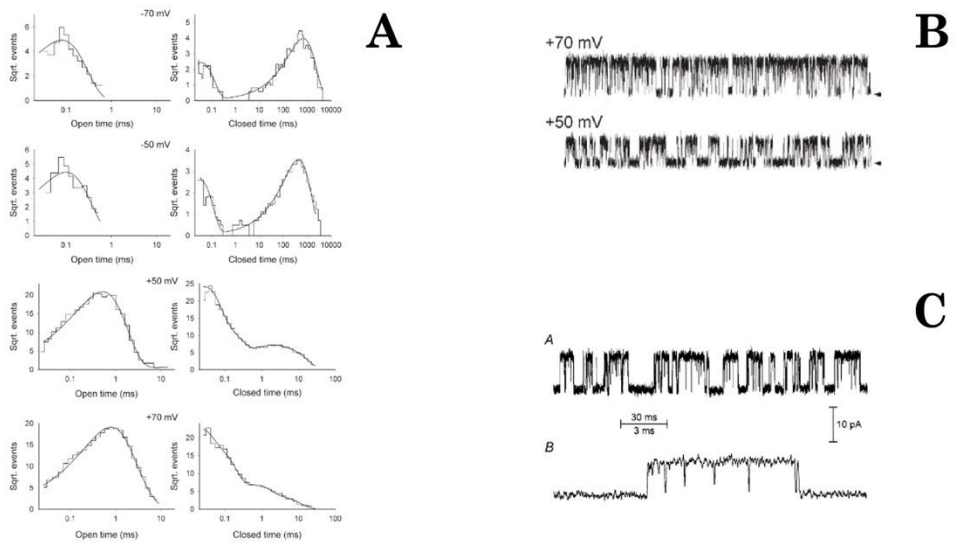


Figure 8.63: Kinetic Analysis and example traces from (Rothberg & Magleby, 1998; Shelley et al., 2010).
A shows Shelly et al. is a noteable work investigating the kinetics of the BK channel where it is seen that the one open time exponential is seen to shift to the left (and increase in mean open time) as voltage increases. Our work shows more visible peaks in the dwell time histograms for the deep learning models, and the manual analysis producing longer open dwell times than any of the dwell time peaks in Shelly et al. This may be to do with differences in recording conditions, such as an increased concentration in Ca2+ in Shelly et al's work. If this previous work is to be taken as a ground truth, then there is some arguments to me made that the B5D model is more accurate than QuB for kinetic analysis of this type. B and C show raw data samples from Shell et al and Rothberg et al – showcasing the clean nature of this data compared with the particularly challenging nature of our data seen in Figures 8.26 8.32 8.38.

8.5.2 Penitrem A sensitivity of BK channels.

In this experiment, we again start from knowledge that BK channels are inhibited by extracellular Penitrem A. In this case, an independent analysis and idealisation of outside-out patch clamp recordings was performed; a deep learning model should not only

detect a decrease in open probability as the concentration of Penitrem A increases, but ideally do this faster, objectively and have a stronger discrimination than the manual analysis.

Whilst the channel (BKaB1) and recording mode here were the same; the data was collected by a different experimenter on a different "rig" in a different lab (see methods). In many disciplines it may be expected that such differences should be inconsequential, but patch clamp is very much a skilled manual procedure and so it is possible results would be different. I again found that single-channel amplitude followed a broadly Ohmic relationship similar to in the "Calcium" dataset above and the published literature . The unitary slope conductance was a little larger (150pS) than that observed above, but still well within the range of that seen in the literature (Lee & Cui, 2010). Dose response relationship of Penitrem has been performed before and revealed that the efficacy is different depending on the isoform of BK channel, in our case we had BKa/B1 which Asano et al (2012) reported to have an IC50 of 64nM (Asano et al., 2012) and an almost irreversible nature (approximately 1nM in the absence of the ß - subunit). It was active from either the inside or the outside of the channel. We next compared the amplitudes of channels recorded at -60mV, across a range of Penitrem concentrations. As expected there was no change in amplitude with Penitrem.

We have three sets of relative *nPo* dose response curve parameters; two for our favoured B5D model (with and without automatic TANE) and one for the QUB analysis. These were in the range of 5 to 12uM, but noting the manual TANE B5D model method performed rather badly (orange line, Figure 8.43). In this sense the automatic TANE method shows great promise as a method to achieve a sensible dose response curve with the B5D model. As is typical of single-channel recording the dose response curve have high variance. This is quite unlike what one might expect with a whole-cell dose response curve where potentially hundreds of ion channels are averaged together. In the case of single channel recording an individual channel can naturally be open or closed at any concentration and so much longer periods of recording necessary to get the variance down.

Nonetheless, given that we know the expected Hill relation a priori we found these were fittable using conventional curve fitting algorithms, with initial conditions set as the expected Hill parameters. .Note the QUB idealised data, kindly supplied by a group member for this project showed nPos that did not fall below approximately 0.5% whereas our model had a dose response curve minimum of near zero nPos. In fact close inspection of the Asano et al paper shows that although Penitrem acts rather slowly it would be expected to reach neglible nPo at 100nM of Penitrem (which is approximately that we used). Whilst the focus of this thesis is novel application of deep learning methods, a further potential flaw is the biological experimental design of the experiments curated for this project. In addition to the high variance (probably) due to low sample numbers, for an expected IC50 of 65nM it may have been better to use the dose range from the sub nanomolar to the μM range. It is a common pharmacological observation that in the vicinity of the IC50 variability is highest. This can be over-come with sufficient numbers, but ideally the data would have included a number of patches that ran from 0, 0.1nM to 10uM. Furthermore, the near irreversibility reported by Asano et al 2012 (reference) creates complications for dose response curve analysis. We do not at this post-hoc analysis have information about whether the cells were replaced between each experimental run etc., If that was *not* done it would confound the analysis, and weaken the usefulness of these data as functional ground truth. It should be remembered that our novel analysis approach is however, directed towards industry where multi-parallel equipment would make curation of much greater datasets possible.

We could not find a thorough kinetic analysis of Penitrem A's effect on BK channels, but in our study here, the B5D method appears to reveal a change in kinetics (Figure 8.44, Table 8.4); and this would be similar with that seen when BK channels are inhibited by U-37883A (Teramoto et al., 2004). It would fit with a model where the channel is subject to flickery block or the shorter events are promoted by an allosteric interaction. These would be interesting observations to pursue in a follow-up study.

8.5.3 Vernakalant Hydrochloride sensitivity of BK channels

In this experiment, we start with no prior knowledge of what effect Vernakalant will have on this isoform of potassium channel. In this section of this work, the channels and cells were the same as for the Calcium and Penitrem A Datasets above, but again collected on a different "rig", this time by myself. Also, this used the most difficult, or involved, form of patch clamp, "outside-out" patch, where cells are first giga sealed, then the membrane is ruptured and then the patch pulled off. The additional steps leave more to go wrong and therefore make the acquisition of data harder. Furthermore, since the process of pulling the patch of membrane from the cell follows a few minutes after making the initial seal, it is logical to suppose seals are less good and recordings would have greater noise (since noise is proportional to 1/seal resistance). From the literature, it is clear that the primary action of vernakalant in heart disease is as an inhibitor of sodium ion channels (Burashnikov et al., 2012; Naccarelli et al., 2008; Seyler et al., 2014),, but there is also information to suggest it also has an inhibitory action on a type of (non-BK) atrial potassium channel. We thought it was therefore important to investigate if it also inhibited BK channels which, as described previously, are a somewhat ubiquitous protein.

Observationally, there were fewer issues with model collapse with this dataset, however there are still a number of obvious errors in idealisation. In terms of unitary conductance the values are considerably higher at around 200pS, still entirely compatible with the literature, but presumably larger due to the recording conditions having a lower concentration of Ca^{2+} in the bath solution. Again there was no difference in amplitude with drug concentration. In terms of dose response analysis; most conditions (with most models) did report a statistically significant dose relationship, but this was clearer at +ve membrane potentials. This would be that expected with a small negatively charged drug (accessing the pore), since electrostatic forces would tend to drive it into the aqueous pore and sterically inhibit the channel; but this would not be expected for Vernakalant which is a rather lipid soluble compound, with low water solubility (drugbank.com) and largely uncharged (Alagem, Dvir, & Reuveny, 2001; Hille, 1978, 1992; Hurst, Latorre, Toro, &

Stefani, 1995). In terms of kinetic analysis, we see the model detects no significant effect of Vernakalant on the kinetic behaviour of the channel, with QuB only detecting a significant difference in the Tau parameters. The DL models again report a greater number of very short events than QuB; this may be due to flickering classifications, an artefact error from training.

In summary, from a biological perspective, we do detect a significant dose effect, but it appears rather small, especially given that we went up to 100uM, about 10x that used in clinics (drugbank.com). That said these types of screens are critical in pharmacological research since over-doses do occur and even subtle disturbance in ion channel function can be dangerous.

8.5.4 Strengths and Weakness of Idealisation methods

The main advantage of using these deep learning models over QuB is that they can be completely automatic. All of the pre-processing has analytical methods for automatic parameter detection, and (with an appropriate standard model) the model prediction requires no user input. In fact; the entire process from signal to summary statistics such as nPo, dwell time histograms and amplitude histograms can be fully automated; and (with a few exceptions where manual fixing had to be done), this was how the data was analysed in the work above.

While this system can be fully automatic; there is plenty of room to improve initial idealisations by manually changing either the pre-processing parameters, or the deep learning model itself; this is particularly important when considering improvements in automatic patch clamp systems; if an end-to-end pipeline can be established for patch clamp experimentation to analysis, the drug discovery process speeds up significantly. With this system, it is possible to generate bespoke datasets for the given ion channel if the Markovian network is known *a priori*, train a deep learning model on this dataset, and use it to idealise new data from recordings. For our work; this was not needed, and in

fact the Markovian model used for simulation was the "simple" 3-state model rather than the more complex, 5-state model with parameters taken from the literature; showing the robustness of the model to Markovian changes. This is particularly important for cases where the Markovian transition rate matrix may change due to the application of a drug, or a change in conditions.

The choice of model forms a central trade-off for this model, with the ability to sacrifice applicability to a larger number of ion channels for more accurate idealisation. In the case of this work, no channel record had visible number of channels greater than five, so the five channel model provided a good balance between being able to detect the maximal number of channels, and accurately idealising what we had. A more sophisticated approach could involve training multiple models for each number of channels visible in the data; however this depends on the robustness of the channel detection algorithm, and increases the overall model size dramatically, as one model is needed per number of channels opening.

This flexibility however forms a double-edged sword; the model is large and relatively slow compared to current methods computationally, although still significantly faster when considering human input is needed for semi-automatic idealisation. The model (and accompanying pre/post-processing code) is several gigabytes large, and realistically requires significant GPU processing power to run; this is not uncommon for deep learning models however.

In some cases, particularly when the data was very noisy or had a significant baseline drift present; the model would produce an idealisation that was either all-open or all-closed; or very few events present. This was common during testing of the models and was fixed not by adjusting the model itself, but the pre-processing parameters. One major drawback of deep learning models is that as the testing data gets more dissimilar to the training data, the performance will drop; overfitting causes this problem in the extreme however it is always present to some extent. Due to the diversity of the recorded data, it is easier to change the pre-processing parameters to make the input data more like the

training data than the other way around, which would involve retraining of said models. This is done by intelligent selection of the decimation parameter to make the dwell times in line with what was seen in the training data.

The dissimilarity of the training data to the lab recorded data is a necessity; deep learning is a supervised learning method where labels are required for training. By simulating the data, we can achieve accurate ground truth for records on demand; one alternative to this method that would make the training dataset better reflect the application would be to use human labelled data for the ground truth. This comes with two significant considerations; firstly there is always some disagreement between experts when it comes to idealisation as there is some subjectivity to deciding what an "event" construes; and it is also extremely time consuming to obtain the amount of data needed for training a deep learning model. In the previous DeepChannel work, simulated data was passed through an analogue amplifier; although this gave an objective ground truth along with data with realistic noise, the data was still collected in real-time, resulting in a long feedback process for development. In comparison, with this method hours of data can be simulated in a matter of seconds; allowing for parameters to be changed quickly to improve dataset quality. Throughout this work, the simulation algorithm has been improved, and continuous improvements to improve the realism of the simulated data will reflect in better model performance.

Comparison with QuB poses an additional problem; it is not a comparison to a "ground truth" *per se* but a comparison to an already established form of analysis; the disagreement between the QuB and the models' nPos in the "Calcium" dataset for example only tell us that the type of errors present in the deep learning idealisation are consistent within themselves, but altogether different to those present in the QuB idealisation. This is not to say that either idealisation is incorrect, but as the approach to idealisation between the deep learning models and QuB models (SKM) is markedly different; it's perhaps unsurprising to see some disagreement here. The agreement between the models implies that the same issues were arising irrespective of the model chosen; this was why

only the "B5D" model was taken forward to the other datasets; as performance was so similar between the models.

A significant problem during pre-processing was the issue of scaling. By default, min-max scaling was employed as a way to normalise data and channel "size" during training; however this ran into the problem of different raw data signals having different number of conductance levels, and therefore min-max scaling resulted in vastly different channel sizes. Ideally, a general model should be channel-size agnostic, only considering the significant changes in the data stream; however this proved extremely challenging. A number of different scaling methods were attempted, including scaling the mean and standard deviation of the data to 0 and 1 respectably (this did not help the "channel size" problem;); and scaling the data on a "per file" basis by using the distance between amplitude histogram peaks to force the mean channel size to be a certain value (this did not work due to noise and inaccuracies in the amplitude histogram peak fitting process). These sophisticated methods, while perhaps more logical than the min-max followed by scaling to the number of channels open, were significantly worse when used as inputs to the deep learning networks. Moving forward, we anticipate improvements in pre-processing to hold they key to better model performance.

9 Discussion

This project aimed to investigate, develop and test novel deep learning methods for single channel patch clamp analysis. We successfully adapted well known convolutional network models to recover, for the first time, continuous Markov states from simulated ion channel data using deep learning. Furthermore, we produced models that considerably outperformed the existing state-of-the-art deep learning analysis model, DeepChannel, which had already been shown to outperform shallow learning models in some circumstances. In additional experiments we developed a deep learning method for synthesis of ion channel data, given a small amount of "seed" data and tested a number of novel models on real world data.

Despite this success, there is still a substantial amount of work needed to optimise pre-processing and model architecture; and developing ways of deploying these such models in user-friendly frameworks for the practical use by ion channel researchers.

9.1 Data Sourcing

9.1.1 GAN Data

Our novel work here (Chapter 3) shows for the first time that generative adversarial networks (GANs) can create synthetic, fully labelled data that has characteristically specific qualities depending on the kind of ion channel used for training, and this approach inherently has several advantages and disadvantages; but as with the classification problems these are often intertwined.

The GAN produced data has been shown to be characteristically similar (via T-SNE and UMAP methods) to the real data it is trained from, and also characteristically different to data from different sources. This is of interest as generating large amounts of "similar" data to a real ion channel is desirable for training humans how to idealise ion channels, or even other AI models. Moreover, this data is anonymous and generatable on demand;

albeit more slowly than other methods when including the training process; meaning that the quality of training is likely to be much higher than other methods with more data available.

Our GAN model had limitations, however, for example the only form of control we have on the output data currently is the input data we use to train the model. Metrics such as open probability, channel size, drift and noise cannot be controlled by the user for pretrained models; so simulating slight changes in conditions, or adjusting the data for training another model is difficult, compared to, for example, equivalent stochastic models (O'Brien, Feetham, Staunton, Hext, & Barrett-Jolley, 2022). One potential solution to this would be to use conditional GANs (Mirza & Osindero, 2014) to attach some control on these metrics during model training. In conditional GAN networks, an additional input (such as nPo) is added to the generator, corresponding with some information about the data. After training with this additional input, it can be used to control characteristics about the output from noise via changing this value manually in the generator.

9.1.2 Markovian Simulation

While our GAN models represented a successful *data driven* approach to data generation; the limitations meant that we felt for the rigorous testing we needed for our novel deep learning model; there would be an advantage in using an updated Markovian simulation method. This is the approach usually used in the literature (Bruno et al., 2005; Gnanasambandam et al., 2017; Qin, 2004) and possible since we know many of the properties of ion channel recordings (such as they follow an underlying Markovian mechanism, contain some form of 1/f noise) that allow us to build a mathematical model to generate signals. Therefore, most of this thesis used Markov models to simulate data, and added synthetic noise to the data through a series of "noise layers" that emulate the noise observed on real ion channel recordings (e.g. comparing Figure 5.3 and 8.38), .

These noise terms are highly customisable and extendable; from simple Gaussian noise to 1/f noise that scales based on the number of channels open and the time since the last

opening (Howe, 1996; Sigworth, 1985; Yu, Dhingra, Dick, & Galán, 2017). This control over the noise on the signal was used throughout this work to ensure the simulated ion channel data was as close as possible in shape to the real data recorded from the lab; some variation was built into the noise parameters to cover a wider range of recording situations; resulting in a range of signal to noise ratios and datasets with and without baseline drift.

The Markovian simulations for the Markovian state recovery work used two Markovian networks; a simple 3-state model designed as the simplest non-trivial Markovian scheme as a baseline, and a 5-state Markovian scheme taken from the literature from a real ion channel analysis fit. The 5-state model produced data that had a far larger range of dwell times; the mean dwell time for one of the samples was only a few samples long, whereas for another state it was thousands; which caused difficulty in training.

In the idealisation work, multiple copies of these Markovian simulations were added together before noise was added in post; initially the 5-state model was used but it was quickly found that due to the nature of the Markovian model, it was rare to have two channels open simultaneously, and in cases where it did, the dwell times were so short (single point) that models would not idealise the bursts at all. Therefore, the simple 3-state model was used and parameters manipulated to make the channel more realistic compared to a sample of lab recorded data, with clear, isolated events with appropriate dwell times. This created balanced, higher quality datasets, but the underlying Markovian model was unreasonably simple. In our testing in Chapters 5, 6 and 7, it was found that this gave generally better results than models trained on the 5-state data. Presumably this is because models would encounter a narrower range of dwell times (instead of the range of single-point to large dwells of no activity), which were easier to "learn" for the neura networks.

9.1.3 GAN vs Markovian Simulation

The GAN generated data showed a novel and promising approach to data synthesis, however, as stated above, it was not used for the subsequent work on Markovian channel recovery or channel idealisation. This was mainly due to two reasons: lack of a reliable ground truth for the Markovian case, and lack of control on event frequency. It was critical in Chapters 5,6,7 and 8, to have a wide range of training data "phenotypes" as possible or the classic deep learning pit fall of over-fitting would likely occur (Aghajanyan, 2017; Zhang et al., 2021)

The DeepGANnel, our GAN channel synthesis method is designed to produce idealisation in parallel to ion channel data (this was the key breakthrough), however, this idealisation may not be perfect, since, by definition it would reproduce the same wrongly labelled events that the humans did that labelled the seed data in the first place. Therefore, using these labels as training data for the deep learning models of Chapter 5, 6, 7 and 8 could potentially introduce new errors into the system. In other words, the new models would not only learn the true nature of the underlying data, but also learn the inaccuracies from the GAN or human labelled data used. This is the major advantage of using a mathematical simulation; we can ascertain an accurate, objective ground truth for every point in the signal. This allows the model to form its own opinion of the dataset rather than rely on a reproduction from a third party.

Secondly, the GAN data does not allow us to increase the event frequency arbitrarily to create a more varied dataset. Deep learning models are infamously "black boxes" (Holzinger, Langs, Denk, Zatloukal, & Müller, 2019) with little controls on the internal mechanisms present, so building a balanced dataset is difficult when the GAN produces low nPo data. With a mathematical simulation, we can adjust the Markovian transition rate matrix to increase the event frequency when the data is particularly imbalanced.

9.1.4 Digital vs Analogue recordings

Another decision to make in the data sourcing process is whether or not to use analogue equipment in the data simulation process. In previous work (Celik et al., 2020), data was simulated via digital means, but noise added by passing this data through the real analogue patch-clamp equipment used in a lab environment, adding genuine noise with similar characteristics.

This comes with a major drawback in that by passing the data through an analogue machine, it bottlenecks the production of data to a real-time process; data is re-recorded out of the analogue amplifier in real time. Furthermore, whilst it includes authentic "amplifier noise", there are still native physiological noise types that are not included (open channel noise etc, see above). On the other hand, with the mathematical models, since the noise added is digital, it can be simulated orders of magnitude faster than the analogue method. As we found the performance of the deep learning models had a high reliance on the amount of data we had, this proved important in model tuning; as doubling the amount of data available to the models gives a significant improvement in performance.

9.1.5 Synthetic Noise Analysis

One of the key criticisms of synthetic, digitally generated data (as with the case in the Markovian simulations) is the unrealistic nature of the signal compared to lab recorded data. In other work (Hotz et al., 2013), data is simulated by adding Gaussian noise to a square wave generated via a Markovian process. This creates starkly different looking data to lab-recorded data and may form an unrealistic standard of the quality of data these models are likely to receive.

For this work, the quality of the data is a larger concern as it's the primary source where the models learn their patterns from; if the data is not "close" to what is expected in the real, lab recorded case, model performance is likely to be poor. In fact, when tuning the

channel idealisation models to perform better on the real data, changing the data and retraining the model was the largest source of improvements.

As discussed previously, the noise seen in ion channel recordings is 1/f noise; but the clearest way to see the difference between the noise terms in this work and previous work is to look to the power spectrum density plots for our simulated data, some real data from the lab, and Gaussian noise (Figure 9.1).

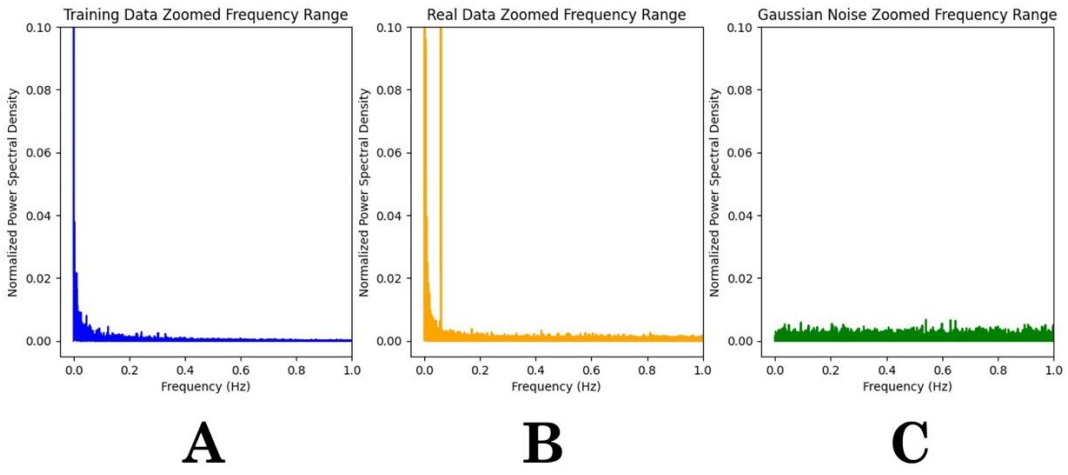


Figure 9.1: Noise analysis of simulated training data (A) versus real data (B) versus Gaussian noise (C).
In this work, simulated data was used with customized noise layers to better reflect the nature of the noise seen in real lab recordings. In other work, simple Gaussian noise was used layered on top of a Markovian simulation – but it is clear from the power spectrum analysis here that it does not accurately reflect the nature of the noise as in the methods in this work. Note that the density of the noise trails off in the training data and real data, but is constant throughout in the Gaussian noise.

We clearly see similarity between the simulated data and real data, and a large difference between these two and the Gaussian noise; this is no accident and is the result of a large amount of testing and tuning for the training dataset to be as high quality as possible.

9.1.6 Pre-processing

Pre-processing forms a crucial part of any artificial intelligence problem, with many algorithms depending on certain properties of the data to function correctly; for example

PCA assumes equal scaling across each of the input features. Deep learning is no exception; the inputs for each of the models in my work has needed significant processing performed on it before being passed into the model itself, and these were very different in Chapters 5, 6 and 7.

In Chapter 5 I found that relatively simple pre-processing allowed outstanding model performances; with Cohen's Kappa scores for recovery of Markovian state of 0.9665. However, in Chapter 6, I found that implementing the Viterbi algorithm significantly improved model training times yet further. I also found that (as with the later channel idealisation work) by constricting the expected classification range to the middle 50% of points ("progressive windowing, Chapter 7) we also saw a significant improvement in performance, beyond even the Viterbi implementation of Chapter 6. Both of these pre-processing adjunct steps are focused on giving the model the "easiest" job possible; it's simply not possible to expect a neural network to correctly predict the output of a truly stochastic process, and is not helpful to expect the same performance for a model on a point with context in both time directions versus a point with context in only one direction on the edge of the time window.

For the GAN work (Chapter 3), pre-processing went a step further into data augmentation (Wong, Gatt, Stamatescu, & McDonnell, 2016). Due to the extremely low amount of data we had available, we needed to ensure that the data we used was high quality, and utilised as much as possible. During training, it was clear that any input with no events in must be removed, leaving us with even less data; but by flipping each input on the time axis (reading it back to front), and by adding a small amount of Gaussian noise to each input, we generated enough variation within the training data for the models to converge. In contrast, our predictive models (Chapters 5, 6,7 and 8) did not use any data augmentation as such, since there was, in effect, an infinite volume of training data available (see above).

9.2 Potential of Deep Learning Models

Deep learning models have seen growing use for complex tasks across a range of fields due to their ability to learn complex patterns from data without many assumptions about the nature or structure of the data itself. This flexibility has allowed for models to gain better-than-expert metrics at tasks such as cancer detection (Ardila et al., 2019; Hu et al., 2019). Deep learning approaches have several advantages over other AI methods which we have successfully exploited in our single channel analyses in this work.

9.2.1 Complex Pattern Recognition

Some AI algorithms (for example, PCA, support vector machine (SVM)) enforce some kind of structure in the function being used to build a model; for example, in linear PCA; the projection to the PCA space is a linear combination of the features in the original space. In linear SVM, the support vectors are always linear, even if it is not necessarily the best choice of fit for the data. This can be changed through transformations in pre-processing, or adjusting the functions used for the support vectors, however this comes at a performance cost as new parameters are added.

In deep learning, due to the number of large number of parameters (often several million) and depth of the models, far more complex patterns can be fitted to the data. In fact; the universal approximation theorem shows that with enough neurons in a single hidden layer, a network can approximate any function to an arbitrary accuracy within a bounded region (Hornik, Stinchcombe, & White, 1989).

This is perhaps why deep learning tasks show successful metrics for multimodal or complex data tasks (Ramachandram & Taylor, 2017); since the dimensionality of the input data is high, and covariance is likely, naturally the best fit is likely to be a function with a large number of parameters. For the case of deep learning application to ion channel data idealisation, we know from dwell time analysis that there are underlying Markovian processes occurring and that the best fit for idealisation of this data would ideally take

this into account; for example a poor model might reproduce aggregate statistics (for example dwell time histograms) missing key characteristics (e.g. the non-unimodal nature of the dwell time histograms reflecting the Markovian nature of the data), or try and enforce the Markovian structure from the training dataset onto any new data. The approaches I used in this thesis (Chapters 6 and 7) maintain this characteristic behaviour shown in the aggregate statistics in Chapter 8 (see model dwell time histograms in Figure 8.32), showing that the model is either "understanding" some underlying structure to the data or is agnostic to changes in the underlying Markovian structure and continues to idealise correctly despite a significant change in the nature of the data. Either way, the model produces useful outputs for testing ion channel behaviour; showing the ability of the models to work with the complex underlying mechanisms.

9.2.2 Assumptions in Model Development

Several AI models make significant assumptions about the nature of the data in training; for example, PCA is sensitive to the scaling of the features, outliers, and imposes a linear relationship (by default without any processing) for it to function correctly (Wold, Esbensen, & Geladi, 1987). Deep learning models on the other hand take a data-driven approach, making few assumptions and allowing the model to infer itself via gradient descent any patterns in the data . Due to the complexity of ion channel data, and the flexibility of deep learning models, it is possible that models I deployed successfully in this thesis are in fact using other information (for example, noise shape) than simply the change in current for idealisation. An experience patch clamper (feels) they can determine subtle open and closed events in raw data by taking all these "features" into account. It is possible that if it were possible to have a ground truth with real data the models would be even more successful than we found here. We do not have complete knowledge of ion channel function, and the existence of better-than-expert models in other fields (Hollon et al., 2020; McKinney et al., 2020) implies that deep learning models are capable of ascertaining patterns outside of our understanding for better analysis of the data.

9.2.3 Fully Automatic Idealisation

Previous work (Hotz et al., 2013; Qin, 2004) has all needed some form of human supervision to achieve an idealisation (for example, an initial guess of Markovian model), or faces significant caveats or assumptions on the data inputted (e.g. absence of baseline drift). One of the main allures of deep learning models is the "black-box" ability to take in a wide variety of complex data and output labels without any human input.

For ion channel idealisation this is particularly important; as fully automated ion channel recording becomes more and more widespread; the analysis becomes a more limiting bottleneck for results, even with semi-automatic analysis methods. An ideal model should be as general as possible, allowing for fully-automatic idealisation without user input, allowing for end-to-end analysis from cell culture to summary statistics.

In this work, I have shown that this is possible, albeit with the caveat that some parts of the model pre-processing work better with some user tuning, although this is not strictly necessary (for example, automatic versus manual TANE in Chapter 8.3.5.3 and Figure 8.43). During analysis, the full analysis was run automatically on the raw data, outputting both the raw point-by-point idealisation (for manual analysis where needed), and the summary statistics (nPos, dwell time histograms etc).

9.3 Drawbacks of Deep Learning models

While we found our deep learning models to show high performance on the complex problem of Markov state recovery, and are generally more flexible than previous methods; in practice, in other domains, alternative machine learning solutions often show similarly high performance under certain circumstances due to a number of reasons , for example over-abstraction or over-fitting on the data (Nitze, Schulthess, & Asche, 2012). Furthermore, deep learning models are often far larger and slower than their non-deep learning counterparts (Vakili, Ghamsari, & Rezaei, 2020); causing them to be impractical in applications where speed or size are more important than accuracy.

9.3.1 Model Size

One of the main disadvantages I foresee with my deep learning models compared to other AI analysis models is time performance; i.e. how fast the model runs. The clear example of this is the Segmented K-Means algorithm, used by the popular software QuB (Qin, 2004). This model is far smaller and faster than my models, and runs on any (Windows) machine (not just ones with CUDA enabled graphics cards). However, the end-to-end nature of my work means in practice very often then deep learning models take less time to analyse the data as less human input is needed to give a sensible output.

I found throughout the present work (Chapter 5 ,6 and 7) that my very large models, adapted from those used for complex image recognition tasks (such as ResNet) performed similarly to the lighter deep channel models (e.g., my SimpleCNN) for channel idealisation and Markovian state prediction; this may be unsurprising as CNN models with fewer layers that focus on edge detection are likely to detect most events. For this reason, I dropped the ResNet based model after Chapter 5. It should be noted, however, that even my relatively simple models are still significantly larger than the SKM QuB model although smaller than the ResNet model (17 and 12 million parameters versus 58 million parameters for ResNet); so are faster. This would be a consideration for, for example, web deployment of ion channel detection models (see below) although, computer power continues to increase with such a pace the limitation may evaporate soon.

9.3.2 Model Specificity

As computational power increases, the size and speed of our model becomes less of a concern; however occasional model failures could be critical when considering a fully-automated pipeline. As such, further work should be done to make the model more robust to a larger range of conditions (such as longer or shorter dwell times; a larger range of noise; or artefacts in the data). We have shown that models trained on synthetic data can be used on data with a different Markovian network by adjusting some pre-processing parameters. We also found that our new models work on data with more baseline drift

than seen in previous work (Celik et al., 2020; Hotz et al., 2013; Qin, 2004), so our models are fairly robust, but if the Markovian network is changed too much, or the signal to noise ratio is too low, model performance decreases significantly (see comparison of "Static" to "Perturbed" metrics in Chapters 5, 6, 7, or the model's subjective change in performance in the Vernakalant dataset compared to the higher noise Penitrem Dataset).

9.3.3 Reliance on Large Datasets

One way of overcoming model specificity issues is to use a wider range of datasets, with data coming, ideally from different sources and different methods. Deep learning models typically require a larger dataset to train from versus simpler models, and it was clear from earlier work that maximising the scale of this data was a significant factor in model performance; initial preliminary training for the Markovian state recovery experiments had significantly less data to train and model performance was ~20% lower across the board for every model (data not shown). This showed that for training deep learning models, human labelled data was not viable, and even the previous DeepChannel method of using simulated data through an analogue amplifier might not give the throughput needed for sufficiently training these large models (which simulates data in real-time, needing 160 hours to reproduce the data used in Chapters 5,6,7 for example); hence the continued development of the data simulated pipeline to be able to generate a large amount of similar data on command.

The extent of the success of the models trained on synthetic data predicting lab-recorded data was perhaps surprising and highly promising; lab-recorded data is complex, with many factors affecting both the cell's function as well as the accompanying noise. Initially, it was planned that the simulated data might be too dissimilar to the lab-recorded for the models to be able to analyse, so the models would be "transfer learned" on a small set of hand-labelled lab-recorded data to fine tune the models to a point where they would work on the real data. Transfer learning is a process where all, or some of a model, is retrained on a different dataset to tune the model to a desired domain. This is common in applications where the applied domain has difficulties in acquiring high quality data (as

in our case), or when applying another party's model (for example a generic image recognition model) to a more specific use case (i.e. animal detection). The theory behind transfer learning is that the early neurons are responsible for the fundamental pattern recognition processes that are shared across several domains, and only the last layers are specific to the use case. By "freezing" the first few layers of a model, these "fundamental" patterns (such as line, shape detection) are kept the same, whereas the more specific processes (such as ear, nose, mouth detection) are allowed to retrain on the new, specific dataset. As it transpired, in preliminary experiments, we found transfer learning failed (*data not shown*), initially models regressed in their detection ability before learning the new structure apparently no better than a naive model. This implies that the quality of the synthetic data was relatively high, and my approach correct. Still, there is significant room for improvement in this area; as deep learning is very much a data driven AI method, improving the data for training is a primary method of improving model performance.

9.3.4 The Opaqueness of Models

Another very important aspect to training the deep learning models throughout this work was fine-tuning the pre-processing for each application. Deep learning models are notoriously hard to debug (as they are essentially "black box" algorithms with limited ways to see the inner workings of); and ensuring the pre-processing pipeline takes some time and development for each use case. Often this will cause development cycles to slow down as some training is needed before changes can be shown to have a positive effect on the model's performance.

In the GAN work, initially all the standard pre-processing steps were applied to the signal; normalisation, data augmentation via flipping, scaling and moving; however in early training experiments all generator outputs were simply empty records with no events. This was due the fact that there were input windows in the training set without events in, and the generator model had clearly decided that these were the easiest sections of the data to replicate to fool the discriminator. Removal of empty windows rectified this

problem; but is an illustrative example of pre-processing decisions being iteratively improved over many training runs over a longer period of time.

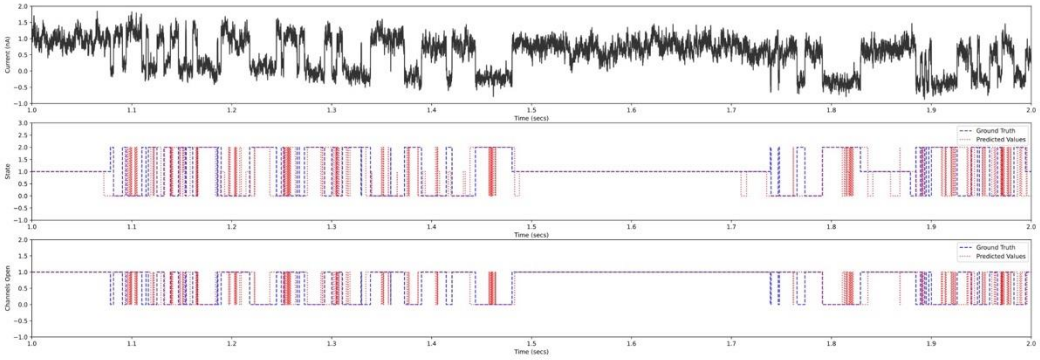


Figure 9.2: Example of "Flickering" classification on simulated testing set.
Here we see that the model sometimes flickers between two states – this can either be due to under-training, or the model being undecided between two states and splitting its decision between two states in order to reduce the loss function on balance.A better loss function may solve this problem, or improvments to the dataset.

A training problem that we were not able to overcome is finding a suitable loss function that accurately defines what a "good" idealisation consists of. Currently the loss function used in the Markovian state recovery and channel idealisation is categorical cross-entropy; this is an unweighted point-wise comparison of the prediction to the ground truth via log-likelihood. While this is helpful, and generally correlates to model performance; there are a few cases where a model can abuse this loss function to get a lower loss without giving a "helpful" output. The first of these is to simply predict no channels are present when channels are particularly rare. In unbalanced (but perhaps realistic) datasets such as these, a model's categorical cross-entropy will be low, and accuracy high if the model simply predicts nothing is happening; however it is extremely important that the model is sensitive to rare events as this is often biologically important in recordings (for example in a channel inhibitor). Another way a model might try and "game" the loss function is to "hedge" across two conductance levels or Markovian states in cases where it is not

particularly sure by flickering between the two classes (Figure 9.2). Again, while this results in increased accuracy, this is extremely damaging to analysis as each flicker counts as an event and creates many small dwell times in dwell time histogram analysis. Both of these problems can be approached by a use of a different metric; for example both the micro F1-Score and Cohen's Kappa score mitigate these problems; however they cannot be used as loss functions as they are not differentiable. They also still have limitations themselves (see below). The compromise we use here is to try and use balanced datasets that exhibit constant, reasonable activity relative to the window size.

All of these problems require solutions either through pre-processing or post-processing, since there is very little that can be done within training itself (or on the model object in post) to solve problems. The model is very much a black box and it is hard to understand the internals, therefore instead it must be done symptomatically by looking at particularly problematic sections of data and outputs.

9.4 Future Work

This work shows promising results for ion channel analysis using deep learning models, however there is still room for improvement when it comes to model performance; these improvements can be broadly sorted into two groups; iterative improvements that take the general method of the work and iterate it to get better performance, and major changes to the methodology that may offer a jump in improvement.

9.4.1 Iterative Improvements

In addition to the apparently unresolvable issue of fitting loss functions being non-ideal for our use case, the choice of metrics makes an enormous difference to apparent success. As is widely discussed in ML circles, and seen in Chapters 5, 6 and 7. Machine learning can "game" the popular F1 metric for example. With a low activity channel and F1 would be near perfect despite modal failure of the model. For this reason we focussed on Cohen's Kappa which factors in the possibility of chance observations (McHugh, 2012). Note that in Chapter 5 we found instances where the model had entirely failed, reported Coehn's

Kappa as zero and yet F1 remained satisfyingly high. However, even the Kappa statistic may not capture what electrophysiologists would judge as good and bad idealisation. For example, a strip of data with one missed event would disturb a patch clamper much more than a strip of data with several open and closed events a few of which have been missed. Electrophysiologists would most likely look for events ("sojourns"), rather than numbers of correct or incorrect points. In theory it should be possible to calculate metrics that represent events rather than points, but as we thought about this during the project, the problem became surprisingly complex. What proportion of the event would have to be detected to be called correct, how closely aligned to the original would it need to be etc. Nevertheless, in the future, a custom metric that could be formally validated would allow better selection of optimal models.

9.4.1.1 Data Synthesis

Significant work was done in this paper to generate synthetic data for training deep learning models; through theoretically based probabilistic Markovian simulations with crafted noise added in post, or a more data driven approach of GAN data generation from a small seed set of labelled data. As deep learning is so reliant on the data used to train; this is where I believe the major improvements could be made.

Perhaps the most important omission from the current work is the lack of incorporating real artifacts or "extreme" noise elements in the simulated dataset. In practice, there are several scenarios that can happen in an ion channel recording; from sharp spikes in the ion channel apparatus being disturbed to 50 cycle noise being visible in the output signal. Currently, the simulated data does not reflect this reality, assuming perfect recording conditions for analysis. Ideally the model should be able to (internally or externally via labelling) identify problematic sections of a recording trace and deal with them appropriately.

9.4.1.2 Improved Pre-processing Pipelines

Currently the pre-processing pipeline is fairly involved and has many parameters with sensible, but untuned defaults (such as the peak detection parameters for anomaly detection). Tuning these parameters is a long and difficult process, and it's not clear if the best settings for one dataset would be the same for another; so some work could be done in trying to dynamically choose these parameters depending on characteristics of the data in a similar way to how the dynamic data scaling currently works.

Similarly, the scaling process showed to be a key factor in getting helpful outputs from the deep learning model. However the current method is rather rudimentary and is likely to have room for improvement. In detecting the number of channels, the peak fitting algorithm also outputs the mean channel size normalising the data to make this size constant was attempted, but the peak fitting algorithm on the amplitude histograms was not robust enough for this to be a viable option. Again; improvement in the peak fitting algorithm (or perhaps a different approach entirely) would likely result in a better pre-processing pipeline, and therefore better model performance.

9.4.2 Alternative Deep Learning Approaches

9.4.2.1 Transformers

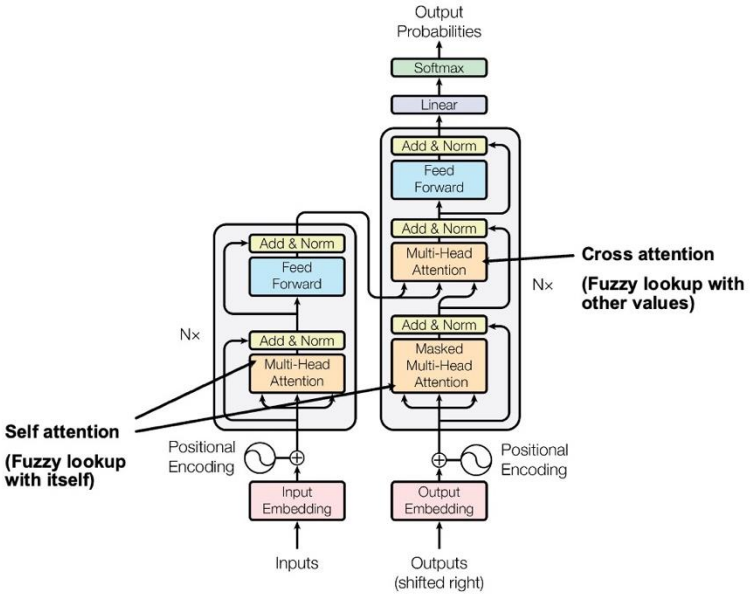


Figure 9.3: Transformer architecture adapted from (Vaswani et al., 2017)
Transformers are a relatively new architecture using attention layers in order to build long and short term dependencies in the input data. Transformer networks have seen significant usage in natural language processing tasks, as well as image classification tasks using Visual Transformers.

The bulk of the work in this thesis is based on convolutional networks, however perhaps entirely different architecture could be useful. One such example is the new transformer development. Transformers are a relatively new type of deep learning model that have recently made a particular impact in the world of natural language processing with models such as the GPT family (GPT3, GPT4, ChatGPT etc), Google's Bard, and Meta's LLAMA models. These models are extremely large, and use a vast dataset (usually scraped from the entire internet), but share a new, attention based mechanism developed by Google in late 2017 (Figure 9.3).

Transformers were first used in the context of natural language processing for translation processes; the central concept of how transformers work is the concept of attention, which can be compared to a "fuzzy dictionary"; instead of returning a single value when queried as in a normal dictionary, it instead outputs a vector of probabilities corresponding to the

relation of the key to the values; in other words how much "attention" should be paid to each value.

Transformer models have several advantages to convolutional counterparts; they are computationally fast due to processing entire sequences at a time, consider the entire sequence at once through the use of matrix multiplication in the attention layers, and are more interpretable due to the attention matrices having clearer meanings than the convolutional layers.

In a parallel to CNNs being used on natural language processing (NLP) tasks, Visual Transformers (Dosovitskiy et al., 2010) are now being used on image recognition tasks by segmenting images into portions, embedding them in a space and continuing as in the language case. Visual transformers are seeing state-of-the-art performance on image datasets, perhaps due to their ability to consider global trends in the data (relating image segments from opposite sides of the image, for example).

Transformers are a relatively new technology in the space of deep learning, and so it is unsurprising that they did not come up during a literature review on deep learning in electrophysiological signals. However - we have attempted to use transformers in our work; in both the sequence to sequence (predicting the next point's idealisation value based on the historical idealisation and current signal) and in a visual transformer method similar to the CNN methods we have used earlier. We experienced that the models would fail during testing, predicting the same class throughout the entire file (Figure 9.4). There are several reasons this might happen; transformers are known to require relatively more data than a CNN to perform well, but perhaps more importantly is that the behaviour we are interested in for a successful idealisation is mostly local; events happen over the course of a few samples, with relatively little relation to points hundreds of samples further on, in comparison to long sequences of text for example. Therefore, the model architecture might be too general for the application and considers too much information in the process of idealisation.

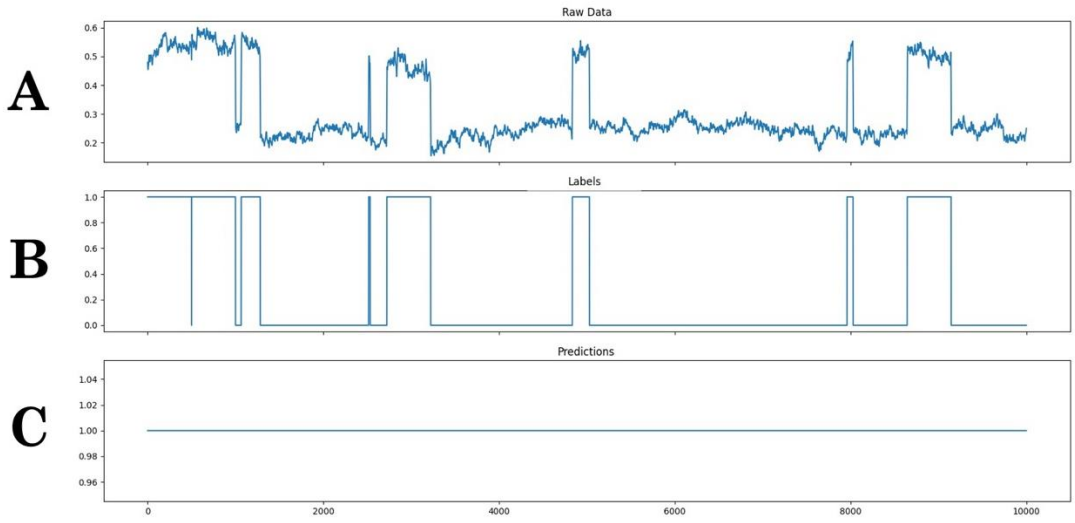


Figure 9.4: Example 1s (10,000 data points) data trace (A) with ground-truth labels (B) and transformer predictions (C) for a transformer model trained on the "Static Three State Model without Drift" dataset for channel idealisation.
Transformer models are currently the best in class model for natural language processing tasks, and are also showing very promising performance in image segmentation tasks via Visual transformers. However when applied to our data, we see total modal collapse in testing, with the model predicting a total number of channels open as 1 for the entire file.

9.4.2.2 Accessibility of Models (Webapp)

Currently, the requirements for using the deep learning models covered in this work are fairly complex; users are required to have a CUDA enabled GPU for training, with a specific combination of GPU drivers, CUDA versioning and Tensorflow versioning for the code to run correctly; although for idealisation this isn't strictly required. Achieving a working set up for this is infamously difficult and technical; to the extent where Tensorflow providers have produced a virtual machine image using Docker to bundle all these systems together.

All these problems form barriers for researchers to use our tools; and so some work was done into making these models accessible to as many people as possible. The result of this was the DeepChannel webapp (Figure 9.5); a platform for users to be able to use any of the pretrained models in this work themselves, without having to install Python or other technical tools.

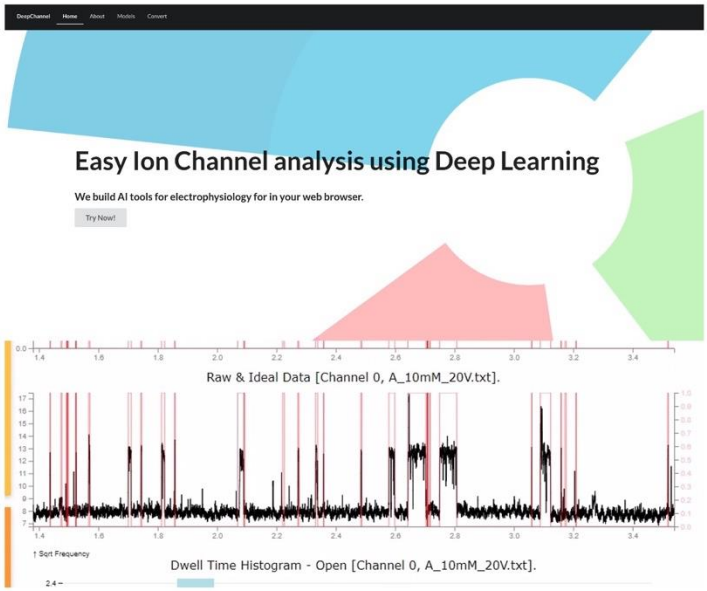


Figure 9.5: Web application interface for using Deep learning models.
The models trained in this work can be uploaded to the DeepChannel platform to allow for in-browser idealization. The major advantages of this method are twofold – it allows researchers to get idealisations in browser without installing any deep learning software or Python; and also allows data to be idealized without the sensitive data leaving the researchers' system due to the analysis being performed client side.

This approach comes with a few advantages; firstly, all the models are stored on a host platform and fetched by the user as and when needed; this allows for new versions of the model to by uploaded without affecting existing users' workflows. Secondly, the data being analysed never leaves the user's machine; the model is downloaded, and all analysis occurs within the browser using TensorflowJS. This is particularly ideal for security; potentially sensitive data never leaves the users' machine. Finally; building as a webapp allows for extremely fast prototyping at little cost to model performance; TensorflowJS uses either the WebGL or newer WebGPU backends to utilise GPU resources for speed where available; without the installation of the required drivers. My pilot app experimentally is now being developed in partnership with AstraZeneca by the Computational Biology Facility at the University of Liverpool.

Several Nobel Prizes have been awarded for ion channel work in the past 50 years, but there are fewer labs that perform this research to day, as biology becomes swamped with the powerful, but technically easier 'omic research. The combination of my best models developed within this thesis, which can deliver fast and accurate results (for example a

Kappa score of 0.9906 for single channel idealisation) together with wide deployment via a successful web app mean that these methods could become the future of single channel analysis in Universities and in industry. This would potentially re-invigorate single channel biology, one of the most powerful single molecule biological techniques ever invented for understanding pharmacological effects and discovering new, novel therapies, but currently under-exploited due to the skill needed to both acquire and analyse the data.

10Appendix

10.1 Chapter 6 Supplementary Figures: F1-Scores

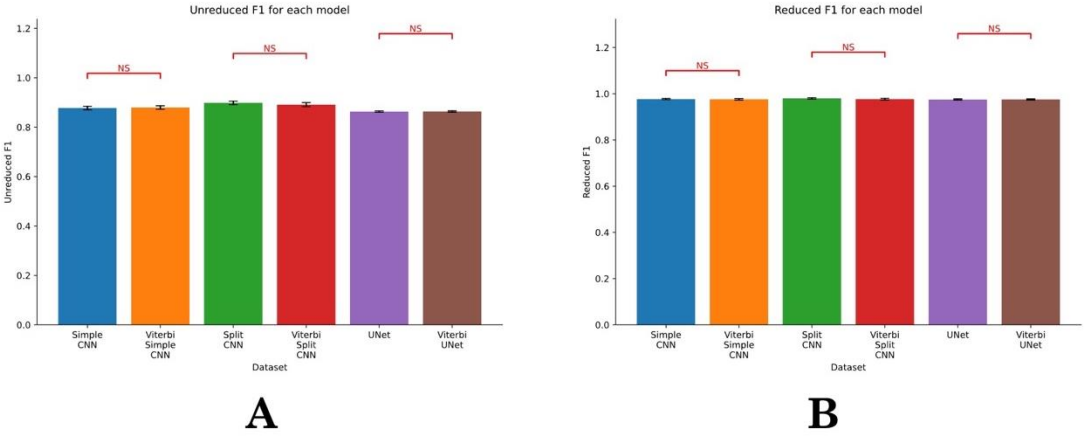


Figure 10.1: Model Training F1 Scores for "Static Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

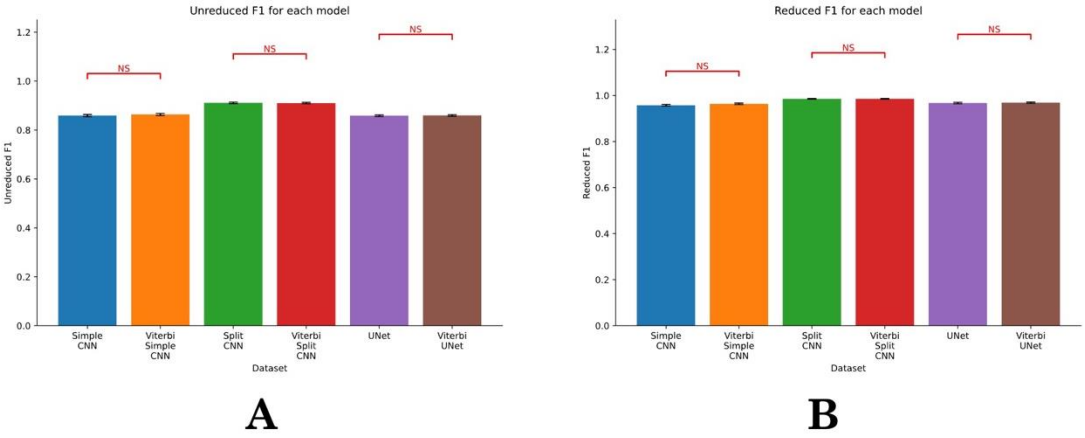


Figure 10.2: Model Training F1 Scores for "Static Three State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

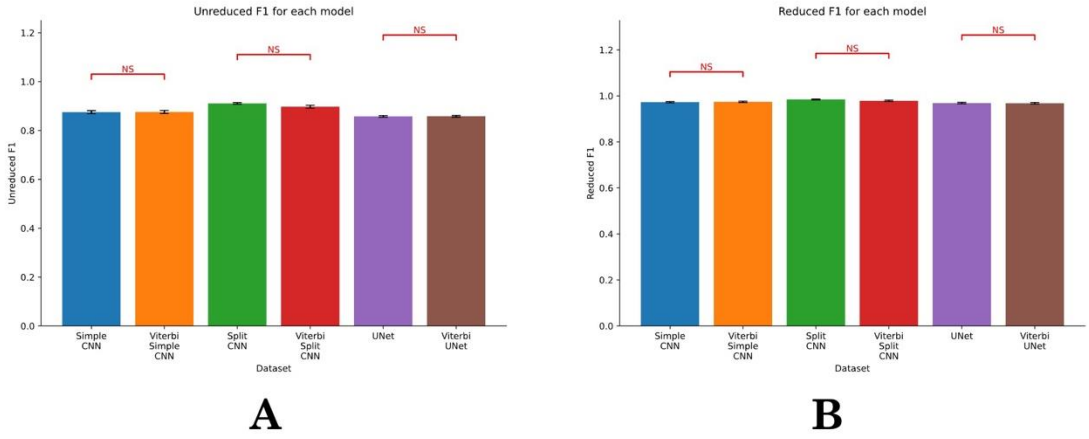


Figure 10.3: Model Training F1 Scores for "Perturbed Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

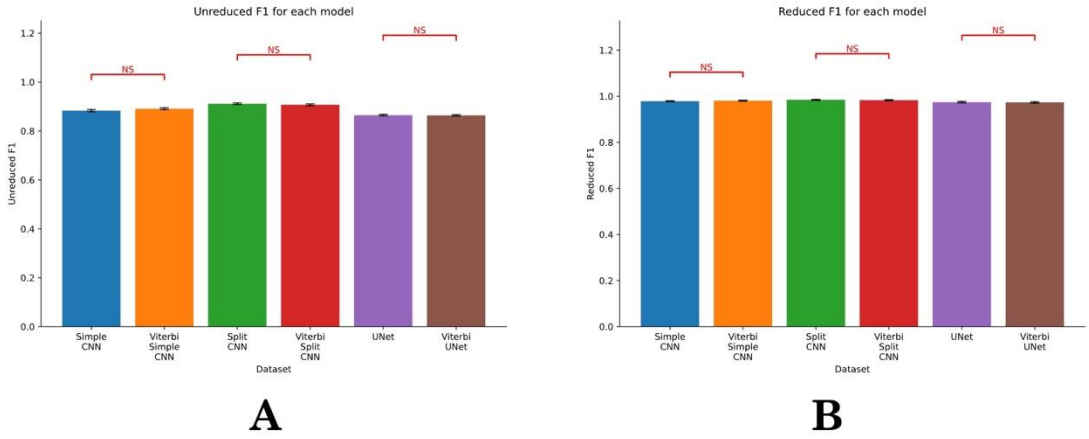


Figure 10.4: Model Training F1 Scores for "Perturbed Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

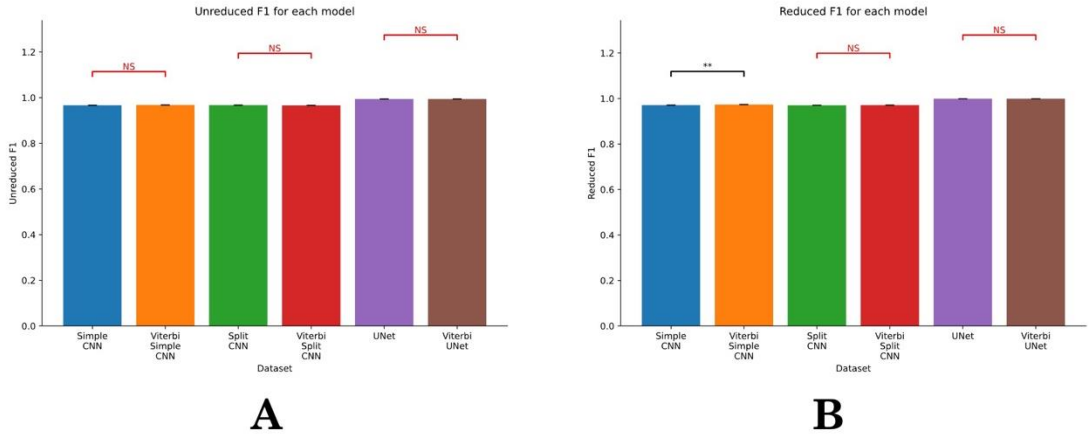


Figure 10.5: Model Training F1 Scores for "Static Five State Model without Drift" Dataset
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

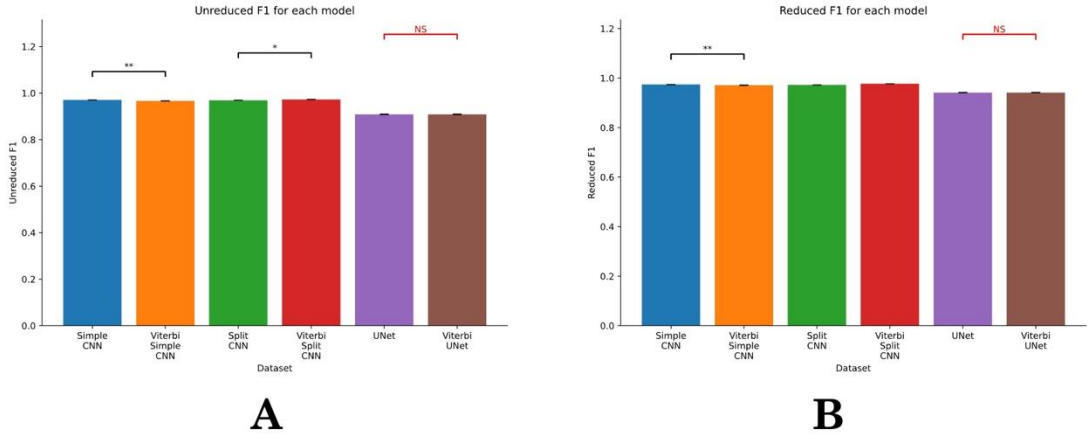


Figure 10.6:Model Training F1 Scores for "Static Five State Model with Drift" Dataset
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

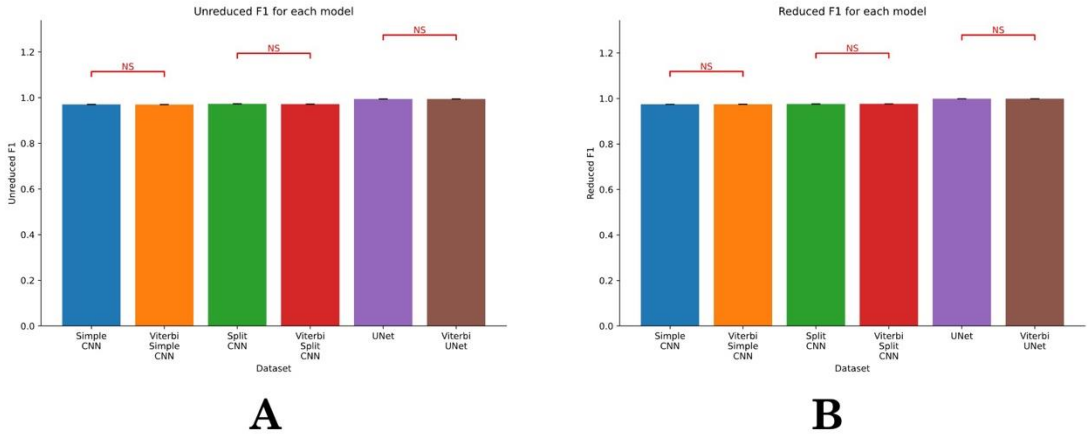


Figure 10.7: Model Training F1 Scores for "Perturbed Five State Model without Drift" Dataset
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

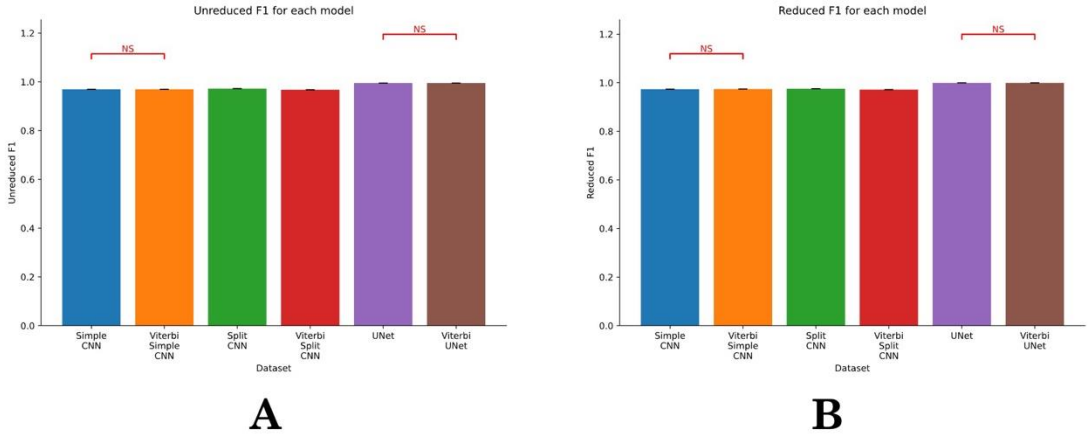


Figure 10.8: Model Training F1 Scores for "Perturbed Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all Viterbi/non-Viterbi pair of models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

10.2 Chapter 7 Supplementary Figures: F1-Scores

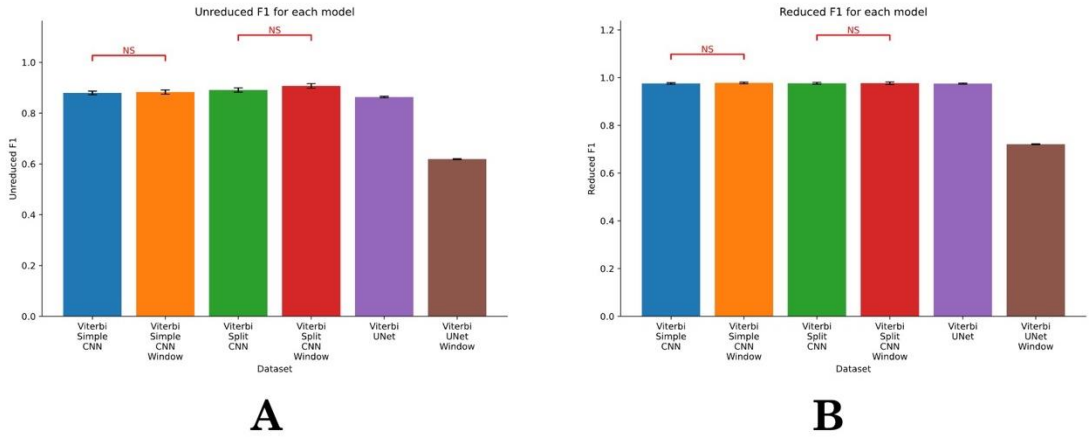


Figure 10.9: Model Training F1 Scores for "Static Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

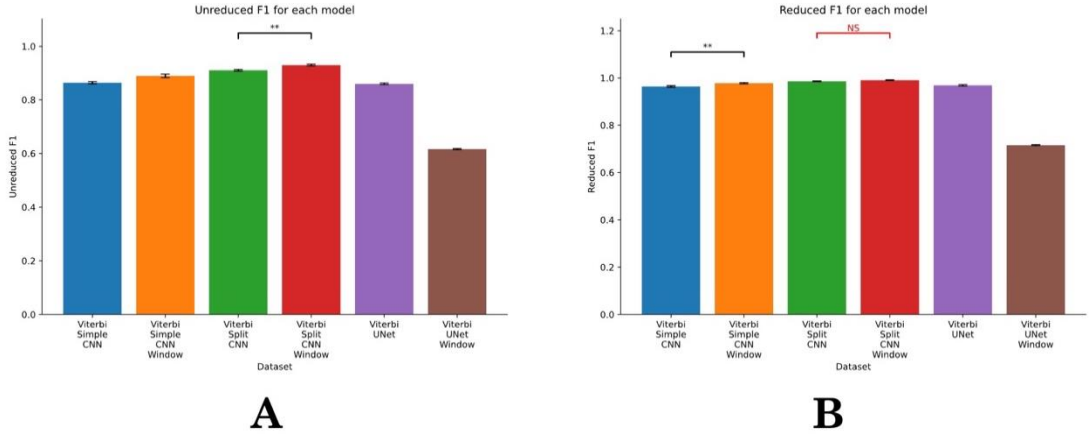


Figure 10.10: Model Training F1 Scores for "Static Three State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

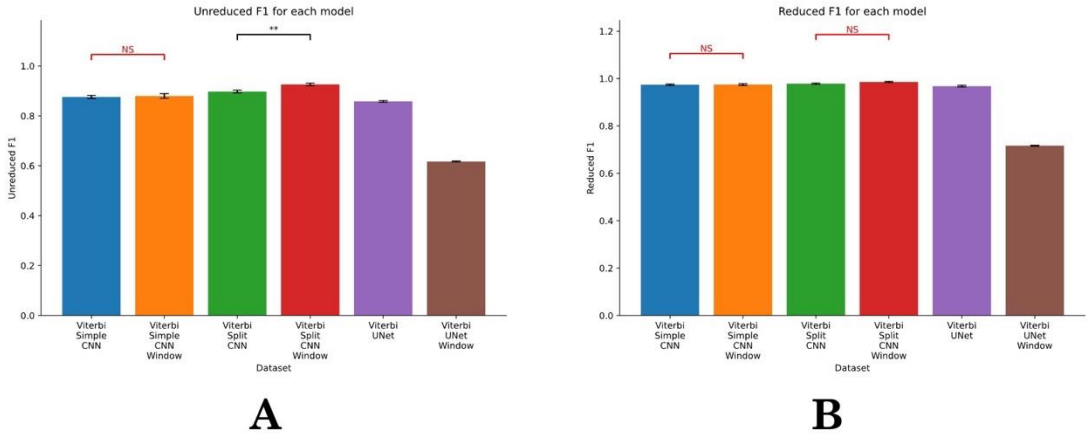


Figure 10.11: Model Training F1 Scores for "Perturbed Three State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

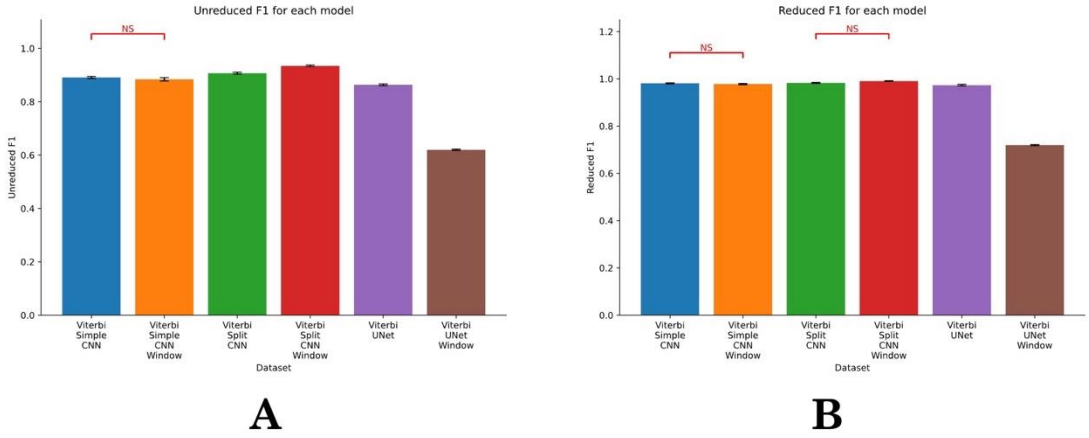


Figure 10.12: Model Training F1 Scores for "Perturbed Three State Model with Drift" Dataset
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

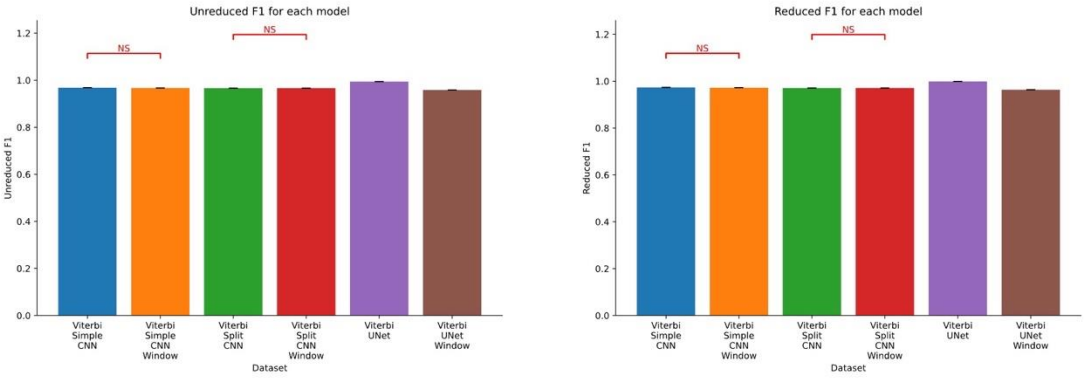


Figure 10.13: Model Training F1 Scores for "Static Five State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

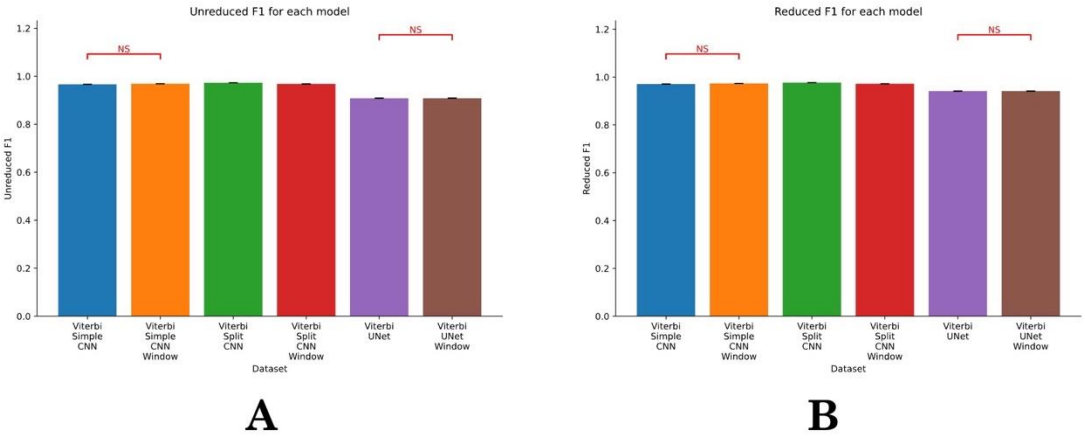


Figure 10.14: Model Training F1 Scores for "Static Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

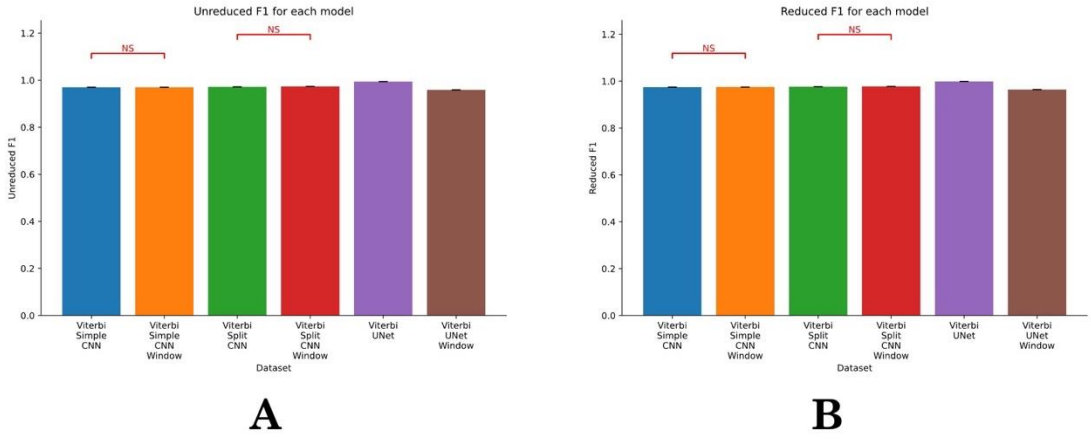


Figure 10.15: Model Training F1 Scores for "Perturbed Five State Model without Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

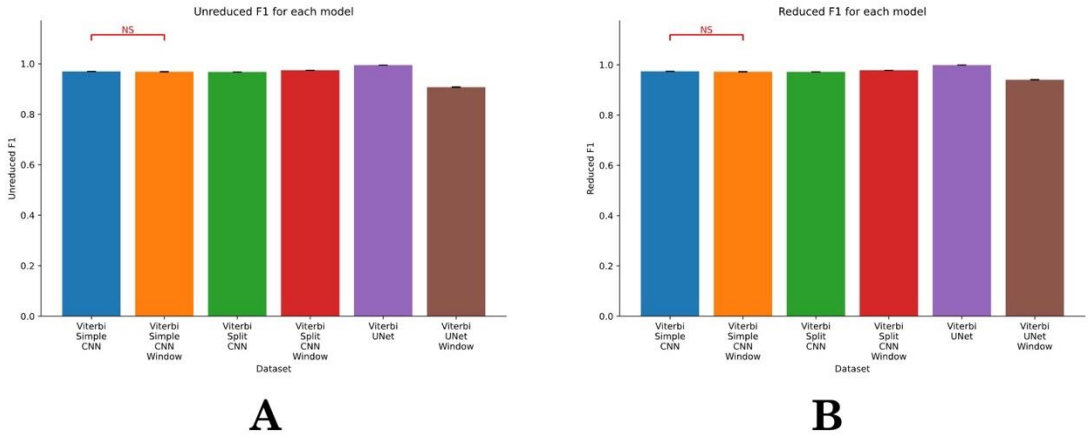


Figure 10.16: Model Training F1 Scores for Perturbed Five State Model with Drift" Dataset.
Two way ANOVA showed significant differences of the models for F1 score ($p < 0.001$) and between the datasets ($p < 0.001$); post-hoc tests were performed for each model pair within the datasets – all models were significantly different ($p < 0.001$) unless stated otherwise. Red pairs labelled with an "NS" denote no significance ($p > 0.05$), black pairs with a single star "" denote significance between 5% and 1%. Black pairs with two stars denote significant between 1 and 0.1%.*

10.3 Chapter 8 Dwell Time Area Parameter Plots

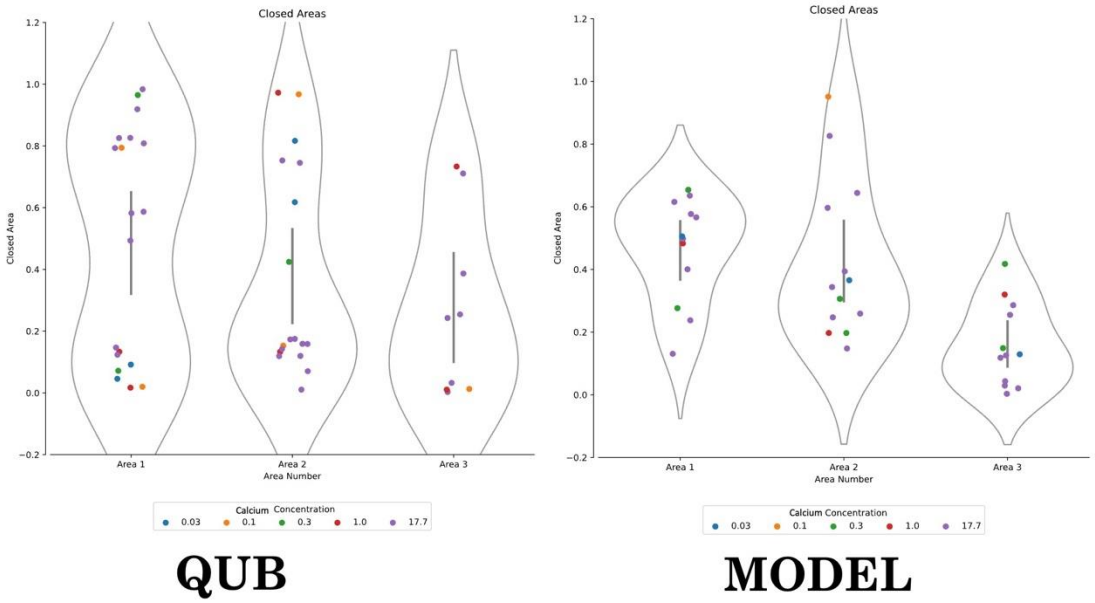


Figure 10.17: Dwell Time Parameter Plots for Closed Area Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these closed area parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the closed areas for either the QuB or model idealisations ($p > 0.05$, $n = 34$). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test ($p > 0.05$, $n = 34$). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters ($p = 0.0319$, $n = 34$); however this does not necessarily imply one is better than the other.

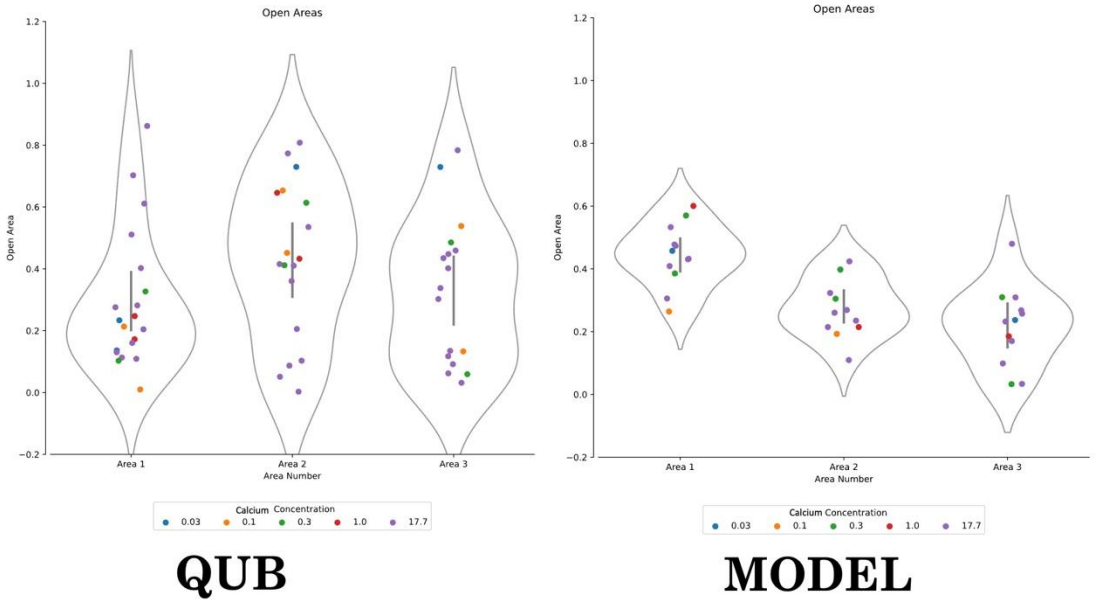


Figure 10.18: Dwell Time Parameter Plots for Open Area Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these open area parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the open areas for either the QuB or model idealisations ($p > 0.05$, $n = 34$). Furthermore, while the model exhibits a tighter clustering of parameters, it is not significantly so via an F-Test ($p > 0.05$, $n = 34$). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters ($p = 0.0319$, $n = 34$); however this does not necessarily imply one is better than the other.

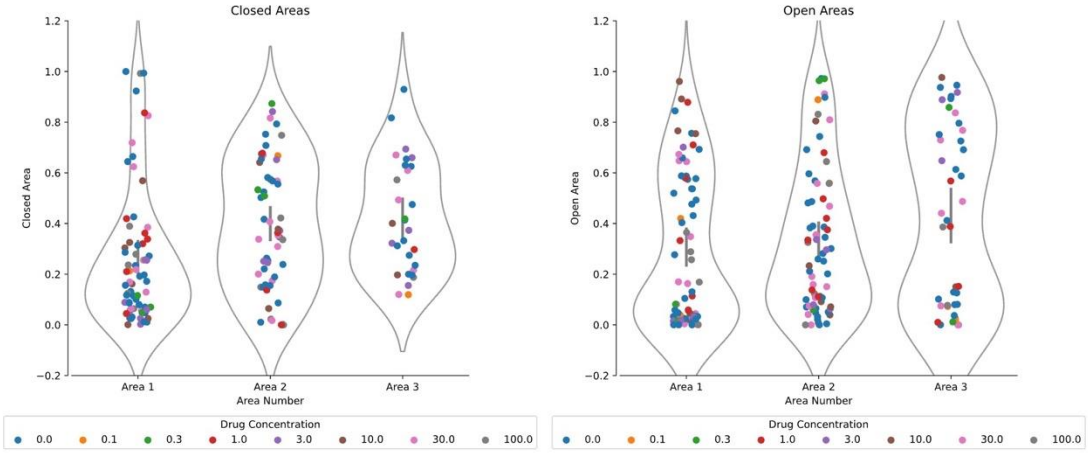


Figure 10.19: Dwell Time Parameter Plots for Closed and Open Area Parameters for the model (Auto TANE B5D) idealisations at -60mV.

For the idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these area parameters for the model idealisations. A two way ANOVA showed no significant effect of the drug on the closed or open areas ($p > 0.05$, $n=73$).

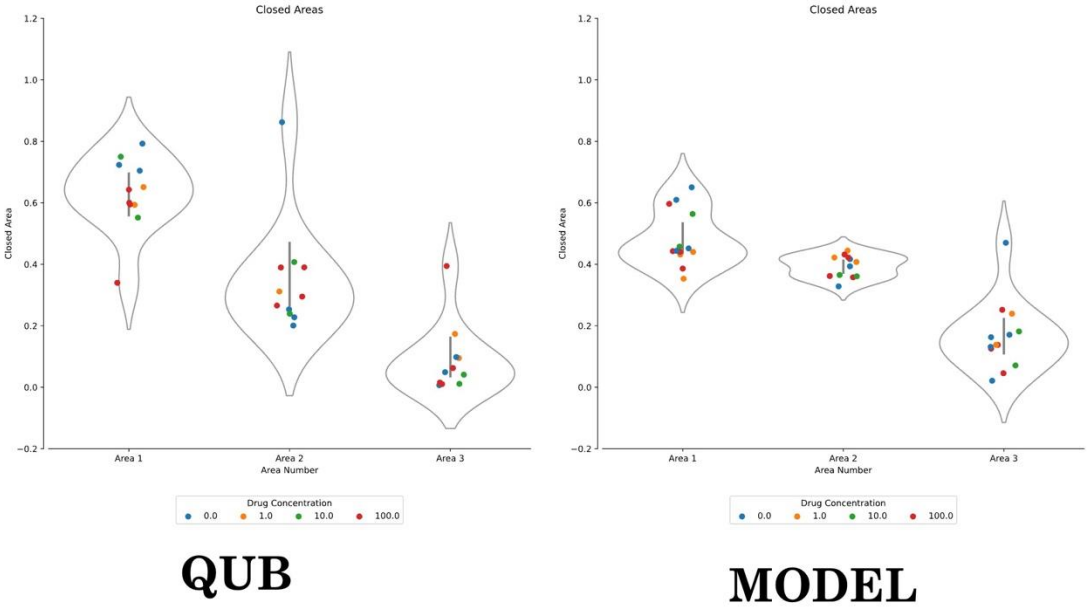
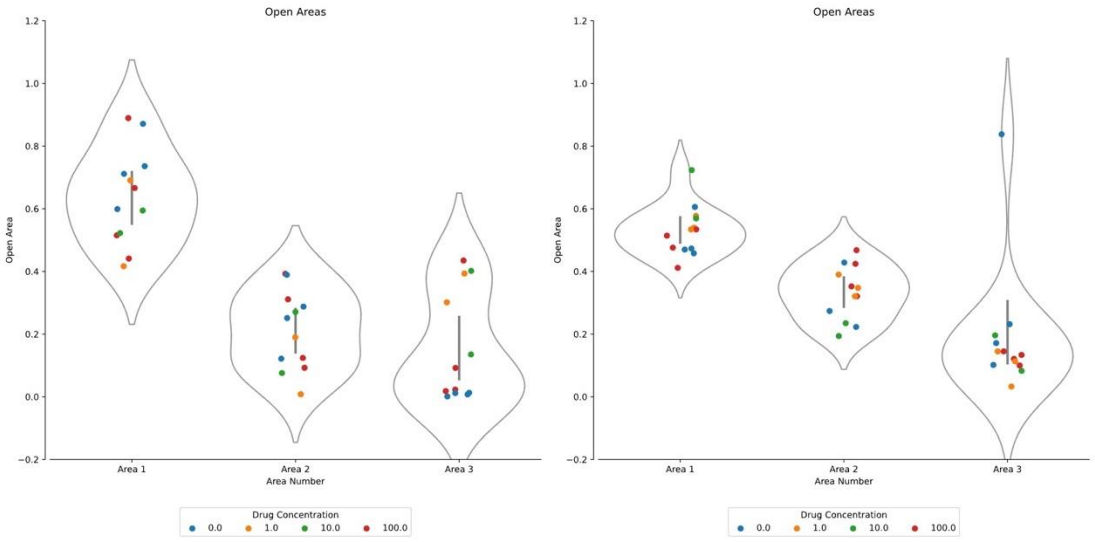


Figure 10.20: Dwell Time Parameter Plots for Closed Area Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these closed area parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the closed areas for either the model idealisations or QuB idealisations ($p > 0.05$, $n = 70$). Furthermore, while the model exhibits a tighter clustering of parameters (apart from here in the third area), it is not significantly so via an F-Test ($p > 0.05$, $n= 70$). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters ($p < 0.001$, $n = 70$); however this does not necessarily imply one is better than the other.



QUB **MODEL**

Figure 10.21: Dwell Time Parameter Plots for Open Area Parameters for both the manual (QuB) and model (Auto TANE B5D) idealisations at 40mV.

For both idealisations, the open and closed dwell times were recorded and the corresponding histograms fitted with three exponential curves (with area and tau parameters). Here we show the distribution of these open area parameters for both idealisations. A two way ANOVA showed no significant effect of the drug on the open areas for either the model idealisations or QuB idealisations ($p > 0.05$, $n = 70$). Furthermore, while the model exhibits a tighter clustering of parameters (apart from here in the third area), it is not significantly so via an F-Test ($p > 0.05$, $n= 70$). Two way ANOVA across all the area and tau parameters showed a significant difference overall between the QuB and Model parameters ($p < 0.001$, $n = 70$); however this does not necessarily imply one is better than the other.

Bibliography

Abadi, Ashish, Paul, Eugene Brevdo, Zhifeng Chen, Craig Citro, . . . Xiaoqiang Zheng. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. In.

Ackerman. (1998). The Long QT Syndrome: Ion Channel Diseases of the Heart. *Mayo Clinic Proceedings, 73*(3), 250-269. doi:10.4065/73.3.250

Aghajanyan. (2017). *Softtarget regularization: An effective technique to reduce over-fitting in neural networks.* Paper presented at the 2017 3rd IEEE International Conference on Cybernetics (CYBCONF).

Alagem, Dvir, & Reuveny. (2001). Mechanism of Ba2+ block of a mouse inwardly rectifying K+ channel: differential contribution by two discrete residues. *The Journal of Physiology, 534*(2), 381-393.

Ardila, Kiraly, Bharadwaj, Choi, Reicher, Peng, . . . Corrado. (2019). End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine, 25*(6), 954-961.

Armijo, Shushtarian, Valdizan, Cuadrado, & Adin. (2005). Ion channels and epilepsy. *Current pharmaceutical design, 11*(15), 1975-2003.

Armstrong. (1971). Interaction of Tetraethylammonium Ion Derivatives with the Potassium Channels of Giant Axons. *Journal of General Physiology, 58*(4), 413-437. doi:10.1085/jgp.58.4.413

Asano, Bratz, Berwick, Fancher, Tune, & Dick. (2012). Penitrem A as a tool for understanding the role of large conductance Ca2+/voltage-sensitive K+ channels in vascular function. *Journal of Pharmacology and Experimental Therapeutics, 342*(2), 453-460.

Ashmore. (2008). Cochlear outer hair cell motility. *Physiological Reviews, 88*(1), 173-210.

Bae, Kim, & Lim. (2021). Carbon monoxide activates large-conductance calcium-activated potassium channels of human cardiac fibroblasts through various mechanisms. *The Korean Journal of Physiology & Pharmacology: Official Journal of the Korean Physiological Society and the Korean Society of Pharmacology, 25*(3), 227-237.

Ball, Celik, Sayari, Abdul Kadir, O'Brien, & Barrett-Jolley. (2022). DeepGANnel: Synthesis of fully annotated single molecule patch-clamp data using generative adversarial networks. *Plos One, 17*(5), e0267452.

Barbi, & Petracchi. (1992). Coupled and uncoupled Markov systems: a possible way to distinguish between them. *20*(6). doi:10.1007/bf00196593

Barrett, Magleby, & Pallotta. (1982). Properties of single calcium-activated potassium channels in cultured rat muscle. *The Journal of Physiology, 331*, 211.

Barrett, & Tsien. (2008). The Timothy syndrome mutation differentially affects voltage- and calcium-dependent inactivation of CaV1. 2 L-type calcium channels. *Proceedings of the National Academy of Sciences, 105*(6), 2157-2162.

Baum, Petrie, Soules, & Weiss. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics, 41*(1), 164-171.

Bawaskar, & Bawaskar. (1992). Management of the cardiovascular manifestations of poisoning by the Indian red scorpion (Mesobuthus tamulus). *Br Heart J, 68*(5), 478-480. doi:10.1136/hrt.68.11.478

Bean, Cohen, & Tsien. (1983). Lidocaine block of cardiac sodium channels. *The Journal of general physiology, 81*(5), 613.

Belle, Diekman, Forger, & Piggins. (2009). Daily electrical silencing in the mammalian circadian clock. *Science, 326*(5950), 281-284.

Bellono, Kammel, Zimmerman, & Oancea. (2013). UV light phototransduction activates transient receptor potential A1 ion channels in human melanocytes. *Proceedings of the National Academy of Sciences, 110*(6), 2383-2388.

Bezrukov, & Winterhalter. (2000). Examining Noise Sources at the Single-Molecule Level:1/fNoise of an Open Maltoporin Channel. *Physical Review Letters, 85*(1), 202-205. doi:10.1103/physrevlett.85.202

Blatz, & Magleby. (1987). Calcium-activated potassium channels. *Trends in Neurosciences, 10*(11), 463-467.

Bootman, Collins, Peppiatt, Prothero, MacKenzie, De Smet, . . . Berridge. (2001). *Calcium signalling—an overview.* Paper presented at the Seminars in cell & developmental biology.

Botros, Siddiqi, & Deiri. (1993). *Automatic speech recognition using hidden Markov models and artificial neural networks.* Paper presented at the IEEE International Conference on Neural Networks.

Brenner, Peréz, Bonev, Eckman, Kosek, Wiler, . . . Aldrich. (2000). Vasoregulation by the ß1 subunit of the calcium-activated potassium channel. *Nature, 407*(6806), 870-876.

Browne, Gancher, Nutt, Brunt, Smith, Kramer, & Litt. (1994). Episodic ataxia/myokymia syndrome is associated with point mutations in the human potassium channel gene, KCNA1. *Nature Genetics, 8*(2), 136-140.

Brugada, Hong, Cordeiro, & Dumaine. (2005). Short QT syndrome. *CMAJ, 173*(11), 1349-1354.

Bruno, Yang, & Pearson. (2005). Using independent open-to-closed transitions to simplify aggregated Markov models of ion channel gating kinetics. *Proceedings of the National Academy of Sciences, 102*(18), 6326-6331. doi:doi:10.1073/pnas.0409110102

Bryson. (1961). *A gradient method for optimizing multi-stage allocation processes.* Paper presented at the Proc. Harvard Univ. Symposium on digital computers and their applications.

Buitinck, Louppe, Blondel, Pedregosa, Mueller, Grisel, . . . Grobler. (2013). API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.

Buono, Lohoff, Sander, Sperling, O'Connor, Dlugos, . . . Scattergood. (2004). Association between variation in the human KCNJ10 potassium ion channel gene and seizure susceptibility. *Epilepsy research, 58*(2-3), 175-183.

Burashnikov, Pourrier, Gibson, Lynch, & Antzelevitch. (2012). Rate-dependent effects of vernakalant in the isolated non-remodeled canine left atria are primarily due to block of the sodium channel: comparison with ranolazine and dl-sotalol. *Circulation: Arrhythmia and Electrophysiology, 5*(2), 400-408.

Cahalan. (1975). Modification of sodium channel gating in frog myelinated nerve fibres by Centruroides sculpturatus scorpion venom. *The Journal of Physiology, 244*(2), 511-534.

Candia, Garcia, & Latorre. (1992). Mode of action of iberiotoxin, a potent blocker of the large conductance Ca (2+)-activated K+ channel. *Biophysical Journal, 63*(2), 583-590.

Carvalho, Pereira, & Cardoso. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics, 8*(8), 832.

Celik, O'Brien, Brennan, Rainbow, Dart, Zheng, . . . Barrett-Jolley. (2020). Deep-Channel uses deep neural networks to detect single-molecule events from patch-clamp data. *Communications Biology 2020 3:1, 3*(1), 1-10. doi:10.1038/s42003-019-0729-3

Chambers, Witton, Adams, Marrington, & Kammonen. (2016). High-throughput screening of NaV1. 7 modulators using a giga-seal automated patch clamp instrument. *Assay and drug development technologies, 14*(2), 93-108.

Chari, & Pachter. (2023). The specious art of single-cell genomics. *Plos Computational Biology, 19*(8), e1011288. doi:10.1371/journal.pcbi.1011288

Colquhoun, Hatton, & Hawkes. (2003). The quality of maximum likelihood estimates of ion channel rate constants. *The Journal of Physiology, 547*(3), 699-728.

Contreras, Neely, Alvarez, Gonzalez, & Latorre. (2012). Modulation of BK channel voltage gating by different auxiliary β subunits. *Proceedings of the National Academy of Sciences, 109*(46), 18991-18996.

Cox. (2014). Modeling a Ca2+ channel/BKCa channel complex at the single-complex level. *Biophysical Journal, 107*(12), 2797-2814.

Cox. (2015). CHAPTER 1 Ion Channel Drug Discovery: a Historical Perspective. In *Ion Channel Drug Discovery* (pp. 1-15): The Royal Society of Chemistry.

Cox, Cui, & Aldrich. (1997). Allosteric gating of a large conductance Ca-activated K+ channel. *The Journal of general physiology, 110*(3), 257-281.

Dani. (2015). Neuronal Nicotinic Acetylcholine Receptor Structure and Function and Response to Nicotine. In (pp. 3-19): Elsevier.

Davies, Purves, Barrett-Jolley, & Dart. (2010). Interaction with caveolin-1 modulates vascular ATP-sensitive potassium (K ATP) channel activity. *J Physiol, 588*, 3255-3266. doi:10.1113/jphysiol.2010.194779

Delaney, Brophy, & Ward. (2019). Synthesis of realistic ECG using generative adversarial networks. *arXiv preprint arXiv:1909.09150*.

Deng, Dong, Socher, Li, Li, & Fei-Fei. (2009). *Imagenet: A large-scale hierarchical image database.* Paper presented at the 2009 IEEE conference on computer vision and pattern recognition.

Denson, Duchatelle, & Eaton. (1994). The effect of racemic ketamine on the large conductance Ca+ 2-activated potassium (BK) channels in GH3 cells. *Brain Research, 638*(1-2), 61-68.

Denson, & Eaton. (1994). Ketamine inhibition of large conductance Ca (2+)-activated K+ channels is modulated by intracellular Ca2+. *American Journal of Physiology-Cell Physiology, 267*(5), C1452-C1458.

Donahue, McAuley, & Puckette. (2019). Adversarial Audio Synthesis. *arXiv pre-print server*. doi:None

arxiv:1802.04208v3

Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, . . . Gelly. (2010). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv 2020. *arXiv preprint arXiv:2010.11929*.

Doyle, Cabral, Pfuetzner, Kuo, Gulbis, Cohen, . . . MacKinnon. (1998). The Structure of the Potassium Channel: Molecular Basis of K⁺ Conduction and Selectivity. *Science, 280*(5360), 69-77. doi:doi:10.1126/science.280.5360.69

Dreyfus. (1962). The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications, 5*(1), 30-45. doi:https://doi.org/10.1016/0022-247X(62)90004-5

Du, Bautista, Yang, Diez-Sampedro, You, Wang, . . . Cui. (2005). Calcium-sensitive potassium channelopathy in human epilepsy and paroxysmal movement disorder. *Nature Genetics, 37*(7), 733-738.

Faust, Hagiwara, Hong, Lih, & Acharya. (2018). Deep learning for healthcare applications based on physiological signals: A review. *Computer Methods and Programs in Biomedicine, 161*, 1-13.

Fedida, Orth, Chen, Lin, Plouvier, Jung, . . . Beatch. (2005). The mechanism of atrial antiarrhythmic action of RSD1235. *Journal of Cardiovascular Electrophysiology, 16*(11), 1227-1238.

Feetham, Nunn, Lewis, Dart, & Barrett-Jolley. (2015). TRPV 4 and KCa ion channels functionally couple as osmosensors in the paraventricular nucleus. *British Journal of Pharmacology, 172*(7), 1753-1768.

Ffrench-Constant, Williamson, Davies, & Bass. (2016). Ion channels as insecticide targets. *Journal of Neurogenetics, 30*(3-4), 163-177. doi:10.1080/01677063.2016.1229781

Fride, Skolnick, & Arora. (1990). Immunoenhancing effects of alprazolam in mice. *Life sciences, 47*(26), 2409-2420.

Gaita, Giustetto, Bianchi, Schimpf, Haissaguerre, Calò, . . . Wolpert. (2004). Short QT syndrome: pharmacological treatment. *Journal of the American College of Cardiology, 43*(8), 1494-1499.

Galvez, Gimenez-Gallego, Reuben, Roy-Contancin, Feigenbaum, Kaczorowski, & Garcia. (1990). Purification and characterization of a unique, potent, peptidyl probe for the high conductance calcium-activated potassium channel from venom of the scorpion Buthus tamulus. *Journal of Biological Chemistry, 265*(19), 11083-11090. doi:https://doi.org/10.1016/S0021-9258(19)38560-6

Geng, Deng, Zhang, Budelli, Butler, Yuan, . . . Magleby. (2020). Coupling of Ca2+ and voltage activation in BK channels through the αB helix/voltage sensor interface. *Proceedings of the National Academy of Sciences, 117*(25), 14512-14521.

Geng, & Magleby. (2015). Single-channel kinetics of BK (Slo1) channels. *Frontiers in Physiology, 5*, 532.

Gibb, Ogden, McDaniel, Vance, Kell, Butch, . . . Traynelis. (2018). A structurally derived model of subunit-dependent NMDA receptor function. *The Journal of Physiology, 596*(17), 4057-4089.

Gill, Ramos-Rodriguez, & Raine. (2012). Combined pesticide exposure severely affects individual- and colony-level traits in bees. *Nature, 491*(7422), 105-108. doi:10.1038/nature11585

Gnanasambandam, Nielsen, Nicolai, Sachs, Hofgaard, & Dreyer. (2017). Unsupervised Idealization of Ion Channel Recordings by Minimum Description Length: Application to Human PIEZO1-Channels. *Frontiers in Neuroinformatics, 0*, 31-31. doi:10.3389/FNINF.2017.00031

González, Baez-Nieto, Valencia, Oyarzún, Rojas, Naranjo, & Latorre. (2012). K+ channels: function-structural overview. *Comprehensive physiology, 2*(3), 2087-2149.

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, . . . Bengio. (2014). Generative adversarial nets. *Advances in neural information processing systems, 27*.

Hammond. (2015). Ionic gradients, membrane potential and ionic currents. *Cell Mol Neurophysiol*, 39-54.

Handelman, Kok, Chandra, Razavi, Huang, Brooks, . . . Asadi. (2019). Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods. *American Journal of Roentgenology, 212*(1), 38-43.

Harris, Millman, van der Walt, Gommers, Virtanen, Cournapeau, . . . Oliphant. (2020). Array programming with NumPy. *Nature, 585*(7825), 357-362. doi:10.1038/s41586-020-2649-2

Harz, & Hegemann. (1991). Rhodopsin-regulated calcium currents in Chlamydomonas. *Nature, 351*(6326), 489-491. doi:10.1038/351489a0

Haswell, Phillips, & Rees. (2011). Mechanosensitive Channels: What Can They Do and How Do They Do It? *Structure, 19*(10), 1356-1369. doi:https://doi.org/10.1016/j.str.2011.09.005

He, Zhang, Ren, & Sun. (2015). Deep Residual Learning for Image Recognition. *arXiv pre-print server*. doi:None

arxiv:1512.03385

Heinemann, & Sigworth. (1988). Open channel noise. IV. Estimation of rapid kinetics of formamide block in gramicidin A channels. *Biophysical Journal, 54*(4), 757-764.

Henton, & Tzounopoulos. (2021). What's the buzz? The neuroscience and the treatment of tinnitus. *Physiological Reviews, 101*(4), 1609-1632.

Hille. (1971). The Permeability of the Sodium Channel to Organic Cations in Myelinated Nerve. *Journal of General Physiology, 58*(6), 599-619. doi:10.1085/jgp.58.6.599

Hille. (1975). The receptor for tetrodotoxin and saxitoxin. A structural hypothesis. *Biophysical Journal, 15*(6), 615.

Hille. (1978). Ionic channels in excitable membranes. Current problems and biophysical approaches. *Biophysical Journal, 22*(2), 283-294.

Hille. (1992). Potassium channels and chloride channels. In Ionic Channels of Excitable Membrane. *Edited by Hill B*, 130-133.

Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, & Schmidhuber. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/NECO.1997.9.8.1735

Hodgkin, & Huxley. (1939). Action potentials recorded from inside a nerve fibre. *Nature, 144*(3651), 710-711.

Hodgkin, & Huxley. (1952). The components of membrane conductance in the giant axon of Loligo. *J. Physiol*, 473-496.

Hollon, Pandian, Adapa, Urias, Save, Khalsa, . . . Lewis. (2020). Near real-time intraoperative brain tumor diagnosis using stimulated Raman histology and deep neural networks. *Nature medicine, 26*(1), 52-58.

Holzinger, Langs, Denk, Zatloukal, & Müller. (2019). Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9*(4), e1312.

Hornik, Stinchcombe, & White. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366.

Horrigan, & Aldrich. (2002). Coupling between voltage sensor activation, Ca2+ binding and channel opening in large conductance (BK) potassium channels. *The Journal of general physiology, 120*(3), 267-305.

Hotz, Schutte, Sieling, Polupanow, Diederichsen, Steinem, & Munk. (2013). Idealizing Ion Channel Recordings by a Jump Segmentation Multiresolution Filter. *Ieee Transactions on Nanobioscience, 12*(4), 376-386. doi:10.1109/tnb.2013.2284063

Howe. (1996). Homomeric and heteromeric ion channels formed from the kainate-type subunits GluR6 and KA2 have very small, but different, unitary conductances. *Journal of Neurophysiology, 76*(1), 510-519. doi:10.1152/jn.1996.76.1.510

Hu, Bell, Antani, Xue, Yu, Horning, . . . Befano. (2019). An observational study of deep learning and automated evaluation of cervical images for cancer screening. *JNCI: Journal of the National Cancer Institute, 111*(9), 923-932.

Hubel, & Wiesel. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology, 160*(1), 106.

Hunter. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*(3), 90-95. doi:10.1109/mcse.2007.55

Hurst, Latorre, Toro, & Stefani. (1995). External barium block of Shaker potassium channels: evidence for two binding sites. *The Journal of general physiology, 106*(6), 1069-1087.

Jakob, Klesen, Allegrini, Darkow, Aria, Emig, . . . Beyersdorf. (2021). Piezo1 and BKCa channels in human atrial fibroblasts: Interplay and remodelling in atrial fibrillation. *Journal of Molecular and Cellular Cardiology, 158*, 49-62.

John, & Stefan. (2001). Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. In *A Field Guide to Dynamical Recurrent Networks* (pp. 237-243): IEEE.

Juette, Terry, Wasserman, Altman, Zhou, Zhao, & Blanchard. (2016). Single-molecule imaging of non-equilibrium molecular ensembles on the millisecond timescale. *Nature Methods, 13*(4), 341-344.

Kalia, Milescu, Salvatierra, Wagner, Klint, King, . . . Bosmans. (2015). From foe to friend: using animal toxins to investigate ion channel function. *Journal of Molecular Biology, 427*(1), 158-175.

Karras, Laine, & Aila. (2019a). *A style-based generator architecture for generative adversarial networks.* Paper presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.

Karras, Laine, & Aila. (2019b). A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv pre-print server*. doi:None

arxiv:1812.04948v3

Katz, & Miledi. (1972). The statistical nature of the acetylcholine potential and its molecular components. *The Journal of Physiology, 224*(3), 665-699. doi:10.1113/jphysiol.1972.sp009918

Kelley. (1960). Gradient Theory of Optimal Flight Paths. *ARS Journal, 30*(10), 947-954. doi:10.2514/8.5282

Kienker. (1989). Equivalence of Aggregated Markov Models of Ion-Channel Gating. *236*(1284), 269-309. doi:10.1098/rspb.1989.0024

Kim. (2014). Channelopathies. *Korean Journal of Pediatrics, 57*(1), 1. doi:10.3345/kjp.2014.57.1.1

Kim, & Nimigean. (2016). Voltage-gated potassium channels: a structural examination of selectivity and gating. *Cold Spring Harbor perspectives in biology, 8*(5), a029231.

Kingma, & Ba. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koppenhöfer, & Schmidt. (1968). Effect of scorpion venom on ionic currents of the node of ranvier: I. Die Permeabilitäten P Na und PK. *Pflügers Archiv, 303*, 133-149.

Larget. (1998). A canonical representation for aggregated Markov processes. *Journal of applied probability, 35*(2), 313-324.

Latorre, Castillo, Carrasquel-Ursulaez, Sepulveda, Gonzalez-Nilo, Gonzalez, & Alvarez. (2017). Molecular determinants of BK channel functional diversity and functioning. *Physiological Reviews, 97*(1), 39-87.

Laumonnier, Roger, Guérin, Molinari, M'rad, Cahard, . . . Elia. (2006). Association of a functional deficit of the BK Ca channel, a synaptic regulator of neuronal excitability, with autism and mental retardation. *American Journal of Psychiatry, 163*(9), 1622-1629.

Leavesley, & Rich. (2016). Overcoming limitations of FRET measurements. *Cytometry. Part A: the journal of the International Society for Analytical Cytology, 89*(4), 325.

LeCun. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Boser, Denker, Henderson, Howard, Hubbard, & Jackel. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*(4), 541-551. doi:10.1162/neco.1989.1.4.541

Lee, & Cui. (2010). BK channel activation: structural and functional insights. *Trends in Neurosciences, 33*(9), 415-423.

Lewis, Feetham, Gentles, Penny, Tregilgas, Tohami, . . . Barrett-Jolley. (2013). Benzamil sensitive ion channels contribute to volume regulation in canine chondrocytes. *British Journal of Pharmacology, 168*(7), 1584-1596. doi:10.1111/j.1476-5381.2012.02185.x

Lewis, & Garcia. (2003). Therapeutic potential of venom peptides. *Nature Reviews Drug Discovery, 2*(10), 790-802. doi:10.1038/nrd1197

Lucchesi, & Moczydlowski. (1991). On the interaction of bovine pancreatic trypsin inhibitor with maxi Ca (2+)-activated K+ channels. A model system for analysis of peptide-induced subconductance states. *The Journal of general physiology, 97*(6), 1295-1319.

Martinac. (2017). Single-molecule FRET studies of ion channels. *Progress in Biophysics and Molecular Biology, 130*, 192-197. doi:https://doi.org/10.1016/j.pbiomolbio.2017.06.014

Martinez. (1987). Ion transport and water movement. *Journal of Dental Research, 66*(1_suppl), 638-647.

Massah, Gharaghani, Lordejani, & Asakere. (2016). New and mild method for the synthesis of alprazolam and diazepam and computational study of their binding mode to GABA A receptor. *Medicinal Chemistry Research, 25*, 1538-1550.

McCulloch, & Pitts. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115-133.

McHugh. (2012). Interrater reliability: the kappa statistic. *Biochemia medica, 22*(3), 276-282.

McInnes, Healy, & Melville. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. Retrieved from https://arxiv.org/abs/1802.03426v3

McKinney. (2010). *Data Structures for Statistical Computing in Python*.

McKinney, Sieniek, Godbole, Godwin, Antropova, Ashrafian, . . . Darzi. (2020). International evaluation of an AI system for breast cancer screening. *Nature, 577*(7788), 89-94.

McLeod. (2004). Clinical pharmacokinetics of nateglinide: a rapidly-absorbed, short-acting insulinotropic agent. *Clinical pharmacokinetics, 43*, 97-120.

McManus, & Magleby. (1988). Kinetic states and modes of single large-conductance calcium-activated potassium channels in cultured rat skeletal muscle. *The Journal of Physiology, 402*(1), 79-120.

Miljanich. (2004). Ziconotide: neuronal calcium channel blocker for treating severe chronic pain. *Curr Med Chem, 11*(23), 3029-3040. doi:10.2174/0929867043363884

Milne, Borgnia, Bartesaghi, Tran, Earl, Schauder, . . . Subramaniam. (2013). Cryo-electron microscopy - a primer for the non-microscopist. *Febs Journal, 280*(1), 28-45. doi:10.1111/febs.12078

Mirza, & Osindero. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mobasheri, Lewis, Maxwell, Hill, Womack, & Barrett-Jolley. (2010). Characterization of a stretch-activated potassium channel in chondrocytes. *Journal of Cellular Physiology, 223*(2), 511-518. doi:Doi 10.1002/Jcp.22075

Murakami, & Kijima. (2000). Transduction ion channels directly gated by sugars on the insect taste cell. *Journal of General Physiology, 115*(4), 455-466. doi:10.1085/jgp.115.4.455

Naccarelli, Wolbrette, Samii, Banchs, Penny-Peterson, Stevenson, & Gonzalez. (2008). Vernakalant–a promising therapy for conversion of recent-onset atrial fibrillation. *Expert Opinion on Investigational Drugs, 17*(5), 805-810.

Neher, & Sakmann. (1976). Single-channel currents recorded from membrane of denervated frog muscle fibres. *Nature, 260*(5554), 799-802.

Nicolai, & Sachs. (2013). Solving Ion Channel Kinetics With The QuB Software. *Biophysical Reviews and Letters, 08*(03n04). doi:10.1142/S1793048013300053

Nielsen, Schroeder, & Lewis. (2000). Structure–activity relationships of ω-conotoxins at N-type voltage-sensitive calcium channels. *Journal of Molecular Recognition, 13*(2), 55-70.

Nimigean, & Magleby. (2000). Functional coupling of the ß1 subunit to the large conductance Ca2+-activated K+ channel in the absence of Ca2+ increased Ca2+ sensitivity from a Ca2+-independent mechanism. *The Journal of general physiology, 115*(6), 719-736.

Nitze, Schulthess, & Asche. (2012). Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification. *Proceedings of the 4th GEOBIA, Rio de Janeiro, Brazil, 79*, 3540.

Noguchi, Wang, Pan, & Welsh. (2012). Fibroblast circadian rhythms of PER2 expression depend on membrane potential and intracellular calcium. *Chronobiology international, 29*(6), 653-664.

Numata, Sato-Numata, & Yoshino. (2021). BK channels are activated by functional coupling with L-type Ca2+ channels in cricket myocytes. *Frontiers in Insect Science, 1*, 662414.

O'Brien, Feetham, Staunton, Hext, & Barrett-Jolley. (2022). Temperature modulates PVN pre-sympathetic neurones via transient receptor potential ion channels. *bioRxiv*, 2022.2001. 2026.477880.

Olivera. (1997). EE Just Lecture, 1996: Conus venom peptides, receptor and ion channel targets, and drug design: 50 million years of neuropharmacology. *Molecular biology of the cell, 8*(11), 2101-2109.

Pein, Bartsch, Steinem, & Munk. (2021). Heterogeneous Idealization of Ion Channel Recordings – Open Channel Noise. *Ieee Transactions on Nanobioscience, 20*(1), 57-78. doi:10.1109/TNB.2020.3031202

Pein, Eltzner, & Munk. (2021). Analysis of patchclamp recordings: model-free multiscale methods and software. *European Biophysics Journal, 50*(2), 187-209. doi:10.1007/s00249-021-01506-8

Pein, Tecuapetla-Gomez, Schutte, Steinem, & Munk. (2018). Fully automatic multiresolution idealization for filtered ion channel recordings: Flickering event detection. *Ieee Transactions on Nanobioscience, 17*(3), 300-320. doi:10.1109/TNB.2018.2845126

Perrais, & Ropert. (1999). Effect of zolpidem on miniature IPSCs and occupancy of postsynaptic GABAA receptors in central synapses. *Journal of Neuroscience, 19*(2), 578-588.

Pinney, MacMullen, Becker, Lin, Hanna, Thornton, . . . Stanley. (2008). Clinical characteristics and biochemical mechanisms of congenital hyperinsulinism associated with dominant K ATP channel mutations. *The Journal of clinical investigation, 118*(8), 2877-2886.

Qin. (2004). Restoration of Single-Channel Currents Using the Segmental k-Means Method Based on Hidden Markov Modeling. *Biophysical Journal, 86*(3), 1488-1488. doi:10.1016/S0006-3495(04)74217-4

Qin, Wang, Yao, Zhou, Zhao, Wang, & Huang. (2020). Using a one-dimensional convolutional neural network with a conditional generative adversarial network to classify plant electrical signals. *Computers and Electronics in Agriculture, 174*, 105464. doi:10.1016/j.compag.2020.105464

Quinton. (1990). Cystic fibrosis: a disease in electrolyte transport. *The FASEB Journal, 4*(10), 2709-2710. doi:10.1096/fasebj.4.10.2197151

Radford, Metz, & Chintala. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ramachandram, & Taylor. (2017). Deep multimodal learning: A survey on recent advances and trends. *IEEE signal processing magazine, 34*(6), 96-108.

Reid, & Zhao. (2014). The electrical response to injury: molecular mechanisms and wound healing. *Advances in wound care, 3*(2), 184-201.

Rim, Sung, Min, & Hong. (2020). Deep Learning in Physiological Signal Data: A Survey. *Sensors, 20*(4), 969.

Roden. (2008). Long-QT Syndrome. *New England Journal of Medicine, 358*(2), 169-176. doi:10.1056/nejmcp0706513

Rodríguez-Soriano. (1998). Bartter and related syndromes: the puzzle is almost solved. *Pediatric Nephrology, 12*(4), 315-327. doi:10.1007/s004670050461

Ronneberger, Fischer, & Brox. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9351*, 234-241. Retrieved from https://arxiv.org/abs/1505.04597v1

Rothberg, & Magleby. (1998). Kinetic structure of large-conductance Ca2+-activated K+ channels suggests that the gating includes transitions through intermediate or secondary states: a mechanism for flickers. *The Journal of general physiology, 111*(6), 751-780.

Rumelhart, Hinton, & Williams. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536. doi:10.1038/323533a0

Safavi-Hemami, Brogan, & Olivera. (2019). Pain therapeutics from cone snail venoms: From Ziconotide to novel non-opioid pathways. *Journal of Proteomics, 190*, 12-20. doi:10.1016/j.jprot.2018.05.009

Sakmann, & Neher. (1984). Patch Clamp Techniques for Studying Ionic Channels in Excitable Membranes. *Annual Review of Physiology, 46*(1). doi:10.1146/annurev.ph.46.030184.002323

Schuster, Roewer, Schneiderbanger, & Johannsen. (2014). Management of malignant hyperthermia: diagnosis and treatment. *Therapeutics and Clinical Risk Management*, 355. doi:10.2147/tcrm.s47632

Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman. (2017). *Stock price prediction using LSTM, RNN and CNN-sliding window model.* Paper presented at the 2017 international conference on advances in computing, communications and informatics (icacci).

Serviss, Gådin, Eriksson, Folkersen, & Grandér. (2017). ClusterSignificance: a bioconductor package facilitating statistical analysis of class cluster separations in dimensionality reduced data. *Bioinformatics, 33*(19), 3126-3128.

Seyler, Li, Schweizer, Katus, & Thomas. (2014). Inhibition of cardiac two-pore-domain K+ (K2P) channels by the antiarrhythmic drug vernakalant–comparison with flecainide. *European Journal of Pharmacology, 724*, 51-57.

Shelley, Niu, Geng, & Magleby. (2010). Coupling and cooperativity in voltage activation of a limited-state BK channel gating in saturating Ca2+. *Journal of General Physiology, 135*(5), 461-480.

Sidach, & Mintz. (2002). Kurtoxin, a gating modifier of neuronal high-and low-threshold ca channels. *Journal of Neuroscience, 22*(6), 2023-2034.

Siekmann, Larry, Yule, Fox, Bryant, Edmund, & Sneyd. (2011). MCMC Estimation of Markov Models for Ion Channels. *Biophysical Journal, 100*(8), 1919-1929. doi:10.1016/j.bpj.2011.02.059

Sigworth. (1985). Open channel noise. I. Noise in acetylcholine receptor currents suggests conformational fluctuations. *Biophysical Journal, 47*(5), 709-720.

Sigworth, & Sine. (1987). Data transformations for improved display and fitting of single-channel dwell time histograms. *Biophysical Journal, 52*(6), 1047-1054.

Simon, Bindra, Mansfield, Nelson-Williams, Mendonca, Stone, . . . Bakkaloglu. (1997). Mutations in the chloride channel gene, CLCNKB, cause Bartter's syndrome type III. *Nature Genetics, 17*(2), 171-178.

Singh, McGoldrick, Saotome, & Sobolevsky. (2018). X-ray crystallography of TRP channels. *Channels, 12*(1), 137-152.

Sivilotti, & Colquhoun. (2016). In praise of single channel kinetics. *Journal of General Physiology, 148*(2), 79-88.

Siwy, & Fuliński. (2002). Origin of 1/f Noise in Membrane Channel Currents. *Physical Review Letters, 89*(15). doi:10.1103/physrevlett.89.158101

Snutch, Peloquin, Mathews, & McRory. (2013). Molecular properties of voltage-gated calcium channels. In *Madame Curie Bioscience Database [Internet]*: Landes Bioscience.

Supratak, Dong, Wu, & Guo. (2017). DeepSleepNet: a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG. *Ieee Transactions on Neural Systems and Rehabilitation Engineering, 25*(11), 1998-2008. doi:10.1109/tnsre.2017.2721116

Teramoto, Aishima, Zhu, Tomoda, Yunoki, Takahashi-Yanaga, . . . Ito. (2004). Effects of U-37883A on intracellular Ca2+-activated large-conductance K+ channels in pig proximal urethral myocytes. *European Journal of Pharmacology, 506*(1), 1-7.

Terlau, Heinemann, Stühmer, Pusch, Conti, Imoto, & Numa. (1991). Mapping the site of block by tetrodotoxin and saxitoxin of sodium channel II. *FEBS letters, 293*(1-2), 93-96.

Timmer, & Koenig. (1995). On generating power law noise. *Astronomy and Astrophysics, 300*, 707. Retrieved from https://ui.adsabs.harvard.edu/abs/1995A&A...300..707T

Truong, Kuhlmann, Bonyadi, Querlioz, Zhou, & Kavehei. (2019). Epileptic seizure forecasting with generative adversarial networks. *Ieee Access, 7*, 143999-144009.

Vakili, Ghamsari, & Rezaei. (2020). Performance analysis and comparison of machine and deep learning algorithms for IoT data classification. *arXiv preprint arXiv:2001.09636*.

Van der Maaten, & Hinton. (2008a). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(11).

Van Der Maaten, & Hinton. (2008b). Visualizing Data using t-SNE. *Journal of Machine Learning Research, 9*, 2579-2605.

Vargas-Alarcon, Alvarez-Leon, Fragoso, Vargas, Martinez, Vallejo, & Martinez-Lavin. (2012). A SCN9A gene-encoded dorsal root ganglia sodium channel polymorphism associated with severe fibromyalgia. *BMC musculoskeletal disorders, 13*, 1-5.

Veech, Kashiwaya, & King. (1995). The resting membrane potential of cells are measures of electrical work, not of ionic currents. *Integrative Physiological and Behavioral Science, 30*, 283-307.

Viterbi. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory, 13*(2), 260-269.

Wang, Liu, DeBerg, Nomura, Hoffman, Rohde, . . . Selvin. (2014). Single molecule FRET reveals pore size and opening mechanism of a mechano-sensitive ion channel. *Elife, 3*, e01834.

Watkins. (2019). Cannabinoid interactions with ion channels and receptors. *Channels, 13*(1), 162-167. doi:10.1080/19336950.2019.1615824

Welsh. (1990). Abnormal regulation of ion channels in cystic fibrosis epithelia; Abnormal regulation of ion channels in cystic fibrosis epithelia. *The FASEB Journal, 4*(10), 2718-2725. doi:10.1096/fasebj.4.10.1695593

White. (1995). Separation of K+-and Cl−-selective ion channels from rye roots on a continuous sucrose density gradient. *Journal of Experimental Botany, 46*(4), 361-376. doi:10.1093/jxb/46.4.361

Wold, Esbensen, & Geladi. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems, 2*(1-3), 37-52.

Wong, Gatt, Stamatescu, & McDonnell. (2016). *Understanding data augmentation for classification: when to warp?* Paper presented at the 2016 international conference on digital image computing: techniques and applications (DICTA).

Wood, Williams, & Waldron. (2004). Patch clamping by numbers. *Drug Discovery Today, 9*(10), 434-441. doi:https://doi.org/10.1016/S1359-6446(04)03064-8

Xiao, & Zhou. (2020). *Research progress of RNN language model.* Paper presented at the 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).

Yamakura, Chavez-Noriega, & Harris. (2000). Subunit-dependent Inhibition of Human Neuronal Nicotinic Acetylcholine Receptors and Other Ligand-gated Ion Channels by Dissociative Anesthetics Ketamine and Dizocilpine. *Anesthesiology, 92*(4), 1144-1153. doi:10.1097/00000542-200004000-00033

Yoshida, Oda, & Ikemoto. (1991). Kinetics of the Ca2+-activated K+ channel in rat hippocampal neurons. *The Japanese Journal of Physiology, 41*(2), 297-315.

Young. (1936). The structure of nerve fibres in cephalopods and crustacea. *Proceedings of the Royal Society of London. Series B-Biological Sciences, 121*(823), 319-337.

Yu, Dhingra, Dick, & Galán. (2017). Effects of ion channel noise on neural circuits: an application to the respiratory pattern generator to investigate breathing variability. *Journal of Neurophysiology, 117*(1), 230-242.

Zamoyski, Serebryakov, & Schubert. (1989). Activation and blocking effects of divalent cations on the calcium-dependent potassium channel of high conductance. *Biomedica biochimica acta, 48*(5-6), S388-392.

Zhang, Li, Cai, Zhang, Chen, & Liu. (2021). Over-fitting suppression training strategies for deep learning-based atrial fibrillation detection. *Medical & Biological Engineering & Computing, 59*, 165-173.

Zhu, Ye, Fu, Liu, & Shen. (2019). Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. *Scientific Reports, 9*(1), 1-11.

Zubcevic, Herzik, Chung, Liu, Lander, & Lee. (2016). Cryo-electron microscopy structure of the TRPV2 ion channel. *Nature Structural & Molecular Biology, 23*(2), 180-186. doi:10.1038/nsmb.3159