Identification of Locust Neuronal Systems with Recurrent Neural Networks

by

Evander D’Costa

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Environmental and Life Sciences
Biological Sciences

January 2019
System identification (SI) approaches have been widely used in the neurosciences for studying neuronal behaviour. In producing statistical models of neuronal data for SI, most approaches choose to use models which are easily interpretable. Interpretable modelling allows researchers to efficiently derive the implied dynamics of the neuronal system(s) under study and gain insight into its function.

The recent increase in computational power and availability higher resolution experimental techniques have allowed researchers in the neurosciences to study increasingly complex neuronal systems. This thesis suggests that some of the widely used interpretable SI models may be inadequate for modelling the growing corpus of big and complex datasets.

Recently, there has been a surge of academic interest in using Artificial Neural Networks (ANNs) for modelling complex datasets. ANNs are one of the hot focii in the machine learning literature, and are increasingly being used in other fields as a tool for modelling, control and classification. The main objective of this thesis is to use a variety of ANNs called Recurrent Neural Networks (RNNs) for modelling time-series datasets derived from neuronal systems.

In achieving this objective, this thesis investigates the efficacy of RNNs in modelling a series of neurons in the locust hind leg. First, the graded potentials of locust Fast Extensor Tibiae (FETi) motor neuron are modelled. The performance characteristics of the trained RNNs are compared with Linear-Nonlinear-Linear (LNL) and Time-Delay Neural Networks (TDNNs) previously used for the FETi modelling task. Next, RNN modelling techniques are adapted for modelling the spiking action potentials of locust proprioceptive afferents. The functionality of the afferents are investigated using the trained models and compared to results achieved by previous work using Wiener methods. Finally, RNNs are used to model the dynamics of the locust Slow Extensor Tibiae (SETi) motor neuron. The trained models are used to simulate the functionality of SETi, and the results are used to compare the efficacy of using open-loop against closed-loop setups in electrophysiological experiments.

Although the techniques developed in this thesis are used and validated in modelling simple insect neuronal systems; this thesis highlights the general issues and pitfalls associated with using RNNs as a tool for system identification. The techniques used in this thesis can be used in modelling any neuronal system, given enough data and computational capacity.
# Contents

List of Figures                                                                                   vii
List of Tables                                                                                   xiii
Declaration of Authorship                                                                       xv
Acknowledgements                                                                               xvii

1 Introduction                                                                                   1
   1.1 Computational neuroscience and big data                                                  2
   1.2 Artificial neural networks                                                               2
   1.3 Functional modelling of neuronal systems                                                4
   1.4 Aims and Contributions                                                                   5

2 Literature Review                                                                            7
   2.1 Modelling and System identification in Neuroscience                                      7
   2.2 Machine learning and Artificial Neural Networks                                         10
       2.2.1 Recurrent Neural Networks                                                           12

3 Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron 19
   3.1 Introduction                                                                            19
   3.2 Methods                                                                                21
       3.2.1 Collecting the Data                                                                  21
       3.2.2 Analysing the Data                                                                  23
       3.2.3 Preprocessing the Data                                                              25
       3.2.4 Experiment 1: Cross-Validation and Training the Models                               26
       3.2.5 Experiment 2: Modelling Randomised Data                                              29
       3.2.6 Experiment 3: Physiological Interpretation of the Models                            30
   3.3 Results                                                                                30
       3.3.1 Experiment 1: Training the Models                                                   31
       3.3.2 Experiment 2: Modelling Randomised Data                                              34
       3.3.3 Experiment 3: Biological Plausibility of the Models                                  34
   3.4 Discussion                                                                             37
       3.4.1 Training the Models                                                                  37
       3.4.2 Evaluating biological plausibility of the models                                     40
   3.5 Conclusion                                                                             42

4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents 43
   4.1 Introduction                                                                            43
   4.2 Methods                                                                                45
       4.2.1 Data Extraction                                                                      45
List of Figures

2.1 The structure of a simple Recurrent Neural Network (RNN). Inputs $x_t$ are used to compute an intermediate (hidden) representation $h_t$. Per-timestep outputs $y_t$ are computed directly from the hidden representations. The hidden activations for a timestep $t$ is a function of the hidden activations, i.e. recurrence is in the hidden layer (left) and can be unfolded for multiple timesteps (right). .......................... 13

3.1 Locust hind leg physiology. Input signals are applied to the apodeme via the forceps. Output stimuli are recorded from the FETi motor neuron using a micro-electrode. ........................................................................ 23

3.2 (a) Time-varying response from the FETi neuron to changes in femoro-tibial angle stimulated with 50 Hz Gaussian White Noise (GWN). (b) FETi response to changes in femoro-tibial angle when stimulated with a 5 Hz sinusoid. (c) Frequency spectrum of FETi response to 50 Hz bandlimited GWN. (d) Frequency content of FETi response to 5 Hz sinusoidal input. In (a) and (b), negative swings in femoro-tibial angle correspond to flexion, which elicits a strong response from the neuron. The output frequency spectrum (c) is roughly consistent with the 50 Hz bandwidth at the input. Output spectra for a 5 Hz sinusoidal input shows sharp, but decaying spectral content at multiples of 5 Hz, along with some background noise. ................................................................. 25

3.3 Illustration of chunking FETi data with time windows: (a) 2s of FETi data (1000 timepoints). (b), (c) Two chunks of the FETi data derived from windows of 500 timepoints each. Each chunk encompasses 1s of data. ......................... 27

3.4 Normalised Mean Squared Errors on validation data for RNN, LSTM and GRU models: (a) Random hyperparameter combinations chosen before any hyperparameter tuning displayed poorly converging models. (b) After hyperparameter tuning with random search, the errors drop significantly indicating model convergence to better performing estimators. ................................................................. 31

3.5 The spread of Mean Squared Errors for each model, trained on each of the 5 locusts. (a) Recurrent Neural Network; (b) Long Short-Term Memory and (c) Gated Recurrent Unit models. Locusts 1 and 4 show the largest errors for all models. .................................................. 32

3.6 Plot of FETi response vs. model response for the same input on 1s of unseen test data. The model shows a reasonably close fit to the FETi response indicating that it has learned the underlying neuronal dynamics. .............................. 33

3.7 Spread of test-set errors for RNNs trained on randomised GWN data. .................. 33

3.8 Plot of FETi response vs. model response for a 5 Hz input stimulus. Here, positive femoral-tibial angles denote flexion and negative angles denote extension. Both neuronal and model response is high in flexion, and inactive in extension. There appears to be a close fit between model response and FETi response, despite the higher frequency variation in the FETi responses. .................................................. 35
3.9 Frequency content by Fourier coefficients. (a) Frequency content of FETi output from Locust 3 for 5 Hz stimulus. (b) Frequency content from model trained on FETi data with 5 Hz stimulus. (c) Frequency content after passing a 5 Hz sinusoid through a half-wave rectifier. The frequency harmonics of a half-wave rectifier appear similar to the frequency content from FETi data and RNN models, but lack the harmonics at 15 Hz, 25 Hz and 35 Hz. 36

3.10 Delay between input peak and output peak at various frequencies. There is roughly a 20-23ms delay between input and output for all frequencies. 37

4.1 (a) Illustration of the locust hind leg showing the femoral chordotonal organ (FeCO) apodeme. The apodeme was grasped with forceps, and cut distal to the forceps. Electrical signals from afferents were collected using an electrode placed intracellularly in axons. (b) A Gaussian white noise stimulus (left) applied to the apodeme with forceps elicits spiking responses from the sensory afferents (right). 46

4.2 (a) 3.5s of recorded spiking activity. (b) A closeup of approximately 70ms shows a group of spikes with various amplitudes occurring within a 30ms timespan. Each spike has a duration of approximately 10ms. 47

4.3 Illustration of the procedure used for binarising spikes. (a) A continuous-valued signal (blue) is thresholded, in this case to values above 0. Consecutive groups of timepoints with values above the threshold are grouped together (green). (b) The maximal value of each group is taken as the discretised value of where a spike occurs. (c) Binarising a continuous-valued electrode recording (above) in this way creates a discretised spike-train (below). 49

4.4 Clustering spikes by distinguishing features. (a) Each spike can be characterised by its amplitude and the time before the previous spike. (b) Normalising these numerical features in the range [0, 1] and plotting each spike as a point in a 2D graph shows that these can be clustered with algorithms like K-means clustering. Clusters with the largest relative amplitude were chosen for discretisation and modelling. 50

4.5 Illustration of the windowing procedure used to split a stimulus-discretised spike-train dataset into chunks of data. Overlapping windows of 180ms are used to split each chunk. Successive chunks are not entirely independent, with 20ms of overlap between them. 52

4.6 Examples of receiver operating characteristic (ROC) curves plotting the proportion of true positives in a prediction set to the proportion of false positives in the prediction set as the probability threshold for two predicting classifiers is varied from 0 to 1. (a) An example of a ‘good’ classifier where the true positive rate stays close to 1 as the threshold is varied. (b) A relatively worse classifier which has lower true positive rates as the probability threshold is varied. In both cases the dashed line indicating equal true-positive rates and false-positive rates is indicative of the worst type of classifier which is no different from random guessing. The area under the ROC curves (AUC) can be considered as a scalar metric of the quality of a classifier. Here, a perfect classifier has an AUC of 1 and a random-guessing classifier has an AUC of 0.5. 53

4.7 Illustrations of Singular Spectrum Analysis (SSA) used to detrend time series. (a) A repeated sinusoidal signal (top) is trended by adding an \( x^2 \) component (middle). SSA is used to detrend the drifting signal, recovering the original structure with much smaller drift magnitudes (bottom). (b) Using SSA removes similar drift in recordings of the sensory afferents, retaining the spiking structure. 55

4.8 The percentage of timepoints with spikes to in all afferent recordings. The most densely spiking datasets are approximately 10% spiking with a majority having between 1%-5% spikes. These statistics are for discretised spike-trains sampled at 1000Hz. 57
4.9 Test data spike trains with respective predictions from data trained with (above) cross-entropy loss. (below) weighted cross-entropy loss. The models trained with cross-entropy loss overfit to predicting ‘no-spike’ for every timepoint, eliciting no spiking for any stimuli. Models trained with weighted cross-entropy are penalised higher for false negatives and therefore learn to predict ‘spikes’ for the stimuli most likely to elicit these from the afferent. .................................................. 58

4.10 Illustrations of the change in cross-entropy on training and validation data for successive training iterations (epochs). (above) The training and validation loss for a cross-entropy loss function. (below) The training and validation loss for a weighted cross-entropy loss function. In both cases, successive training iterations see the training loss decrease rapidly. The losses on validation data correspondingly decrease, despite not being used for training. For the weighted cross-entropy loss case, validation losses saturate due to the increased false-negative penalties. 59

4.11 Final area under receiver operating characteristic (AUC) performance scores on test data for models trained on all 35 afferents. For comparison, typical scores for a random, uninformative classifier should be 0.5. AUC scores for a perfect classifier should be 1. This figure shows that all models form good estimators of their respective afferents. ........................................................................... 60

4.12 Distribution of AUC scores for models trained on randomised Gaussian white noise inputs. The spread on each of the 35 recordings represents 50 models trained on ‘randomised’ versions of the same recording. Here the mean AUCs are roughly centered around 0.5, displaying the characteristics of a random classifier. ........ 61

4.13 Illustrative examples of the types of spike-triggered averaged (STA) stimuli observed from the models. These are illustrated as normalised tibial angle between [-1, 1] plotted against time. Positive tibial angles correspond to tibial flexion. (a) A monophasic (single peak) STA indicating position sensitivity. (b) Biphasic STAs indicating velocity with two different polarities in the left and right graphs. (c) Triphasic STAs indicating acceleration sensitivity with two different polarities in the left and right graphs, respectively. .................................................. 62

4.14 All spike-triggered averages (STA) from the RNN models plotted together. (a) All the STAs from the models. (b) Position-sensitive STAs. (c) Velocity-sensitive STAs of opposite polarities in the left and right graphs. (d) Acceleration-sensitive STAs of opposite polarities in the left and right graphs. (e) STAs with no clear sensitivity observable from their patterns. .................................................. 63

4.15 Examples of STAs with ambiguous sensitivity. (Above) Two predominant peaks followed by a smaller peak at 40 ms, indicating a predominant velocity component with a small sensitivity to acceleration. (Below) A larger negative (extension) peak indicating position sensitivity surrounded by two relatively smaller peaks indicating sensitivity to all of position, velocity and acceleration; with predominant position sensitivity. .................................................. 65

4.16 Spread of AUCs for the four observed STA transitions. Here, models trained on higher-frequency afferent data are used for predicting the responses to lower-frequency inputs from the same afferent. The models’ responses are compared with the real afferent responses to derive an AUC score. For most of the transition types, the scores are centered around 0.5 indicating poor estimators. There are outliers in the velocity-to-velocity transition types and acceleration-to-acceleration transition types. Additionally, the velocity-to-acceleration transition types have a large variance of estimator AUCs with a large inter-quartile range and a higher median than the other transition types. .................................................. 66
4.17 STAs of cases where models trained on higher frequency inputs show high AUC scores when predicting the afferent responses to lower frequency inputs. In all cases, the model STAs on lower frequency inputs are identical to model STAs on their original high frequency input data. (a) Acceleration-acceleration transition STAs with an AUC of 0.71, here all of the STAs are similar, with the low-frequency model STA lagging less than the higher frequency model STAs. (b) Velocity-velocity transition STAs with an AUC of 0.92. Here all STAs are nearly identical. (c) Velocity-acceleration sensitive STAs with an AUC of 0.72. Here, the STA of the high-frequency model’s prediction shares a large extension (negative) peak with the STA of the low frequency model.

5.1 (A) Locust fixation in modelling clay. (B) Attachment of the shaker mounted forceps to the FeCO apodeme. (C) Lateral face of the femoro-tibial joint showing the vertically positioned tibia with cuticle removed from the distal section of the femur. (D) Exposed accessory flexor muscle and apodeme of the femoral chordotonal organ, false coloured in blue and red respectively.

5.2 (A) Pathways of the four-identified motor neurones innervating the ETi muscle, showing extensions from the ventral cortex of the metathoracic ganglion through nerve 3,4 and 5 towards the ETi muscle. Including Fast Extensor Tibiae Motor Neuron (FETi); Slow Extensor Tibiae Motor Neuron (SETi); Common Inhibitory Neuron (CIi) and Dorsal Unpaired Median Neuron (DUMETi). Diagram adapted from Burrows (1996). (B) Intact ventral surface of the thorax. (C) Exposed metathoracic ganglion (yellow). (D) Nerves 3 and 5 shown following the removal of the left main trachea, false coloured in blue and red respectively.

5.3 Example of a Gaussian white noise signal used to drive displacements of the FeCO apodeme.

5.4 Displacement of the FeCO apodeme as femoro-tibial joint increases from full flexion to full extension of the leg (Dewhirst et al. (2013); Burrows (1996)).

5.5 (a) A typical spike train recording from SETi. (b) A closeup of the spike train shows spikes of different shapes in the recording. (c) A further closeup shows a burst of continuous electrical activity within 7 ms, with no refractory period.

5.6 (a) An input signal (blue) is thresholded above a predefined value (green). (b) Consecutive timepoints above the threshold are grouped, and inflection point for each group is identified as the spike. (c) Using this procedure on continuous-valued spiking outputs discretises a spike train (top) to a boolean output of ‘spike’ or ‘no-spike’ (bottom).

5.7 Each point represents a spike represented by two features: minimum and maximum normalised spike amplitude. This figure shows three distinct clusters of spikes in this features space. Clusters were selected based on their spike-triggered averages (STAs).

5.8 The stimulus-response data is separated into independent chunks of 400ms each. This illustration shows the effect of windowing on a SETi response spike train. Here, successive windows overlap by 200ms.

5.9 Illustration of the three walking gaits used for testing the final trained models. These were derived by Angarita-Jaimes et al. (2012) using high-speed cameras.

5.10 Typical progress of binary cross-entropy loss training and validation set. Here, every epoch is a complete evaluation through the training dataset.

5.11 Spread of AUC scores for models tested on each of the three walking gaits. (a) Models trained on SETi data from closed-loop experiments. (b) Models trained on SETi data from open-loop experiments. Open-loop models show higher interquartile ranges. Closed-loop models show lower interquartile range and higher median scores.

5.12 Distribution of AUC scores where all models are trained on randomised gaussian white noise inputs and tested on the three walking test sets. For each test dataset, the spread is centered around a mean AUC of 0.5, displaying the characteristics of a random classifier.
5.13 For novel GWN inputs, (a) Plot of all spike-eliciting stimuli from a single model for 150ms before a spike. Stimuli are plotted by opacity for the probability of eliciting a spike from the model. The STA, or mean of all these stimuli is plotted in red. (b) STAs from all the trained models plotted together. (c) The mean of all the model’s STAs. .................................................. 90

5.14 Spike-triggered averages (STA) for RNNs trained on open-loop data vs closed-loop data. (a) (left) The STAs of all closed-loop models. (right) The mean STA of closed-loop models. (b) (left) The STAs of all open-loop models. (right) The mean STA of open-loop models. Apart from a single anomaly among the closed-loop STAs, both groups of STAs appear identical, with a strong peak indicating response to flexion over a span of approximately 60ms. ......................... 91

5.15 Spread of AUC scores for the model of each SETi queried on the test data derived from all other SETi. Here, the x-axis denotes the identity of each model and the corresponding boxplot is the spread of scores in using that model to predict the walking-gait test data of all other models. ................................. 92

5.16 Spread of AUC errors for (a) Models trained on closed-loop SETi data predicting the responses of both closed- and open-loop SETi data. (b) Models trained on open-loop SETi data predicting the responses of closed- and open-loop SETi data. ................................. 92

5.17 Typical response from RNN models trained on SETi data on a sinusoidal stimulus. (Above) The output probability of spiking from the model. (Below) The input stimulus expressed as the tibial angle normalised to [-1, 1]. Positive stimulus values are tibial flexion and negative values are extension. ......................... 94

5.18 Typical responses from RNN models trained on SETi data on a half-wave rectified sinusoid biased toward flexion. (a) With an input sinusoidal amplitude of 1. (b) An input amplitude of 0.1 with dramatically reduced probability of spiking. (c) An illustration of how the maximum spiking probability varies with the amplitude of the half-wave rectified sinusoid. ................................. 95

5.19 Typical responses from RNN models trained on SETi data on a half-wave rectified sinusoid biased toward extension. (a) Returning the tibial angle to neutral after extension results in high spiking probability immediately after the return. (b) The spiking probability pattern persists even if the return is into extension. (c) An illustration of how the output spiking probability varies with the input rate of return from extension. Here, 0 corresponds to a return to a neutral tibial angle and 1 corresponds to pure tibial extension. (d) An illustration of how the output spiking probability varies with the velocity of return from extension. Here, the spiking probability is plotted against frequency of the half-wave rectified stimuli. ................................. 96

5.20 Typical model responses to a step stimuli: (a) A step towards full flexion shows a sustained output of high spiking probability. (b) A step toward full extension shows no response from the model. ................................. 96

5.21 Examples of receiver operating characteristic (ROC) curves plotting the proportion of true positives in a prediction set to the proportion of false positives in the prediction set as the probability threshold for two predicting classifiers is varied from 0 to 1. (a) An example of a ‘good’ classifier where the true positive rate stays close to 1 as the threshold is varied. (b) A relatively worse classifier which has lower true positive rates as the probability threshold is varied. In both cases the dashed line indicating equal true-positive rates and false-positive rates is indicative of the worst type of classifier which is no different from random guessing. The area under the ROC curves (AUC) can be considered as a scalar metric of the quality of a classifier. Here, a perfect classifier has an AUC of 1 and a random-guessing classifier has an AUC of 0.5. ................................. 98
# List of Tables

3.1 Mean NMSE of 20 trials for each model type across all locusts. Lowest errors are highlighted in bold. .......................................................... 32

3.2 Standard deviation of NMSE for 20 trials across all locusts. Lower standard deviations are highlighted in bold. ......................................................... 34

3.3 Comparison of NMSE scores for RNN, LSTM, GRU, TDNN and LNL models across all five locusts. Lowest errors are highlighted in bold. .................. 34

3.4 Distribution of p-values for Mann Whitney U tests comparing the MSE scores of models. P-values less than 0.05 are shown in bold. ............................ 34

3.5 Mean Square Error of RNN, TDNN and LNL models predicting the output for a 5 Hz constant stimulus. Lowest errors are highlighted in bold. ............... 35

A.1 Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. Window length and Window overlap were parameters of the windowing scheme discussed in Chapter 3. Train %, Validation % and Test % are the proportions of data assigned to training data, validation data and test data respectively. .................. 111

A.2 Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on FETi data. Number of hidden layers denotes the number of intermediate layers between input and output for the networks. Hidden units per layer denotes the number of neurons in the intermediate layer(s). Output layer activation function denotes the nonlinear activation function used in the hidden layer(s). Network initialisation procedure denotes the network initialisation procedure (see Chapter 3). Dropout probability denotes the extent of dropout applied to the network as regularisation. ........... 111

A.3 Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. Learning rate denotes the size of the gradient step per iteration used during optimisation. Alpha is an Adam-specific parameter used to denote the momentum used during gradient descent. Number of training epochs is the number of epochs used in training; where each epoch is a complete iteration through the dataset. Minibatch size denotes the size of each minibatch used to estimate stochastic gradient. ......................................................... 112

B.1 Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. Window length and Window overlap were parameters of the windowing scheme discussed in Chapter 3. Threshold for grouping spike points denotes the threshold used for the normalised output amplitude beyond which points were grouped together as a single spike for discretisation. Train %, Validation % and Test % are the proportions of data assigned to training data, validation data and test data respectively. ............... 113
B.2 Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on sensory afferent data. Number of hidden layers denotes the number of intermediate layers between input and output for the networks. Hidden units per layer denotes the number of neurons in the intermediate layer(s). Output layer activation function denotes the nonlinear activation function used in the hidden layer(s). Network initialisation procedure denotes the network initialisation procedure (see Chapter 3). Dropout probability denotes the extent of dropout applied to the network as regularisation.

B.3 Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. Learning rate denotes the size of the gradient step per iteration used during optimisation. Alpha is an Adam-specific parameter used to denote the momentum used during gradient descent. Number of training epochs is the number of epochs used in training; where each epoch is a complete iteration through the dataset. Minibatch size denotes the size of each minibatch used to evaluate stochastic gradients at every iteration.

C.1 Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. Window length and Window overlap were parameters of the windowing scheme discussed in Chapter 3. Threshold for grouping spike points denotes the threshold used for the normalised output amplitude beyond which points were grouped together as a single spike for discretisation. Train %, Validation % and Test % are the proportions of data assigned to training data, validation data and test data respectively. The test data was held at 0% as the models were tested against natural stimuli from walking gaits.

C.2 Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on SETi data. Number of hidden layers denotes the number of intermediate layers between input and output for the networks. Hidden units per layer denotes the number of neurons in the intermediate layer(s). Output layer activation function denotes the nonlinear activation function used in the hidden layer(s). Network initialisation procedure denotes the network initialisation procedure (see Chapter 3). Dropout probability denotes the extent of dropout applied to the network as regularisation.

C.3 Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. Learning rate denotes the size of the gradient step per iteration used during optimisation. Alpha is an Adam-specific parameter used to denote the momentum used during gradient descent. Number of training epochs is the number of epochs used in training; where each epoch is a complete iteration through the dataset. Minibatch size denotes the size of each minibatch used to evaluate stochastic gradients at every iteration.
Declaration of Authorship

I, Evander D’Costa, declare that this thesis entitled Identification of Locust Neuronal Systems with Recurrent Neural Networks and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others, this is always clearly attributed;

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

5. I have acknowledged all main sources of help;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. Either none of this work has been published before submission, or parts of this work have been published as: .

Signed:

Date:
Acknowledgements

To my supervisors, Phil and David, whose continuous support and guidance through good times and bad was invaluable. I could not have done this without them;
To Oliver, Alicia and Devon, the giants whose shoulders I’m privileged to build my work on and who helped immeasurably with the electrophysiology;
To Lorraine, Tracey, Nicki and my cohort and fellow PG researchers at ICSS, Biological Sciences and ISVR, for the years of friendship and camaraderie;
To mum and dad, whose persistant cajoling was invaluable in getting me through the tough times;
To Livvy for her love, support and cakes;
To everyone else who helped in big and small ways.
Chapter 1

Introduction

Achieving a complete understanding of the human brain is one of the greatest challenges of modern science. The brain contains around 86 billion neurons and an equal number of glial cells (Dayan et al. (2003); Markram (2006)). Each neuron has orders of 10-10,000 synaptic connections with other neurons in the brain. These synaptic connections form complex structures such as hierarchical networks, feedback loops and lateral connections (Dayan et al. (2003)). These intricate and complex structures perform correspondingly complex functions such as learning, memory storage, and spatial/temporal pattern recognition (Mullally and Maguire (2014); Garagnani et al. (2008)).

Achieving an understanding how substructures of the human brain operate promises to open innovative areas of research in several technological fields. For example, engineering problems such as power consumption, memory and storage in current generations of computing devices (Furber et al. (2014)); Artificial Intelligence and commercialised robotics (Asada et al. (2001)). Even in medicine, an understanding of the biomolecular structure of the brain can lead to targeted cures for neurodegenerative diseases like Alzheimer’s and Motor Neurone disease.

Neuroscience is dedicated to understanding the structure and function of the brain. It encompasses a variety of scientific approaches ranging from molecular and cellular biology (Bear et al. (2007)) to psychophysics (Read (2015)) and neuropsychology (Walsh (1978)). However, these approaches are often mutually disparate and require some form of common ground towards achieving a common understanding of the brain (Dayan et al. (2003)).

Computational Neuroscience aims to unify the many different neuroscience approaches. Its core principle is to use computational/mathematical modelling to bridge these approaches and provide compact representations of what’s been learned (Dayan et al. (2003)). Modelling can elucidate which neuronal operations might be required to perform a certain function. Models can suggest further experiments exploring the biological plausibility of such operations, improving the overall understanding of the brain (Wu et al. (2006)).

The immense challenges of understanding the fine-grained structures of the human brain mean that neuroscientists require ever more powerful techniques to meet them. Recent advances in experimental methods using optical techniques (Zhu et al. (2017)) and electron microscopy (Mo...
et al. (2018)) have given impressive levels of resolution into the cellular and biomolecular organiza-
tion of neuronal and synaptic systems. However, with higher levels of resolution comes larger amounts of data (Sejnowski et al. (2014)). The classical approach of ad-hoc computational/mathematical modelling may not be able to keep up with this unprecedented amount of experimental data (Markram et al. (2011)). Thus, more generic modelling frameworks are required: frameworks sophisticated enough to learn complex patterns from large datasets, whilst being relatively simple to adopt, implement and analyse.

1.1 Computational neuroscience and big data

As a field of study, neuroscience is a vastly diverse one. Globally, there are several labs which study different parts of the nervous system. These labs use a wide range of techniques - from electron microscopy to calcium imaging: to study neuronal characteristics on a wide range of spatial and temporal scales (Mo et al. (2018)). For example, object recognition might be studied with fMRI in the case of humans (Vuilleumier et al. (2002)), or by optical techniques in mice (Hillman (2007)). These result in vastly different datasets which must be standardised before meaningful comparisons are made and crucial insights can be drawn.

As technology advances, and datasets grow and become more complex, it will become increasingly difficult to analyse data and extract conclusions. For example, the neuroscience team at the National Tsing Hua University in Hsinchu City, Taiwan, took a full decade to image 60,000 neurons in the Drosophilia brain (Chiang et al. (2011)). This is less than half of the estimated 135,000 neurons in the Drosophilia brain. At this rate, achieving a similar connectomic model of the 86 billion neurons in the human brain would take an estimated 17 million years! Keeping up with the increasingly higher-resolution data generated by neuroscience experiments requires new modelling approaches. These approaches must be able to analyse vast quantities of data and find patterns in high-dimensional spaces. These approaches must also provide succinct, accurate descriptions of crucial neurological phenomena. These descriptions can inform neuroscientists about critical neuronal behaviour, and can help guide further experiments. The aim of this thesis is to test the efficacy of neuronal time series modelling using Recurrent Neural Networks (RNNs). RNNs are a variety of artificial neural networks (ANNs) tailored for time series modelling (LeCun et al. (2015)).

1.2 Artificial neural networks

Artificial Neural Networks (ANNs) are a type of computational model popular in machine learning research (LeCun et al. (2015)). In short, ANNs are collections of simple units or ‘artificial neurons’. These units are loosely analogous to biological neurons. Like their biological counterparts, artificial neurons weigh incoming presynaptic signals and combine them. A nonlinear function then ‘activates’ the weighted presynaptic signal to yield a certain output. In large interconnected networks, ANNs can learn the underlying features of a dataset from examples rather than being explicitly programmed.
McCulloch and Pitts (1943) created the first computational model of what was then known of biological neuronal function. This model used simple thresholding logic which could easily be implemented in electronic systems at the time. The weights of this simple model had to be explicitly programmed.

Later Rosenblatt (1958) created the **Perceptron** algorithm. This algorithm trained simple artificial neurons on simple binary classification tasks. Perceptron allowed threshold-logic artificial neurons to learn pattern recognition from examples, rather than being explicitly programmed.

A key breakthrough came in 1986, with the **Backpropagation** algorithm (Rumelhart et al. (1986)). The backpropagation minimises an overall objective function by calculating gradients of the objective function with respect to weights of an ANN. The algorithm then proceeds with adjusting network weights in the direction of these gradients, thus iteratively optimising the performance of the ANN (Rumelhart et al. (1986); LeCun et al. (2015)). For the first time, backpropagation allowed ANNs to learn very complex patterns much more effectively than the simpler perceptron. ANNs could now be stacked in layers for more complexity.

In the early days, backpropagation’s effectiveness was limited. ANNs with more than 2-3 stacked layers often got stuck in poor local optima and showed poor results in benchmark problems. ANNs therefore declined in popularity in favour of Support Vector Machines, Linear classifiers and other simpler machine learning algorithms (LeCun et al. (2015)).

In the mid-2000s Hinton et al. (2006)’s and Bengio et al. (2007)’s work on **unsupervised pretraining** heralded what is currently a very hot focus of machine learning known as **Deep Learning**. In their respective works, Hinton et al. (2006) and Bengio et al. (2007) showed that a greedy layerwise pre-training procedure could train ANNs with many stacked layers and a high degree of complexity to highly efficient optima (LeCun et al. (2015)). Their results encouraged the academic community to explore 'deep' architectures (ANNs with more than one hierarchical layer) on new machine learning tasks.

The post-2006 resurgence of popularity allowed academics to demonstrate deep ANN architectures which are able to far outperform alternative machine learning techniques on a wide variety of benchmark tasks. These include image recognition (Krizhevsky et al. (2012)), speech recognition (Amodei et al. (2016)), natural language processing (Bahdanau et al. (2014)) and collaborative filtering (Zhou et al. (2008)). Additionally, ANNs demonstrated an uncanny ability to perform esoteric machine learning tasks like describing the scene in an image using natural language (Xu et al. (2015)) or even the inverse task of generating an image of a scene from only the natural language description of that scene (Reed et al. (2016)).

The current generation of ANNs shows that they can learn the underlying features and patterns from very large, high-dimensional and complex datasets (Goodfellow et al. (2016); LeCun et al. (2015)). The recent success of these models and their ongoing popularity with industry and academia means large amounts of research and funding towards effective techniques for training ANNs on complex data. The aim of this thesis is to demonstrate the efficacy of these techniques for the functional modelling task in neuroscience research. The hope is that the continuing research and development of ANNs for modelling complex datasets can lend itself to the neuroscience modelling task.
1.3 Functional modelling of neuronal systems

The goal of modelling in neuroscience is to describe a functional relationship between a set of inputs into a neurological system, and that system’s neuronal responses (Dayan et al. (2003)). A high-fidelity functional model of a neuronal system can provide a quantitative description of that system’s behaviour. It can also serve as a guide to further experiments with the system (Wu et al. (2006)).

If the underlying dynamics of a system being studied is unknown, it can be viewed as a black box. There is a host of mathematical techniques devoted to the statistical modelling black box systems (Ljung (1987)). Computational neuroscientists use black box models at various levels of granularity - from studying individual neurons to analysing the functionality of entire clusters of neurons (Dayan et al. (2003)). Among the engineering and signal processing community, this family of modelling techniques is also called System Identification (Ljung (1987)). Variants of System Identification/black box modelling go by many names in neuroscience literature: spike-triggered analysis, reverse correlation and white-noise analysis (Dayan et al. (2003); Wu et al. (2006); Paninski (2003); Chichilnisky (2001)). In all of these, the goal is to estimate a function that describes the way input stimuli are mapped to neuronal responses.

Upon modelling the statistical properties of a black box system, the fidelity of the model can be tested by providing it with natural signals that the neuronal system can be expected to experience. The responses of both model and system to these signals can be compared. Responses that are anomalous can inform further experiments into the system. This approach can reveal novel principles of neuronal dynamics, even in the absence of a prior theoretical or quantitative model (Wu et al. (2006)).

One of the downsides of black box modelling is the sheer variety of modelling techniques, each with its own set of assumptions about the underlying phenomena it’s trying to learn. One of the most reported modelling techniques is a nonlinear, nonparametric set of models known as Wiener/Volterra kernel methods (Marmarelis (2004)).

Despite their success in modelling and informing experiments on neuronal systems, Wiener methods and related methods have their limitations. When faced with increasingly large datasets, the parameters of these models need to scale exponentially to encompass the increasing complexity (Goodfellow et al. (2016)). This makes them harder to train making these models prone to converging onto poor local optima (Victor (2005)).

In comparison, Artificial Neural Networks (ANNs) show the ability to encompass vast levels of complexity by stacking layers (Goodfellow et al. (2016); LeCun et al. (2015)). With the right hyperparameters, the use of stochastic gradient descent methods implemented on parallel computing hardware means ANNs with millions/billions of parameters can be trained in reasonable time (Goodfellow et al. (2016); Kingma and Ba (2014); Sanders and Kandrot (2010)). With the right set of initial conditions, these ANNs can converge to good optima for learning complex patterns on large datasets (Goodfellow et al. (2016); LeCun et al. (2015)).

In this thesis, ANNs are used to model the functional characteristics of a set of neurons in the hind leg of the Schistocerca gregaria, commonly known as the Desert Locust. This animal was chosen as a focus of study because:
Chapter 1 Introduction

1) It possesses a scaled-down neurological system with lower levels of complexity than higher animals like primates (Burrows (1996)). Such a system provides the perfect opportunity for a set of modelling experiments, saving time and resources in developing and debugging novel computational methods. Additionally, from a software development perspective, it is good practice to develop tools alongside a simple system before deploying it to the larger task at hand (Runeson (2006)).

2) Insects, primates and other animals all possess a common evolutionary heritage. This is especially true of nervous systems (Bear et al. (2007)). The modelling tools developed in this thesis should, in principle, be easily extended to neurological modelling in any other animal.

3) Studying the locust builds on a pre-established body of work studying neurological phenomena in the locust hind leg (Meruelo et al. (2016); Newland and Kondoh (1997a,b); Dewhirst et al. (2013); Angarita-Jaimes et al. (2012)). The results of this work can help provide corroboration for the results uncovered in this thesis.

In each of the femurs of the locust is a collection of 90 neurons called the femoro-tibial chordotonal organ (FeCO) (Burrows (1996)). A thin cuticular apodeme anchors the FeCO to the locust’s tibia. The FeCO is responsible for measuring tibial angular position, velocity and acceleration relative to the femur (Newland and Kondoh (1997a,b)). It is also the entry point to a reflexive feedback loop to the flexor and extensor motor neurons in the locust hind leg (Burrows (1996)).

Understanding the underlying neurological processes in the FeCO and neighbouring sensory/motor neurons in the locust hind leg can yield a better understanding of how neuromotor processes work in human beings. This knowledge can lend itself to innovative technological applications like high-fidelity bionic prostheses and advanced robotics.

1.4 Aims and Contributions

The overarching aim of this thesis is to test the efficacy of using a variety of artificial neural networks (ANNs) for modelling neuronal systems. In particular, this thesis uses a variety of ANN called recurrent neural networks (RNNs) for modelling time series. RNNs are used to model and study sensory and motor neurons responsible for feedback control in the locust hind leg. The neurological implications of the dynamics modelled by RNNs are corroborated against previous work on modelling the respective neurons. Where applicable, comparisons are provided against competing modelling techniques previously used for the same task.

Chapter 2) presents a background of the literature on system identification (SI) and black box modelling in neuroscience. Additionally, a literature review of ANNs and RNNs is provided. Here, the background literature and justification for modelling techniques used in subsequent chapters is presented.

Chapter 3) presents RNNs used for modelling the locust Fast Extensor Tibiae (FETi) motor neuron. Here, previously derived data from FETi using Gaussian white noise (GWN) stimuli are used. The RNNs are used to model FETi’s continuous-valued responses to the stimuli. More natural stimuli are then used to query the RNNs for expected FETi behaviour. These behaviours are then compared to results on previous work in modelling FETi.
Chapter 4) presents RNNs used for modelling sensory afferents in the locust hind leg. Here, previously derived data using GWN stimuli are used and the corresponding spiking responses from the afferents are modelled using a discrete formulation of RNNs. A number of technical challenges to getting good performance out of RNNs for this task are detailed. Accuracy metrics for evaluating the quality of the trained RNNs are described. Baselines for comparing the quality of these RNNs are established. Finally, the first-order dynamics of the afferents as implied by the RNNs are compared to previous work in modelling the same afferents.

Chapter 5) presents RNNs for modelling the locust Slow Extensor Tibiae (SETi) motor neuron. As in Chapter 4, RNNs are trained on spiking neuron data derived using synthetic GWN stimuli. SETi behaviours to more natural walking stimuli are then simulated by querying the trained RNNs with these stimuli. Additionally, separate models are trained on data derived from SETi in open-loop conditions to compare against data derived in closed-loop conditions. The expected behaviour of SETi in both setups are compared with the aim of comparing the widely used experimental open-loop setup with a more natural closed-loop setup.

Chapter 6) concludes with a discussion of what was achieved in the experiments in the previous chapters. The general modelling procedure using RNNs is outlined. The positive and negative points of using RNNs in neuroscience modelling are discussed. This is accompanied with a statement of the feasibility for using RNN modelling techniques in other neuroscience experiments.

None of the modelling techniques used in these chapters is inherently truly original. These have been developed and described by other researchers, and appropriately acknowledged as such. The original contributions of this thesis are in combining these techniques to solve inherent difficulties with using RNNs for the system identification of neuronal data. These include issues with data preprocessing, data imbalance, hyperparameter search, regression/categorical modelling, reporting evaluation metrics, establishing baselines for comparing metrics, and using the trained models for deriving the functionality of neuronal systems without interpretable coefficients.

Additionally, none of the data used for system identification were derived for this thesis. These were extracted by others, who are acknowledged where applicable. Apart from high-level software libraries, all software developed for analysing, preprocessing and training RNNs on this data was developed by the author.
Chapter 2

Literature Review

2.1 Modelling and System identification in Neuroscience

Modelling is a widely used tool for studying neurophysiological systems (Dayan et al. (2003)). Computational modelling in particular, is an essential part of modern neuroscience (Sterratt et al. (2011); Dayan et al. (2003)). The computational modelling paradigm provides an objective method for analysing and describing how neurons process information (Sterratt et al. (2011); Dayan et al. (2003)). Quantitative and computational models are widely used to describe sensory systems in simple animals (Frye and Dickinson (2001)), neuronal processing of auditory (McAlpine (2005)) and visual systems (Carandini et al. (2005)) of mammals and in the primate primary visual cortex (Carandini et al. (1997)).

Recent efforts in modelling physiological systems have used tools borrowed from the field of system identification (SI). SI defines a set of statistical methods for modelling the underlying dynamics of systems using measured data (Ljung (1987); Sjöberg et al. (1995)). SI tools are especially powerful for describing the functionality of black box systems, where the dynamics of these systems is a-priori unknown (Sjöberg et al. (1995)), and is only learnable from data.

In the neuroscience literature, SI approaches have a variety of names including: receptive field estimation (Theunissen et al. (2001)), spike-triggered analysis (Dayan et al. (2003)) and reverse correlation (Alonso et al. (2001)). The general approach in all of these is to estimate the dynamical function which describes how stimuli are mapped to neuronal responses. The variety of methods available for performing this function estimation means that the derived stimulus-response models are also referred to by many names including: transfer function (Marmarelis and Marmarelis (1978)), stimulus-response function (Sahani and Linden (2003)), kernel (Marmarelis (2004)) and receptive field (Theunissen et al. (2001)). Wu et al. (2006) provided an excellent analysis of the various SI approaches used in sensory neuroscience, and described these under a common framework of maximum a-posteriori (MAP) estimation.

Neuronal systems which fit the black box description are particularly amenable to SI modelling approaches (Wu et al. (2006)). For these systems, the inputs can be tightly controlled and queried in experiments using synthetic stimuli. If the behaviour of a neuronal system queried by synthetic stimuli closely matches its behaviour with natural stimuli in vivo, generalised models of
the system can be derived using SI approaches. This is why SI modelling has been more successful at modelling neurons in the sensory periphery than more central neurons (Wu et al. (2006)). Central neurons exhibit more complex behaviour than peripheral sensory neurons (Carandini et al. (2005)). Central neurons typically process more abstract semantic information than their sensory counterparts (Gallant et al. (1996); Kobatake and Tanaka (1994)). Their behaviour is more likely to be modulated by extraneous signals like attention and memory which do not depend on directly controllable inputs (Gallant et al. (1996); Kobatake and Tanaka (1994)). These are important factors which need to be considered when applying choosing an SI approach for neuronal modelling.

SI approaches lend a variety of statistical model classes to the neuronal modelling task. Each model class embodies assumptions about the underlying functional properties of the neuronal system being modelled. The simplest model class used in the literature is the linear model (Marmarelis (2004)). Linear models have been used to model auditory (Eggermont et al. (1983)), visual (DeAngelis et al. (1993)) and somatosensory (Arabzadeh et al. (2005)) systems. Linear models are simple to train, but cannot capture more complex nonlinear dynamics inherent in neuronal processing (Wu et al. (2006)).

An alternative technique used in the literature is to apply a cascade of transformations to transform the input stimulus into an estimate of neuronal response. One example is the linearisation strategy, which applies a nonlinear transformation to the input so that the relationship between transformed stimulus and neuronal response is more linear (Aertsen and Johannesma (1981)). A linear model is then fit to the nonlinearly transformed input. For example, Theunissen et al. (2000) showed that cascading an idealised model of a cochlea followed by a linear filter formed better estimators of avian auditory neuronal function.

Another popular cascading technique is to apply a bank of linear filters to the input followed by a nonlinear transformation (Dayan et al. (2003)). This linear-nonlinear (LN) cascade filter is then followed by either a linear filter (LNL) to model continuous-valued neuronal responses (Dewhirst et al. (2013)) or a poisson filter (LNP) to model spiking neurons (Schwartz et al. (2006)). For example, Dewhirst et al. (2013) used LNL cascades to describe the functional dynamics of locust femorotibial extensor tibiae (FETi) neurons. Additionally, Chichilnisky (2001) used LNP cascades to model the light response properties of retinal ganglion cells.

The most popular model class used to describe nonlinear neuronal behaviour is the Wiener (Wiener (1938)) or Volterra (Volterra (1959)) series. This model class represents an input-output relationship as a functional power series expansion, and can model systems with nonlinear dynamics and finite memory (Marmarelis (2004)).

A Wiener/Volterra series expansion consists of multiple orders of terms. The first order terms of the model represent the linear properties of the system being modelled (Marmarelis (2004)). Lee and Schetzen (1965) showed that the first-order coefficients of a Wiener/Volterra series expansion is identical to the spike-triggered average (STA) of a neuron stimulated by gaussian white noise (GWN). The first-order kernel of the Wiener/Volterra series therefore provides an unbiased estimate of the neurons linear receptive field (Dayan et al. (2003)). The second-, third-, and higher-order kernels of a Wiener/Volterra expansion model nonlinear behaviour (Marmarelis (2004); Marmarelis and Marmarelis (1978); Wu et al. (2006)). In practice, it has been unfeasible to estimate terms of Wiener/Volterra models beyond the second order, because the amount of
data required to estimate these terms scales exponentially with the order of the model (Victor (2005)).

Marmarelis (2004) provides an excellent treatment of the theory behind Volterra series and Wiener kernel analysis. The popularity of Wiener/Volterra models can be highlighted with several examples. Marmarelis and Naka (1973a) used Wiener theory to demonstrate the functional dynamics of a three-stage neuron chain in the catfish retina (Marmarelis and Naka (1973c,b)). Kondoh et al. (1995), Newland and Kondoh (1997a) and Newland and Kondoh (1997b) used Wiener models to analyse the function of neurons in the locust hind leg, including both sensory afferents and extensor-tibiae motor neurons. Sensory afferents from the coso-basal (Gamble and DiCaprio (2003)) and thoracic-coxal (DiCaprio (2003)) chordotonal organ in the crab were also modelled using similar techniques. Jing et al. (2012) used dominant feature analysis to characterise the receptive field of neurons for position-, velocity- and acceleration-sensitivity from first order Wiener/Volterra kernels. Additionally, Jing et al. (2012) showed that useful insights could be drawn about the nonlinear behaviour of a neuron from second-order Wiener/Volterra kernels. In theory, Wiener/Volterra models are powerful methods for providing quantitative descriptions of the nonlinear dynamics of a system (Marmarelis (2004)).

In practice, Wiener/Volterra models are limited. These are sensitive to noise in the data, and can be high-variance estimators of system dynamics (Gamble and DiCaprio (2003); Korenberg and Hunter (1990)). Conversely, over-parametrising these models to compensate for noise reduces their ability to generalise to out-of-sample data (Angarita-Jaimies et al. (2012)). Additionally, higher parametrisations require exponentially more data to train (Palm and Poggio (1977b)).

An important debate in the neuron modelling literature is over the use of synthetic stimuli vs natural stimuli for modelling (Wu et al. (2006)). The most prolific model classes in the literature require synthetic stimuli which statistically query a wide range of a neurons receptive field. A popular stimulus class is Gaussian white noise (GWN) (Newland and Kondoh (1997b,a); Dewhirst et al. (2013); Wu et al. (2006); Marmarelis and Marmarelis (1978); Marmarelis and Naka (1973a); Marmarelis (2004)). GWN signals are information-rich stimuli because they contain every possible frequency with a Gaussian distribution on signal amplitude (Newland and Kondoh (1997b)). In fact, estimating the kernels of a Wiener series for modelling a neuron requires quering the neuron with GWN (Marmarelis and Marmarelis (1978); Marmarelis (2004)). On the other hand, natural stimuli have a much richer and more complex statistical structure than synthetic stimuli like GWN (Simoncelli and Olshausen (2001); David et al. (2004)).

There is no guarantee that a model developed using GWN stimuli will accurately reflect neuronal function in natural conditions. In the literature, there is no consensus on the use of one type of stimulus over the other. Some studies have compared models of sensory neurons developed using synthetic and natural stimuli David et al. (2004); Yu et al. (2005); Woolley et al. (2005). For these studies, the models in both cases broadly agree, apart from a consistent demonstration of broadband synthetic stimuli eliciting substantially more inhibition than natural stimuli (David et al. (2004); Wu et al. (2006)). On the other hand, natural stimuli can contain strong correlations and can lead to systematic biases in modelling and estimation (David et al. (2004); Woolley et al. (2005)).
Other studies have taken a simpler approach: aiming to find the stimuli which most elicit a response from the neuron (Paninski et al. (2007)). These studies use a range of methods ranging from exhaustive search of optimal stimuli for simple neurons with a small stimulus space (Machens et al. (2005)), to heuristic search methods to identify stimuli which elicit strong neuronal responses for more complex neurons (Földiák (2001)). Although no consensus exists, most studies of neurophysiological systems assume a hybrid approach: the systems are queried with synthetic stimuli like GWN. These responses are then used to train SI models like Wiener-/Volterra kernels. Finally, the trained models are queried using natural stimuli to learn about neuronal behaviour under natural conditions (Newland and Kondoh (1997b,a); Meruelo et al. (2016)).

2.2 Machine learning and Artificial Neural Networks

The history of artificial neural networks (ANN)s as powerful tools for modelling begins with McCulloch and Pitts (1943)’s work on a simplified model of a biological neuron. The McCulloch-Pitts model performed a simple computation: summation of weighted inputs followed by a thresholding nonlinearity. McCulloch and Pitts (1943) showed that this simple computation can be used to perform basic boolean functions including AND/OR/NOT. When published, the McCulloch-Pitts model was accepted as a promising mechanism for implementing machine learning algorithms capable of logical reasoning (McCulloch and Pitts (1943)).

Unfortunately, the McCulloch-Pitts model lacked any mechanism for learning the weights required to function as boolean gates. Rosenblatt (1958)’s perceptron algorithm extended the McCulloch-Pitts model to learn these weights in a Hebbian fashion (Hebbs (1949)). Given a training set of input-output examples, the Hebbian learning rule can be used to iteratively adjust model weights until the model learns the output to a set of inputs. Rosenblatt (1958)’s work introduced the idea that the weights of artificial neurons can be adjusted with supervised learning using training data.

Despite the promise, it was proven that perceptrons were constrained in the number of classification tasks they could be trained on. For example, Minsky and Papert (1969) showed that single-layer perceptrons could not learn the XOR function, despite being able to learn the AND, OR and NOT boolean functions. As Minsky and Papert (1969) proved, this is because single-layer perceptrons are only capable of learning linearly separable patterns, and XOR is not linearly separable. Although stacked perceptrons in multiple layers are capable of learning more complex, non-linearly separable functions like XOR Minsky and Papert (1969); Carpenter and Grossberg (1991), these were difficult to train with the hebbian learning algorithm, causing a decline in interest in artificial neural networks.

In 1986, Rumelhart et al. (1986) specifically addressed the problems of training multi-layered perceptrons, and showed that the backpropagation algorithm offered a much better alternative to the original hebbian learning algorithm for training the weights of a multilayer perceptron (MLP) network. In their groundbreaking work, Rumelhart et al. (1986) showed that the outputs of a MLP can be expressed as a nonlinear function parameterised by network weights. The discrepancy between network outputs and training data outputs can be expressed as a cost function. Rumelhart et al. (1986) experimentally showed that the backpropagation algorithm
minimizes this cost function via gradient descent with respect to the MLP weights. Although
the origin of the backpropagation idea is disputed (Rumelhart et al. (1995)), Rumelhart et al.
(1986) showed that MLPs can learn complex nonlinear patterns when trained using gradient
descent.

Rumelhart et al. (1986)'s work led a resurgence of interest in MLPs. Cybenko (1989) showed that
MLPs are universal function approximators. The increasing popularity saw MLPs being used in
a wide variety of machine learning applications including image recognition, speech recognition
and machine translation Wasserman and Schwartz (1988). Although useful in these applications,
interest in MLPs sooned waned in favour of support vector machines (SVM)s Cortes and Vapnik
(1995), which were more successful than MLPs at the same tasks.

The strength of neural networks for modelling lies in the fact that deeper multi-layer neural
networks can represent more complex, nonlinear, highly varying functions (LeCun et al. (2015);
Curtis and Scheinberg (2017)). Unfortunately, gradient-based optimisation using backpropaga-
tion consistently led to poor solutions (Bengio et al. (2007)). SVMs outperformed single-layer
MLPs for modelling tasks which did not require deep and complex models. For more complex
tasks, deeper networks could not be trained although these could theoretically outperformed
SVMs (LeCun et al. (2015)). For a second time, the interest in neural networks declined.

A solution to the poor convergence of deep networks was proposed by Hinton et al. (2006). In
their groundbreaking work, Hinton et al. (2006) proposed an unsupervised method for training
each successive layer of a deep network before training the entire network on a supervised learning
problem. This method of greedy, per-layer pretraining effectively initialised the weights of the
network to better regions of the feature space, reducing the burden of gradient-based optimisation
on the supervised problem (Hinton et al. (2006); Bengio et al. (2007)). The method proposed by
Hinton et al. (2006) used a Restricted Boltzmann Machine (RBM) for each layer in the network,
and each layer was pretrained to reconstruct its respective input. The optimisation problem was
framed as a minimisation of free energy within the RBM network and the algorithm used to
perform this minimisation was Contrastive Divergence, a popular algorithm for training RBMs
Hinton (2002).

Soon after, Bengio et al. (2007) extended the greedy layer-wise pretraining procedure to networks
of stacked autoencoders. Bengio et al. (2007) empirically showed that the benefit of layer-wise
pretraining comes from initialising the network weights into a favourable part of the search space.
Bengio et al. (2007) also showed that layerwise pretraining can be done using autoencoders in-
stead of RBMs. Like RBMs, autoencoders are generative models which are trained to reconstruct
their input via a semantic feature layer (LeCun et al. (2015)). Unlike RBMs, autoencoders can
be trained with the backpropagation algorithm: which is easier to implement and generally con-
verges to good solutions faster than Contrastive Divergence (Bengio et al. (2007)). Bengio et al.
(2007)'s result led to rising interest in using greedy layerwise pretrained autoencoder networks.
Many variations of the basic autoencoder have been proposed including autoencoders robust to
noise in the input space (Vincent et al. (2008)), autoencoders with enforced sparsity constraints
in their hidden layers (Le (2013)) and autoencoders which form bayesian generative models of
their training data (Kingma and Welling (2013)).

In 2011, Glorot et al. (2011) showed that using rectifying nonlinearities leads to better conver-
gence of neural network models than the sigmoid nonlinearity which was far more popular at
the time. Glorot et al. (2011) empirically showed that the use of rectifying nonlinearities removes the need for unsupervised pretraining. Additionally, Glorot and Bengio (2010) proposed an initialisation procedure for network weights which led to better post-optimisation solutions than randomly initialising network weights. In essence, a combination of good initialisation and rectifying nonlinearities removed the need for unsupervised pretraining.

A parallel growth of big data technologies: solving problems of analysing data collected in the terabyte scale and Graphical Processing Unit (GPU) computing further enabled deep neural networks to excel at supervised learning problems (LeCun et al. (2015)). The availability of large datasets allowed neural network models to learn robust features for accurately solving complex problems. Additionally, the availability of GPU computing made it easier for researchers to rapidly develop and iterate on neural network designs for solving complex challenges (Sanders and Kandrot (2010)). For example, Krizhevsky et al. (2012) demonstrated a deep convolutional neural network trained entirely on GPUs which beat the state of the art on ImageNet (Deng et al. (2009)), a database of 14 million images of 20,000 different categories. Similarly, Taigman et al. (2014)s work demonstrated a nine-layer deep neural network which beat the previous state of the arts accuracy by 27% on a face recognition task. Similar advances have also been demonstrated on tasks including machine translation (Bahdanau et al. (2014)), handwriting recognition (Cireşan et al. (2010)), and multi-task learning (Collobert and Weston (2008)).

2.2.1 Recurrent Neural Networks

One of the results of the work and interest in deep neural networks was the development of Recurrent Neural Networks (RNNs) for modelling complex sequential tasks. Simply described, RNNs have a simple layered structure with one or more recurrent layers (Figure 2.1) (Goodfellow et al. (2016)). The recurrent structure allows the network to be unfolded for multiple steps (Figure 2.1). The capacity for unfolding the network structure for multiple steps allows RNNs the capacity to model sequential data, where the internal recurrent layers model sequential dependence between datapoints (Goodfellow et al. (2016); Gers et al. (1999)). Siegelmann and Sontag (1994) proved that RNNs are Turing-complete ie RNNs are capable of learning and simulating the internal dynamics of any computation. For example, Graves et al. (2014) showed that an RNN model with an internal structure mimicking a Turing machine was capable of learning the program for generating complex visual sequences.

**Algorithm 1:** Simple Recurrent Neural Network (RNN) algorithm. The intermediate layer activations are a nonlinear function of the weighted sum of the current input $x_t$ and hidden activations from the previous timestep $h_{t-1}$. The output is a nonlinear function of the weighted hidden layer activations. $f(\cdot)$ and $g(\cdot)$ are nonlinear activation functions. $W_{zh}, W_{hh}, W_{hy}, b_h, b_y$ are trainable parameters of the model.

1. for $t$ from 1 to $n$ do
2.   $h_t \leftarrow f(W_{zh}x_t + W_{hh}h_{t-1} + b_h)$;
3.   $y_t \leftarrow g(W_{hy}h_t + b_y)$;
4. end

Although there are many ways of introducing recurrence into a neural networks structure, this thesis focuses on recurrence in the hidden layers of the network (Figure 2.1). The simplest RNN of this kind was introduced in the 1980s Elman1990, Werbos1988 and is elaborated in Algorithm
Chapter 2 Literature Review

Figure 2.1: The structure of a simple Recurrent Neural Network (RNN). Inputs $x_t$ are used to compute an intermediate (hidden) representation $h_t$. Per-timestep outputs $y_t$ are computed directly from the hidden representations. The hidden activations for a timestep $t$ is a function of the hidden activations, i.e. recurrence is in the hidden layer (left) and can be unfolded for multiple timesteps (right).

1. In essence the network has two computational elements, one computes the forward pass for each element in an input sequence. The second computes a forward pass from the previous hidden activation. The outputs of the two elements are summed before calculating the current activation of the hidden layer using an appropriate activation function. Finally, the output of the network for the timestep is calculated by a forward pass from the networks current recurrent activations (see Algorithm 1).

The typical method for training RNNs is to unroll the network in time, and use the backpropagation through time (BPTT) algorithm to compute corrections to network weights Bengio et al. (1994); Pascanu et al. (2013). BPTT treats the unfolded RNN as a deep network in time, computing gradients for every timestep with respect to all preceding timesteps Pascanu et al. (2013). Bengio et al. (1994) showed that the BPTT algorithm is highly susceptible to either exploding gradients or vanishing gradients, both leading to poor estimators. The exploding gradients problem is an unbounded increase in the gradient norm with increasing timesteps. Conversely, the vanishing gradients problem is an exponential decrease in the gradient norm with increasing timesteps. Pascanu et al. (2013) explored the exploding/vanishing gradient problem and established analytical bounds for the jacobian of the network where these problems do not occur. Additionally, Pascanu et al. (2013) proposed clipping gradient norms to combat the exploding gradient problem, and a soft constraint strategy to offset vanishing gradients.

The purpose of recurrent connections in the RNN structure is to encompass the history of the input sequence Lipton et al. (2015); Pascanu et al. (2013); Gers et al. (1999). This allows the network to express any history- or memory-based dynamics of temporal sequences. In short, the conditional probability of an output at time $t$: $y_t$ given the history of inputs preceding $y_t$: $(x_0, x_1, ..., x_{t-1}, x_t)$ is denoted by: $P(y_t|x_t, x_{t-1}, ... x_0)$ and is modelled by two feedforward activations, with prior temporal information carried forward in an intermediate representation $h_t$ (see Algorithm 1). Practically, it is difficult to train simple RNNs to capture long-term
dependencies as these result in exploding/vanishing gradients as Pascanu et al. (2013) observed. In addition to the strategies published by Bengio et al. (1994), Hochreiter and Schmidhuber (1997) proposed changing the structure of the RNN to include mechanisms implementing long-term memory: (see Algorithm 2).

**Algorithm 2:** Long Short Term Memory (LSTM) network.

1. for \( t \) from 1 to \( n \) do
2.     \( \text{gate}_{\text{forget}} \leftarrow \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f); \)
3.     \( \text{gate}_{\text{input}} \leftarrow \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i); \)
4.     \( \text{gate}_{\text{output}} \leftarrow \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o); \)
5.     \( C_{\text{intermediate}} \leftarrow \gamma(W_{cx}x_t + W_{ch}h_{t-1} + b_c); \)
6.     \( C_t \leftarrow \text{gate}_{\text{forget}} \cdot C_{t-1} + \text{gate}_{\text{input}} \cdot C_{\text{intermediate}}; \)
7.     \( h_t \leftarrow \text{gate}_{\text{output}} \cdot \gamma(C_t); \)
8.     \( y_t \leftarrow g(W_{hy}h_t + b_y); \)
9. end

Hochreiter and Schmidhuber (1997)’s Long Short Term Memory (LSTM) network separates the intermediate layer representation into a feedforward computation (\( g_{\text{input}} \)) and an internal memory channel governed by \( C_t \). This channel serves to store information within the channel activations \( C_t \). Memory can be cleared/updated using the parameterised gates \( g_{\text{input}} \) or \( g_{\text{forget}} \) (see Algorithm 2). Parameterising the memory in this way allows the network to learn memory-based operations conditioned on the input data. Hochreiter and Schmidhuber (1997) empirically showed that LSTMs outperform simple RNNs in tasks which require long-term memory in a time series. Compared to the simple RNN, the LSTM network has far more trainable parameters. These therefore require longer training time to convergence, with either larger datasets or heavy regularisation to avoid overfitting (LeCun et al. (2015)).

**Algorithm 3:** Gated Recurrent Unit (GRU) network.

1. for \( t \) from 1 to \( n \) do
2.     \( \text{gate}_r \leftarrow \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r); \)
3.     \( \text{gate}_{\text{update}} \leftarrow \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u); \)
4.     \( h_t \leftarrow \gamma(W_{hx}x_t + W_{hh} \cdot (\text{gate}_r \cdot h_{t-1}); \)
5.     \( h_t \leftarrow (1 - \text{gate}_{\text{update}}) \cdot h_{t-1} + \text{gate}_{\text{update}} \cdot \tilde{h}_t; \)
6.     \( y_t \leftarrow g(W_{hy}h_t + b_y); \)
7. end

Cho et al. (2014) proposed an alternative memory-gating mechanism for RNNs. Unlike LSTMs, Cho et al. (2014)’s Gated Recurrent Unit (GRU) networks do not maintain a memory channel state \( C \) and only maintain 2 gating channels instead of 3 (see Algorithm 3). Instead, GRUs store information by controlling how much of the hidden representation activations are fed forward to the next-state hidden activations. Although GRUs still contain more trainable parameters than simple RNNs, these have far fewer parameters than LSTMs, whilst still being able to model long-term dependencies in time series data (Cho et al. (2014); Chung et al. (2014)). Consequently, Chung et al. (2014) empirically showed that GRUs are faster to converge than LSTMs. However, Chung et al. (2014) also pointed out that the choice between LSTMs and GRUs depends heavily on the dataset used.

An important design choice in constructing deep neural networks with more than one hidden layer is the choice of activation function in each hidden layer. In the early days of single-layer
MLPs, sigmoid or tanh activation functions were a common choice (Glorot and Bengio (2010)). However, stacking layers of sigmoid or tanh layers into deep networks proved unsuccessful as the networks wouldn’t converge to good estimators (Glorot and Bengio (2010)). Although it wasn’t articulated at the time, Glorot et al. (2011) later showed that the poor convergence was largely due to the nonlinear activation functions saturating during training, resulting in very small gradients which made training networks impractically slow. Glorot and Bengio (2010) showed that initialising a deep network with appropriately normalised weights resulted in the units within each layer operating in the linear regime of their respective activation function. Additionally Glorot et al. (2011) showed that replacing tanh or sigmoid activations with a nonsaturating rectifying nonlinearity allowed gradients to propagate without saturation, thereby increasing the likelihood of networks converging to good estimators.

These strategies can also be applied to RNNs which are deep in time. Unit saturation and ineffective convergence can be offset by proper weight initialisation and using rectifying nonlinearities (Pascanu et al. (2013)). Unfortunately, Pham et al. (2014) observed that rectifying nonlinearities in place of sigmoid or tanh activations hurts network performance in RNNs. In addition to the strategies used to avoid vanishing/exploding gradients including using memory augmentation to encompass long-term dependencies, using appropriate weight initialisation heuristics is crucial for avoiding poor convergence (Pascanu et al. (2013); Le et al. (2015)).

Training RNNs to converge to good estimators of a timeseries dataset requires choosing an appropriate loss function (LeCun et al. (2015); Bishop (2006)). In short, a loss function is a measure of model performance which allows framing the model optimisation problem as a problem of minimising a scalar loss (Bishop (2006)). The notion of minimising a loss/cost function for deriving optimal estimators is pervasive in machine learning and control theory (Bishop (2006)). For neural networks in particular, the most widely accepted means of minimising a cost function is through gradient descent of the network weights (LeCun et al. (2015); Bengio (2012)).

For continuous-valued regression problems, the most widely used loss function is the mean squared error (MSE) loss, which is expressed as the square of the difference between model prediction and true labels (residual) (Bishop (2006)). From a bayesian perspective, Bishop (2006) and Kingma and Welling (2013) showed that minimising the mean squared error with L2 norms on the network weights is equivalent to maximising the log likelihood of the model outputs with gaussian priors on the model weights. More recently, alternative loss functions have been used for continuous regression problems. Examples include the Huber loss which is quadratic for small residuals and linear for larger residuals, with consequentially lower sensitivity for large outliers in the data (Cavazza and Murino (2016)).

For binary-valued regression problems, the most widely used loss function is the binary cross-entropy (BCE) loss, a measure of the number of logits required to express the discrepancy between model predictions and dataset labels. From a probabilistic perspective, Bishop (2006) and Kingma and Welling (2013) that minimising the binary cross entropy is functionally equivalent to minimising the Kulback-Liebler divergence between the model likelihood and data generating distribution (Bishop (2006)).

Once a loss function is established, optimisation can be performed using gradient descent. In the machine learning community, gradient descent optimisation refers to algorithms which perform cost function optimisation using methods of steepest descent (Bottou (2012)). In this sense, there are two broad categories of gradient descent methods: batch and stochastic. Batch gradient
descent methods have a wealth of literature developed over many decades including accelerated gradient (Ji and Ye (2009)), conjugate gradient (Kershaw (1978)), quasi-Newton (Loke and Barker (1996)) and inexact Newton methods (Dembo et al. (1982)). Unfortunately, taking a batch approach for optimisation requires evaluations on every datapoint in the dataset per iteration, which can be computationally intractable for very large datasets of the kind used to train neural networks (Curtis and Scheinberg (2017); LeCun et al. (2015)). In contrast, stochastic gradient descent methods are less developed, but require a much smaller subset of evaluations per iteration than batch methods (Curtis and Scheinberg (2017); LeCun et al. (2015)). Additionally, there are various methods for reducing the variance of the estimated gradient per iteration including: the use of minibatches Bengio (2012), momentum (Rumelhart et al. (1986)) and adaptive learning rates (Duchi et al. (2011); Zeiler (2012); Kingma and Ba (2014)). In the literature on neural networks, stochastic gradient descent (SGD) methods dominate over batch gradient methods (LeCun et al. (2015)).

One of the fundamental procedures for deriving good estimators on a given dataset is to conduct hyperparameter search (LeCun et al. (2015); Bishop (2006)). In the context of neural networks, the term parameters refers to the set of weights in which knowledge about the training data is encoded. The term hyperparameters refers to the set of variables which define the structure of the neural network, including number of hidden layers, number of units in each hidden layer, activation functions, etc. A neural networks parameters are found by optimisation, typically using stochastic gradient descent (LeCun et al. (2015)). Optimal hyperparameters for a problem are typically found using a variety of search algorithms (Bergstra and Bengio (2012); Nareyek (2003); Czogiel et al. (2006); Snoek et al. (2012)).

RNNs, LSTMs and GRUs are all generic neural networks designed for timeseries modelling. Consequently, these networks are not designed for modelling the underlying dynamics of any single dataset. Wolpert and Macready (1997) explained that simply using a generic neural network structure for modelling an arbitrary dataset will yield a suboptimal solution with high probability. As a result, there is a large body of work aimed at finding optimal hyperparameters for modelling a given dataset using typical machine learning models (Bergstra and Bengio (2012); Nareyek (2003); Czogiel et al. (2006); Snoek et al. (2012)).

Unfortunately, the black box nature of neural networks and other machine learning models means there is no easy method for performing a guided search for the best hyperparameters. The most widely used strategy is to use a combination of grid search and manual search (Bergstra and Bengio (2012)). Manual search is simply tuning hyperparameters manually, relying on experience of similar datasets to guide a choice of hyperparameters (LeCun et al. (2015)). Grid search discretises the space of hyperparameters and uniformly chooses candidate hyperparameters to evaluate (Bergstra and Bengio (2012)). Bergstra and Bengio (2012) showed that using a random policy for choosing hyperparameters to evaluate outperforms grid search in that the random policy is more likely to find higher performance hyperparameters using fewer evaluations. Finally, Snoek et al. (2012) used a gaussian process to derive a probability model over the space of hyperparameters. Snoek et al. (2012) used this probability model to guide the hyperparameter selection policy and showed that this method outperforms random search, grid search and manual search for complex models on large datasets, at the expense of significant computational overhead.
Every iteration of hyperparameter search requires an evaluation of model performance on the input dataset. The typical method for training and evaluating a machine learning model is to split the dataset into independent training and test data (LeCun et al. (2015); Bishop (2006)). Here, a model is trained on the larger training set its generalisation performance is evaluated on the independent test set. Evaluating the quality of various hyperparameter combinations on the test set can bias the estimate of generalisation error (Bishop (2006)). This is because improving model performance on the test set through hyperparameter tuning is equivalent to creating an ad-hoc model for improved test set performance, with no guarantees on improved generalisation (Bishop (2006)).

Hyperparameter evaluation is typically performed on a validation set which is independent of both the training and test set (LeCun et al. (2015); Bishop (2006)). A typical method is to randomly sample a predefined percentage of the dataset for training and validation respectively (LeCun et al. (2015); Bishop (2006)). Models with various hyperparameter combinations are trained on the training set and evaluated on the validation set. Another popular method is to split the data into $k$ buckets of equal size. Of these, $k - 1$ buckets are used to train a model, and 1 bucket is used for validation. For any given hyperparameter combination, this procedure is repeated $k$ times, each with a different bucket for validation. This method of $k$-fold cross-validation is widely used in the literature, and yields lower-variance measures of hyperparameter quality (Bengio and Grandvalet (2004)). The $k$-fold method requires $k$ times more evaluations than the holdout method for evaluating a single hyperparameter combination. However, this can be mitigated by parallelising the evaluations (Bergstra and Bengio (2012)).
Chapter 3

Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron

3.1 Introduction

This chapter explores the use of Recurrent Neural Networks (RNNs) for the system identification of the FETi motor neuron which participates in the reflex control of the locust hind leg. System identification is a body of work aimed at understanding the unknown characteristics of systems through modelling. In general, the modelling procedure begins with collecting the inputs and outputs of a black-box system. Statistical tools are then used to identify a functional relationship between stimulus and response from the system. In this way, system identification uses data-driven methods to derive statistical estimates of the functional characteristics of a system (Nelles (2013); Meruelo et al. (2016); Marmarelis (2004)).

The idea of identifying important properties of a system from data has seen wide adoption in science and technology. System identification is used to design robots and implement high-fidelity control systems (Kovac et al. (2008); Clark et al. (2001); Saranli et al. (2001)), and is also behind developing new methods in healthcare for diagnosis (Marmarelis (2004)) and rehabilitation (Reinkensmeyer et al. (2004)). With rapid advances in technology across domains, and a growing research interest in systems of ever greater complexity, system identification is seeing increased adoption across many fields in science and engineering (Garg et al. (2017)).

For neuroscience in particular, system identification has long been an important set of tools for answering questions about neuronal systems (Marmarelis (2004)). Ideally, the modelling tools used for identifying the characteristics of neurons must be accurate enough to predict system responses over a potentially wide range of operation. Additionally, the models must be parsimonious whilst also being open to physiological interpretation (Marmarelis (2004)).
practice, the most pragmatic modelling techniques for a task do not match up to all of these requirements (Wolpert and Macready (1997)), and tradeoffs must be considered.

Based on the particular subject of investigation, neuroscientists use a wide range of models to understand the behaviour of various parts of the nervous system (Nelles (2013)). Of these, linear and nonlinear models based on the Wiener/Volterra series expansion have been prolific (Marmarelis (2004); Meruelo et al. (2016); Newland and Kondoh (1997a); Dewhirst et al. (2013)). While Wiener/Volterra models are powerful function estimators (Marmarelis (2004)) and provide easily interpretable descriptions of the linear characteristics of a system (Newland and Kondoh (1997a)), they are particularly susceptible to estimation errors where there is a lot of background noise in the data (Dewhirst et al. (2013)). If model estimation errors are high, we get a poor description of the underlying system behaviour. Additionally, Wiener-like identification models do not always converge to a good solution for very large stimulus-response datasets (Tötterman and Toivonen (2009); Palm and Poggio (1977a)). These characteristics are especially troublesome in modern neuroscience, where there is an increased adoption of very high resolution techniques (Devous Sr et al. (1998); Zhu et al. (2017)) for experiments. Higher resolution experiments can create larger datasets, and the data gathered can be noisier (Elze (2010)). Although successfully used in many neurological investigations (Marmarelis (2004); Nelles (2013); Newland and Kondoh (1997a); Dewhirst et al. (2013); Meruelo et al. (2016)), Wiener/Volterra models have their limitations.

Recently, there has been a surge of research interest in using Artificial Neural Networks (ANNs) for system identification. Although researchers used ANNs on equal footing with other models like Support Vector Machines (SVMs) for a range of system identification tasks in the past (LeCun et al. (2015)), the wide availability of large datasets and advances in massively parallel computing have recently enabled ANNs to be used on a wider range of tasks than was previously possible (LeCun et al. (2015)). ANNs are empirically shown to be superior to most other model types for very large datasets (LeCun et al. (2015)). Additionally, ANNs are capable of robust modelling with noisy data (Sukhbaatar et al. (2014)). There are also a wide range of optimisation algorithms using techniques like stochastic gradient descent (SGD) to train ANNs to converge to good solutions (Curtis and Scheinberg (2017)). The combination of these modelling properties means that ANNs have seen wide adoption with state-of-the-art results in academic research and industrial applications including Natural Language Processing (Goldberg and Levy (2014); Collobert and Weston (2008); Manning et al. (2014)), Speech Recognition (Hinton et al. (2012); Amodei et al. (2016)), Machine Translation (Bahdanau et al. (2014)), Natural Language Generation (Van Den Oord et al. (2016)), and Image Processing (Szegedy et al. (2017); Targ et al. (2016); Howard et al. (2017)). Going into a future of very large and noisy datasets, models using ANNs are an attractive option for system identification in neuroscience.

The aims of this chapter are twofold. The first is to show that a subset of ANNs called Recurrent Neural Networks (RNNs) are useful for the neuronal system identification task. Here, RNNs are used to model intracellular responses of the Fast Extensor Tibiae (FETi) neuron in the hind leg of the *Schistocerca gregaria*. Next, the RNN models are used to infer the functional characteristics of the FETi neuron. These inferred characteristics are then compared to what is already known about FETi, whether the models corroborate or contradict the well-established body of knowledge.
In the first section, the methods used to collect stimulus-response samples from FETi neurons across different locusts are detailed. Here, the methods used to explore and preprocess the data for modelling is described. Additionally, the details of a list of modelling experiments aimed at training and validating RNN models are presented. The next section describes the results of these experiments, including RNN training and validation errors, validity of the trained RNNs, and their physiological interpretation. Finally, the chapter concludes with a discussion of the results in establishing the efficacy of using RNNs for modelling the FETi neuron, and whether these would be useful tools for system identification in the wider neuroscience community.

3.2 Methods

This section describes a series of experiments where Recurrent Neural Network (RNN) models are trained on intracellular responses of the Fast Extensor Tibiae (FETi) neuron. First, a procedure for exploring the data for anomalies and helpful features is described. Additionally, the procedure for preprocessing the FETi data to make it more amenable to the modelling task is detailed.

Next, the process for training RNNs on the FETi data is described. In addition to RNNs, memory-augmented variants of the RNN are trained: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The quality of these models are assessed using standard Machine Learning metrics. These metrics are then compared with previous work using Time-Delay Neural Networks and Linear-Nonlinear-Linear models (Meruelo et al. (2016); Newland and Kondoh (1997a)).

3.2.1 Collecting the Data

The input-output data from the FETi neuron was collected using the following procedure outlined by Newland and Kondoh (1997a):

First, locusts were placed ventral-side up in modelling clay. The modelling clay held the locust in place, preventing it from moving. A hind leg was rotated through 90° and fixed anterior face-up. The femur was set at a 60° angle from the abdomen. Additionally, the tibia was fixed at a 60° angle from the femur.

The locusts metathoracic and mesothoracic ganglia were exposed by dissecting away a section of the cuticle in its ventral thorax. The ganglia were fixed to a silver platform using fine pins. This platform also served as the ground reference to electrodes later inserted into the locust. The thorax was continuously bathed in locust saline. The saline solution was maintained at room temperature 23°C.

The FETi motor neuron was then identified. Identifying this neuron is easy because of its relatively large soma (Burrows (1996)). A glass microelectrode filled with 2M potassium acetate was driven through the into this soma. For all the experiments, the resistances on the electrodes were between 50-80 MΩ. The excitatory post-synaptic potentials measured in the FETis soma were amplified and converted into a digital format using an Analog-Digital (AD) converter on
a data acquisition board (USB 2527 data acquisition card, Measure Computing, Norton, MA, USA).

Next, the apodeme of the locust Femoro-Chordotonal Organ (FeCO) was exposed by dissecting a small window into the cuticle of the anterior distal femur. The apodeme as cut near its insertion point on the tibia. Separating the FeCO apodeme from the tibia opens the tibia-FeCO-FETi-flexor-tibia loop. The belief is that opening the loop removes feedback responses which may confuse any analysis of the data. The detached apodeme was then grasped by forceps. These forceps were attached to a shaker. To summarise, the complete experimental setup is a shaker which provides input stimulus to the locust FeCO via its apodeme; and a microelectrode which collects the corresponding FETi post-synaptic responses from its soma.

The shaker was driven by 50 Hz bandlimited Gaussian White Noise (GWN) signals. GWN simultaneously excites all frequencies within its bandwidth (Eggermont (1993)). The belief is that GWN signals can stimulate the neuron within its entire operating range, in the short lifespan of the experiment. In this way GWN allows yields richer stimulus-response data for a modelling task (Marmarelis (2004)).

After the application of the GWN inputs, a 5 Hz sinusoidal stimulus was applied to the shaker. This 5 Hz stimulus is functionally relevant as it approximates the walking cycle of locusts (Blackburn et al. (2010)).

All of the stimuli were first generated in MATLAB and passed through Digital-Analog (DA) converters on the data acquisition board. The analogue signals were then amplified before driving the shaker.

A significant point of note is that the input applied to the shaker is measured by the linear displacement in the shaker-apodeme setup. However, for the modelling task, the input needs to be expressed in terms of femoro-tibial angular displacement. Dewhirst et al. (2013) proposed a method for measuring the relationship between femoro-tibial angle and apodeme position. This method was used to calibrate the shaker displacement and express it in terms of femoro-tibial angle.

For each of the 50 Hz GWN and 5 Hz stimuli, FETi responses were collected through the microelectrode imposed on its soma. These responses were amplified and digitised at a sampling frequency of 10000 Hz. Digital versions of the stimulus and corresponding responses were temporally aligned and stored for subsequent analysis.

Before their use in modelling experiments, both the stimulus and response data were decimated to 500 Hz. The decimation was done after passing the original signal through an anti-aliased filter with a cutoff frequency at 250 Hz. A 10000 Hz sampling rate is too large for signals that have 50 Hz bandwidth. It was unlikely that the frequency components of the FETi responses extended 3-4 orders of magnitude beyond the range of the stimulus frequency. Having this buffer region between the frequency bandwidth of the informative signal, and the 10000 Hz sampling frequency simply invites high-frequency noise which could confuse the analysis. Furthermore, the original signal contained 20 times as many datapoints per second than its 500 Hz decimation. This implies 20 times as many computational operations per iteration for training models. Downsampling was therefore used to reduce experimental noise and reduce training time. Once the signals were downsampled to 500 Hz, a high-pass filter was applied to eliminate slow time-varying drift.
The recording procedure described here was used on five different locusts. For each locust, the 50 Hz GWN signals were applied thrice consecutively. Each of these three recordings was synchronised and averaged to reduce the variability of responses from that locust. Because the same GWN signal was used across all animals, the responses of all five animals were similarly aligned and averaged to give an average-response recording. Responses to the 5 Hz stimulus were also averaged. This procedure leaves us with a total of twelve stimulus-response recordings: five 50 Hz GWN, five 5 Hz, one 50 Hz average-response and one 5 Hz average-response.

It should be noted that Dewhirst et al. (2013) used the procedure outlined here to derive the twelve stimulus-response recordings. This data was repurposed for the work detailed in this chapter.

### 3.2.2 Analysing the Data

Before beginning any modelling work using the locust FETi input-output data, the data was explored for any helpful features. Data exploration is an important preliminary to any machine learning or system identification workflow. An exploratory analysis can reveal corruptions with the data like missing data points or large anomalies. Neurological time-series data are particularly susceptible to such corruption (Newland and Kondoh (1997a)). Missing data in an input-output time series can distort important temporal patterns which define the neuronal processes that created the data. Additionally, experimental errors such as slipped electrodes, misconnections,
large voltage sources and unintentional noise can skew the data in a way that biases the data
to the source of the anomalies. Induced biases like these can corrupt the data and hinder any modelling work. This step intended to discover any anomalies which would hinder any downstream modelling work.

Besides discovering anomalies and missing values, data exploration can help improve the quality of models. If helpful patterns can be spotted in a dataset from simple visual inspection, these patterns may be used as features in any modelling work. These patterns can be used to inform the choice of algorithm, hyperparameters and train-validation data split to maximise the efficiency of the trained models. In this way, data exploration can help discover patterns which may either help or hinder modelling work at a crucial early stage.

To begin, the input-output time-series data was plotted. Figure 3.2a illustrates the type of data examined. This particular illustration depicts the input-output time-series for Locust 1. On first examination, none of the recordings contained any missing data points. If they did, these would appear on frequency plots of the input and output signals. Large swaths of missing data points would be visible as rapid drops to 0 in the time-series. Additionally, these shifts would easily identified as high-frequency artefacts on a frequency plot.

An important observation from the time-series data is that negative swings in the input correlate with positive responses in the output (Figure 3.2 b). On the other hand, a positive swing in the input doesn't elicit a proportionally negative response in the output. This is expected because a negative input corresponds to tibial flexion, which should elicit a strong response from the extensor muscle to maintain a reflex response in the tibia (Dewhirst et al. (2013)). On the other hand, a positive response (extension) relaxes the extensor and its response isn't as pronounced.

A second observation from the time-series is that there appears to be a delay between negative input swings and corresponding positive output swings. In all of the 50 Hz Gaussian White Noise (GWN) recordings, this delay was observed to be between 10-20 milliseconds. A delay between input and output is expected since the input stimulus into the FeCO has to travel through the neuronal circuit before eliciting a response at the FETi motor neuron. Later, trained RNN models are also tested check if they model the signal delay characteristics of FETi in addition to its other functional properties.

Next, the frequency characteristics of the time-series data are plotted. (Figure 3.2c) shows an example of the frequency characteristics of Locust 1. For the input, most of the frequency components appear to be contained within the 0-50 Hz band. This is expected because the input GWN was limited to a 50 Hz bandwidth. Interestingly, the frequency of the output also seems limited to a 50 Hz bandwidth. This suggests that the underlying neuronal circuit may not modify the input frequency. However, the frequency characteristics of the output for 5 Hz inputs (Figure 3.2d) show strong frequency components at 5 Hz, 10 Hz, 15 Hz, and 20 Hz. This observation is consistent across all the locusts. This indicates some nonlinearity in the FETi's dynamics, which is explored later.
Figure 3.2: (a) Time-varying response from the FETi neuron to changes in femoro-tibial angle stimulated with 50 Hz Gaussian White Noise (GWN). (b) FETi response to changes in femoro-tibial angle when stimulated with a 5 Hz sinusoid. (c) Frequency spectrum of FETi response to 50 Hz bandlimited GWN. (d) Frequency content of FETi response to 5 Hz sinusoidal input. In (a) and (b), negative swings in femoro-tibial angle correspond to flexion, which elicits a strong response from the neuron. The output frequency spectrum (c) is roughly consistent with the 50 Hz bandwidth at the input. Output spectra for a 5 Hz sinusoidal input shows sharp, but decaying spectral content at multiples of 5 Hz, along with some background noise.

3.2.3 Preprocessing the Data

An important point of consideration when training RNN models, is the quantity of data used for training. If the learning algorithm is exposed to a large number of datapoints, there is greater likelihood of it learning the important underlying patterns and ignoring any surrounding noise. On the other hand, if exposed to a small number of datapoints there is a danger of that the model will ignore the underlying structure and overfit to these datapoints. For a given modelling task, there is no exact measure for the ideal number of datapoints required to train a model. The generally accepted heuristic is more data for higher quality models (LeCun et al. (2015)).

For each locust, the data extracted from its FETi neuron is a stimulus-response timeseries lasting approximately 30 seconds. At 500 Hz, this means each time series has 15000 data points. The aim of using RNNs is to model the underlying hidden patterns which cause the temporal characteristics of this data. Although a model can be trained using the entire sequence 15000 datapoints, it must be noted that an RNN sees a temporal sequence as a single datapoint.
In this work, the input-output time series data were split into multiple smaller time series using a window (Figure 3.3), to emulate a dataset with multiple datapoints for the RNN models.

Intuitively, a large time-window encompasses long-ranging temporal effects. These may include memory-like neuronal behaviours where a stimulus many milliseconds ago affects how the neuron processes its current input. A shorter time-window will ignore these, as cause and effect may be separated into separate temporal datapoints. Windowing effectively splits the time-series into smaller chunks. From a modelling perspective, each chunk is an independent datapoint containing crucial temporal information. For a fixed-length time-series, longer windows and larger chunks mean fewer datapoints to learn from. Conversely, smaller windows mean more chunks but these can miss crucial long-range phenomena.

Here, time-windows 500ms long were used to safely contain any expected latencies between stimulus and FETi response (Newland and Kondoh (1997a)). At a 500 Hz frequency, each chunk encompassed a sequence of 250 consecutive timepoints. If chunks were chosen so that there were no overlapping timepoints between them, there would be a total of 60 chunks for the model to learn from. By windowing in a way that successive chunks had an overlap of 240 timepoints between them, the total number of chunks per recording was 1476, a 24-fold increase in data over the non-overlapping case. The window size and window overlap parameters were set as training hyperparameters which were chosen using a random search procedure outlined in Section 3.2.4.

3.2.4 Experiment 1: Cross-Validation and Training the Models

In machine learning literature, a major shortcoming of the straightforward RNN is that it lacks internal memory mechanisms required to remember and model long-term temporal dependencies. At the pre-modelling stage, it was unclear whether FETi neurons are truly affected by such long-term effects (Gers et al. (1999)). By reasoning that memory-augmented RNN models like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) would outperform the simple RNN if FETi had any long-term effects, RNNs, LSTMs and GRUs were trained separately on each of the five FETi datasets.

Before training the models, the data was split into three independent parts: a training set, a validation set and a test set. The training set was the largest of the three, and contained the data used to train the models. The purpose of the validation set was to fine-tune various hyperparameters of the models. Finally, the test set was used to evaluate the quality of fully trained models. The data was split into proportions of 60:10:30 for training:validation:test.

There is an important reason for using both a validation and a test set in addition to a training set. Where a training set is used to tune the parameters of the model, a test set is used to analyse how well these parameters encode the underlying patterns from the training set. By definition, a test set cannot be used for training and must be entirely independent. Often, the model needs to be tuned to assess which combination of hyperparameters (number of hidden units, regularisation constants, number of hidden layers, gradient optimisation algorithms, etc.) works best at modelling the data. In this situation, it is bad practice to use the test set for tuning hyperparameters, as this would implicitly fit the model onto the test set. Tuning model hyperparameters with a test set defeats the purpose of it remaining and independent and unbiased indicator of model quality.
Chapter 3 Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron

Figure 3.3: Illustration of chunking FETi data with time windows: (a) 2s of FETi data (1000 timepoints). (b), (c) Two chunks of the FETi data derived from windows of 500 timepoints each. Each chunk encompasses 1s of data.

As with the test set, the validation set wasn’t used for training. By choosing the set of model hyperparameters which perform the best, the final models were indirectly fit on the validation set. Therefore, if the models demonstrated good results on training and validation data, but poor performance on unseen test data, it would be a good indication of the models ’memorising’ the data they’re trained on instead of modelling the underlying functional features of FETi and generalising to unseen test data.

The choice of a models hyperparameters is a challenging problem. RNNs and their memory-augmented counterparts, LSTMs and GRUs are designed to be general enough to model any time-series in any domain (Gers et al. (1999)). Their model structure does not embed any prior information about the data they’re trained on, apart from assuming temporal structure between successive timepoints. As such, the space of possible models is large and spanned by the model’s hyperparameters. Important hyperparameters include the number hidden layers, the number of units in the hidden layer, how the models weights are initialised, the activation function of units in each hidden layer and many more. The correct choice of hyperparameters is important to training well-performing models on a specialised domain like the FETi problem. In hyperparameter space, there are generally a small subset of the models which perform well on a given domain, and this subset is not known a-priori.
There are general guidelines for choosing well-performing hyperparameters (Bengio (2012)). It should be noted that these guidelines are sets of heuristics derived empirically rather than any kind of formal analytical process. There are also several search algorithms which help search the model space for high-quality hyperparameters. By scoring models on their performance on a validation set, these search algorithms narrow down on the best performing subset of hyperparameter space.

In this work, the random search algorithm was used to narrow down a choice of hyperparameters. Bergstra and Bengio found that randomly searching through the hyperparameter space is much more efficient at finding well-performing subsets of the space than the alternative grid search algorithm (Bergstra and Bengio (2012)). Unlike grid search, random search does not exhaustively probe the search space and stumble upon good hyperparameters randomly in far fewer iterations. Meruelo used a much more involved method using a combination of Particle Swarm Optimisation and Evolutionary Algorithms for searching the hyperparameter space for Time-Delay Neural Network (TDNN) models (Meruelo et al. (2016)) for the FETi modelling task. In this work, reasonable ranges for hyperparameter values were found using manual search. Thereupon the random search algorithm was used to evaluate various hyperparameter combinations sampled with a uniform probability distribution. The search procedure was halted after 2000 iterations, and the best performing combinations of hyperparameter values were chosen for further modelling.

Appendix 1 displays the hyperparameter values used for the RNN, LSTM and GRU models trained in this experiment. For each of these, the same hyperparameters were used across all five FETi datasets. For all models, a single layer was used. Additionally, the Glorot procedure (Glorot and Bengio (2010)) was used to initialise the weights of the networks. For the RNNs, the tanh activation function was used. Additionally, tanh-activated units were used in the hidden layers of the LSTM and GRU models, but sigmoid activations were used on the memory-gate units of these models following the procedure outlined in their respective literature (Gers et al. (1999); Chung et al. (2015)). Finally, the full networks were trained using the Adam algorithm (Kingma and Ba (2014)) for stochastic gradient descent, with minibatches of 32 randomly chosen data chunks.

The RNN, LSTM and GRU models were trained 20 times on each of the five datasets. Each time, the models were initialised using a different random seed to ensure that the final models converged to different sets of parameters. This was done to statistically compare how well the RNNs, LSTMs, and GRUs converged to a final trained state from randomly chosen initial parameters.

\[
\frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t y_t^2}
\]  

(3.1)

The test-set performances of trained models are expressed using the Normalised Mean-Squared Error (NMSE) metric. Described in Equation 3.1, the NMSE measures the square of the deviation between the measured FETi output and the model’s prediction for timestep \( t \), where \( y_t \) is the FETi output, and \( \hat{y}_t \) is the model-predicted output. This quantity is then normalised by the sum of the squares of the FETi output. Higher fidelity models score lower on the NMSE, because
of lower deviation between FETi output and model prediction. On the other hand, higher NMSE scores indicate poor model fit. NMSE scores are typically reported as a percentage.

### 3.2.5 Experiment 2: Modelling Randomised Data

A predominant trend in Machine Learning and modelling literature is to use a metric like NMSE to report on the quality of a model or learning algorithm. On the whole, this is a sensible approach because these metrics directly measure predictive performance. But if tuned carelessly, learning algorithms like the RNN can become biased toward producing lower NMSEs, feigning performance without properly learning the underlying dynamics of the data. A common example is overfitting, where the algorithm learns a perfect input-output map from its training set but fails to extrapolate to unseen data (Curtis and Scheinberg (2017)). When a model overfits, it effectively memorises the brute-force, per-point, input-output mapping in the training set and doesn’t necessarily learn the underlying phenomena mapping the input to the output. Simply reporting the NMSE on a training set won’t uncover this pathological overfitting.

The previous set of experiments attempted to mitigate the overfitting problem by observing model performance on test data. The models were not exposed to the test data at any point during training, implying that higher NMSE scores on the test data is an indication of the models generalising on and not memorising the training data. However, if the test set is very similar to the training set, there is a danger that a model overfitting on the training set will also be biased on the test set. The danger of training-test similarity is particularly valid in this case because both were created from the same input-output recording. Thus, the following set of experiments were designed to provide an additional measure of confidence to ensure that the trained models weren’t overfit or biased.

First, random Gaussian White Noise (GWN) data was produced with the same frequency characteristics as the original input recordings used to stimulate the FETi. RNN models were then trained on this randomly generated input data. However, the original FETi responses were used as outputs. By randomly generating input data, the underlying dynamics which produce FETi responses are obscured. In repeating the modelling experiments on this data, a low training error is expected compared with a high test-set error, since the models have no underlying patterns to learn and generalise from, leaving them the only option of overfitting to the training data. 70% of this obscured data was used for training and 30% as a test set. There was no need for a validation set, as the same hyperparameters were used as before. 20 different RNNs were trained on obscured data for each of the 5 locusts, each time initialising the RNN weights differently and with a new random seed. The spread of test-set NMSE scores for these 20 models are reported later. If the RNN models are indeed prone to overfitting, high NMSE scores and low performance is expected on the test-set.

By repeatedly training RNNs on obscured data, statistical baselines are established for models which overfit and memorise the data. If the NMSE scores of RNNs trained on the original data are statistically significantly different from these baselines, the null hypothesis of models overfitting to FETi data can be rejected. The alternative hypothesis is that the models have truly learned the underlying FETi dynamics.
3.2.6 Experiment 3: Physiological Interpretation of the Models

After determining the quality of the RNN models on FETi data in the previous two experiments, the physiological interpretation of these models are explored. In the previous experiment, the models were tested against statistical baselines for overfitting. In this experiment, this premise is further tested by querying the trained models on 5 Hz stimuli, and comparing model responses to neuronal responses to the same stimuli. The physiological implications of the model results on the 5 Hz data are explored.

First, each of the trained models were queried for their responses to a 5 Hz sinusoidal input. The corresponding outputs were compared with outputs from the FETi neuron obtained in the data collection stage. Normalised Mean-Square Error (NMSE) scores are reported as a quantitative measure of the discrepancy between FETi output and model prediction for the 5 Hz stimulus. Lower NMSE scores imply smaller discrepancies between the model prediction and the FETi output. Lower NMSEs additionally imply that the models can extrapolate FETis response dynamics to a sinusoidal 5 Hz stimulus despite being trained 5 Hz GWN stimulus.

Next, the frequency content of both the model response and the FETi response to the 5 Hz input signal are compared. In the data exploration stage it was noted that the FETi response to 5 Hz stimuli has frequency harmonics at 10, 15 and 20 Hz. Here, the frequency content of the model output was examined for the presence of these harmonics as an indication of model fidelity to the data dynamics.

Finally, the delay between the peaks of the stimulus to peaks of the response are examined. The data exploration stage noted a 10-20 millisecond delay between input stimulus and output response from the neurons. The aim at this stage is to quantify this delay characteristic by querying the RNN models. First, each model was queried with sinusoidal inputs ranging from 1 Hz to 50 Hz. Cross-correlations were then used between the input and the corresponding model output to derive the phase difference between the two waveforms. Finally, the phase differences were translated into a peak-to-peak time delay, in milliseconds, between input and output.

3.3 Results

So far, a set of experiments are described, aimed at evaluating if Recurrent Neural Networks (RNNs) are suitable for the FETi modelling task. In this section, the results of these experiments are provided. First, the results of tuning hyperparameters using cross-validation are described. Next, the ability of RNNs to generalise on unseen test data is shown. Additionally, the generalisation performance is compared with previous work on Linear-Nonlinear-Linear (LNL) models and Time-Delay Neural Networks (TDNNs) for the FETi modelling task. Next, proof is established that RNNs, being the best-performing models on the FETi task do not display symptoms of bias/overfitting. Finally, the fidelity of the trained RNN models to FETi dynamics is established by examining their responses to 5 Hz input stimuli and drawing parallels with the biological functionality of the FETi neuron.
3.3.1 Experiment 1: Training the Models

Figure 3.4 (a) shows the cross-validation performance of randomly chosen sets of hyperparameters for each of the RNN, LSTM and GRU models, on each of the 5 FETi datasets. The performance of each appears uniform across each dataset, with Normalised Mean-Squared Errors (NMSE) between 99% to 101%, apart from the RNN performance on Animal 5, which has the largest NMSE of 108%.

In contrast, Figure 3.4 (b) shows the cross-validation performance of hyperparameters finalised by the random search algorithm. Here, much lower NMSEs can be observed, ranging from 16% to 45%. The NMSEs are not uniform across animals. However, the performance of the RNN, LSTM and GRU models appear consistent within each dataset.

Figures 3.5 (a), (b) and (c) show the spread of test-set errors for each of the models across all 5 animals. This spread is the result of training each model 20 times on every animal. Here, the variation in the mean of errors across the animals appears consistent for all the models, with Locusts 1 and 4 showing the highest errors for each. An interesting observation is that the RNNs display the most variation in test errors for Locusts 1 and 4.

Table 3.2 lists the standard deviation of the test-set errors for each of the models across the 5 animals. Here, the memory-augmented LSTMs and GRUs generally have a lower spread of errors across all the locusts tested. This result is likely because the memory-augmented models converge to very similar parameters, despite being initialised with different random seeds. On the other hand, the mean of the errors are lowest for the RNN (Table 3.1), indicating that RNNs are the best performing of the three model types.

Table 3.3 Lists the test-set NMSEs for the RNN, LSTM and GRU models trained in this experiment, alongside the NMSEs for LNL and TDNN models. Here, the RNNs are generally the best performing model type, with the lowest errors apart from Locusts 3 and 4. In their work comparing TDNN models to LNL models, Meruelo et al. (2016) reported that the TDNN performance is statistically significantly better than the earlier LNL model. Meruelo did this
Chapter 3 Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron

Figure 3.5: The spread of Mean Squared Errors for each model, trained on each of the 5 locusts. (a) Recurrent Neural Network; (b) Long Short-Term Memory and (c) Gated Recurrent Unit models. Locusts 1 and 4 show the largest errors for all models.

<table>
<thead>
<tr>
<th>Locust 1</th>
<th>Locust 2</th>
<th>Locust 3</th>
<th>Locust 4</th>
<th>Locust 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>27.43</td>
<td>16.60</td>
<td>17.01</td>
<td>40.17</td>
</tr>
<tr>
<td>LSTM</td>
<td>30.27</td>
<td>18.58</td>
<td>20.56</td>
<td>46.40</td>
</tr>
<tr>
<td>GRU</td>
<td>30.75</td>
<td>18.85</td>
<td>20.65</td>
<td>46.49</td>
</tr>
</tbody>
</table>

Table 3.1: Mean NMSE of 20 trials for each model type across all locusts. Lowest errors are highlighted in bold.

using a Mann-Whitney $U$ test for statistical significance. To maintain consistency in this work, the Mann-Whitney test is also used to assess if RNNs, LSTMs and GRUs are significantly better performing than the TDNN models.

Table 3.4 Depicts the results of a Mann Whitney U test to compare the significance between RNN, LSTM and GRU with the test-set performance of all the other tools used to model FETi. Only the RNN and TDNN models are significantly better than the LNL models ($p < 0.05$). None of the other models show any significant difference from each other.
Figure 3.6: Plot of FETi response vs. model response for the same input on 1s of unseen test data. The model shows a reasonably close fit to the FETi response indicating that it has learned the underlying neuronal dynamics.

Figure 3.7: Spread of test-set errors for RNNs trained on randomised GWN data.
Table 3.2: Standard deviation of NMSE for 20 trials across all locusts. Lower standard deviations are highlighted in bold.

<table>
<thead>
<tr>
<th>Locust</th>
<th>Locust 2</th>
<th>Locust 3</th>
<th>Locust 4</th>
<th>Locust 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>1.41</td>
<td>0.36</td>
<td>0.43</td>
<td>2.20</td>
</tr>
<tr>
<td>LSTM</td>
<td><strong>0.41</strong></td>
<td><strong>0.21</strong></td>
<td><strong>0.39</strong></td>
<td><strong>0.80</strong></td>
</tr>
<tr>
<td>GRU</td>
<td>0.47</td>
<td>0.42</td>
<td>0.53</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 3.3: Comparison of NMSE scores for RNN, LSTM, GRU, TDNN and LNL models across all five locusts. Lowest errors are highlighted in bold.

<table>
<thead>
<tr>
<th>Locust</th>
<th>Locust 2</th>
<th>Locust 3</th>
<th>Locust 4</th>
<th>Locust 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>27.43</td>
<td><strong>16.60</strong></td>
<td>17.01</td>
<td>40.17</td>
</tr>
<tr>
<td>LSTM</td>
<td>30.27</td>
<td>18.58</td>
<td>20.56</td>
<td>46.40</td>
</tr>
<tr>
<td>GRU</td>
<td>30.75</td>
<td>18.85</td>
<td>20.65</td>
<td>46.49</td>
</tr>
<tr>
<td>TDNN</td>
<td>29.40</td>
<td>18.10</td>
<td><strong>15.50</strong></td>
<td><strong>33.70</strong></td>
</tr>
<tr>
<td>LNL</td>
<td>32.10</td>
<td>27.50</td>
<td>30.70</td>
<td>41.20</td>
</tr>
</tbody>
</table>

Table 3.4: Distribution of p-values for Mann Whitney U tests comparing the MSE scores of models. P-values less than 0.05 are shown in bold.

<table>
<thead>
<tr>
<th>RNN</th>
<th>LSTM</th>
<th>GRU</th>
<th>TDNN</th>
<th>LNL</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0.14</td>
<td>0.15</td>
<td>0.5</td>
<td>0.04</td>
</tr>
<tr>
<td>LSTM</td>
<td>-</td>
<td>0.41</td>
<td>0.89</td>
<td>0.15</td>
</tr>
<tr>
<td>GRU</td>
<td>-</td>
<td>-</td>
<td>0.89</td>
<td>0.20</td>
</tr>
<tr>
<td>TDNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>0.03</strong></td>
</tr>
<tr>
<td>LNL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

3.3.2 Experiment 2: Modelling Randomised Data

The results of experiment one showed that RNNs performed significantly better than LNL models, and showed similar results to TDNN models for the FETi modelling task. Low NMSE scores indicating high test-set performance gives a degree of confidence in the reliability of RNNs in modelling the underlying FETi neuronal dynamics. However, the relative baselines for this high test-set performance is unknown. The results in this experiment establish NMSE performance baselines using RNNs trained on data where the FETi’s stimulus-response dynamics are obscured.

Figure 3.7 shows the spread of test-set performance for RNNs trained 20 times on the randomised FETi data from each of the 5 animals. Here a large spread of errors for each model ranging between 99% and 106% is observed. Additionally, the means are consistent across animals.

A Mann Whitney U test on these scores shows us that the test-set errors obtained by the RNNs trained on obscured data are very significantly larger than the test-set errors of RNNs trained on the FETi data (U=25.0, p-value=0.006). This result simultaneously establishes good baselines for RNN performance on FETi data, whilst also concretely proving that the RNNs are not overfitting to the FETi training data.

3.3.3 Experiment 3: Biological Plausibility of the Models

The first experiment established that RNN models perform favourably in comparison to other models on the FETi task. The second experiment showed that this performance wasn’t the result
Figure 3.8: Plot of FETi response vs. model response for a 5 Hz input stimulus. Here, positive femoral-tibial angles denote flexion and negative angles denote extension. Both neuronal and model response is high in flexion, and inactive in extension. There appears to be a close fit between model response and FETi response, despite the higher frequency variation in the FETi responses.

Table 3.5: Mean Square Error of RNN, TDNN and LNL models predicting the output for a 5 Hz constant stimulus. Lowest errors are highlighted in bold.

<table>
<thead>
<tr>
<th></th>
<th>Locust 1</th>
<th>Locust 2</th>
<th>Locust 3</th>
<th>Locust 4</th>
<th>Locust 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>36.41</td>
<td>31.97</td>
<td>34.52</td>
<td>30.29</td>
<td>39.31</td>
</tr>
<tr>
<td>TDNN</td>
<td>39.46</td>
<td>46.40</td>
<td>9.00</td>
<td>26.50</td>
<td>14.20</td>
</tr>
<tr>
<td>LNL</td>
<td><strong>13.4</strong></td>
<td><strong>16.30</strong></td>
<td>44.90</td>
<td>82.90</td>
<td>33.33</td>
</tr>
</tbody>
</table>

Figure 3.8 shows the RNNs response of the model to a 5 Hz sinusoidal input, plotted against the actual response to the 5 Hz stimulus from the FETi neuron. Note that this response was extracted from the neuron after the Gaussian White Noise (GWN) experiment, and that this data was unseen by the model during training, validation and testing. Although the figure shows a close fit between the neurons output and the models estimate of the output, there appears to be some high-frequency noise in the FETi output which the RNN output isn’t displaying. As expected, the output response is maximal on the down swings in the stimulus (flexion). Additionally, both neuron and model are unresponsive to extension. This result is similar for all 5 animals.

Table 3.5 Lists the NMSE between FETi output and model response to the 5 Hz stimulus, for RNN, TDNN and LNL models. Here, the RNN has the poorest NMSE performance of the three.
models. However plotting model estimate against FETi response shows a predominantly close fit, as in Figure 3.8.

Figures 3.9a and 3.9b depict the frequency spectra of both model and FETi response to a 5 Hz stimulus. Here the frequency is plotted on a linear range from 0-50 Hz. In both, frequency content appears at multiples of 5 Hz, with decaying amplitude. This observation is consistent across responses for all 5 animals. Additionally, Figure 3.9c shows the frequency spectrum of the 5 Hz signal after its passed through a half-wave rectifier, showing the similar features of the two.

Finally, Figure 3.10 shows the result of running a cross-correlation analysis between sinusoidal signals at various frequencies ranging between 5 and 50 Hz, and their respective model responses. This cross-correlation is a measure of the delay between input stimulus and output response, as measured from the model. The figure produced is averaged across 20 models trained on Locust 3, but the results are similar across all animals. For all models trained across all the animals, there is between 20-23 milliseconds of latency between peak flexion and peak response from the FETi models.
Chapter 3 Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron

3.4 Discussion

So far a series of experiments have been described, aimed at exploring the efficacy of Recurrent Neural Networks (RNNs) in modelling the functional characteristics of the locust Fast Extensor Tibiae (FETi) motor neuron. Methods used to collect, analyse and preprocess stimulus-response data from the FETi neuron of multiple locusts have been discussed. Additionally, experiments for training, validating and evaluating RNN models for the FETi modelling task have been described, and their results detailed. This section provides an explanatory analysis of the results from these experiments. Where applicable, these results are explained in the context of external literature.

3.4.1 Training the Models

The correct choice of hyperparameters is a crucial step in any modelling task. RNNs in particular, are very general learning algorithms. They are designed to be able to learn and model from most types of temporal data. Intuitively, all the possible instantiations of RNN models can be visualised as points in a space spanned by various hyperparameters. These hyperparameters include both structural considerations like number of hidden layers, number of units in each hidden layer, etc. and optimisation considerations like choice of optimisation algorithm, batch size, and weight initialisation procedure. Any particular instantiation of an RNN will have a predefined set of these hyperparameters. Generally, the space of hyperparameters is uncountably large, and there is no deterministic method for knowing which set of hyperparameters will perform best on any given task (LeCun et al. (2015); Bengio (2012)).
Chapter 3 Locust neuronal system identification with Recurrent Neural Networks: the Fast Extensor Tibiae motor neuron

One of the most highly explored deficiencies of RNN models is the tendency for gradients to either vanish or explode during training. (Bengio et al. (1994)) showed that vanishing gradients rapidly drop to zero and provide no learning signal for models where the temporal sequences get arbitrarily large. Similarly, exploding gradients provide too much learning signal and change model weights rapidly enough that training does not converge. The vanishing/exploding gradient problem makes training RNNs difficult without appropriate hyperparameter search.

The vanishing gradient problem can be avoided by initialising the weights of the RNN within certain bounds. The default weight initialisation procedure for artificial neural networks is to sample scalar values from a Gaussian distribution centered at zero (LeCun et al. (2015); Bengio (2012)). However, Pascanu et al. (2013) derived mathematically sufficient bounds on model weights within which vanishing gradients do not occur. The default Gaussian sampling procedure selects weights which fall outside these bounds with relatively high probability. In fact, the vanishing gradient problem was observed in initial runs of the first experiment in this work, resulting in poorly trained estimators (Figure 3.4). Glorot and Bengio (2010) showed that using their ‘Glorot’ weight-initialisation procedure ensured that model weights were contained within the bounds required to inhibit the vanishing gradient problem. In this work, using the Glorot procedure vastly improved the performance of the trained RNNs (Figure 3.4).

In the literature, exploding gradients are prohibited by simply clipping the absolute values of gradient vectors to a certain predetermined threshold (Pascanu et al. (2013)). In this work, gradients were clipped to a maximum L2 norm of 5. Typical gradient magnitudes range from 0.001 to 1. Here, the gradients are relative change of weight strength in connections between neurons in the artificial neural networks. None of the training runs in this work experienced non-converging models due to exploding gradients with this scheme.

Apart from the weight initialisation procedure, other hyperparameters of the network such as the number of hidden layers, and number of units in each hidden layer were derived with the random search algorithm. The problem of hyperparameter search for artificial neural network models is widely studied, and there are a variety of algorithms to choose from. Bergstra and Bengio (2012) empirically showed that random search is able to find higher performance hyperparameters in a fraction of the time required by other search methods like grid search.

Additionally, random search was only run once, and for modelling the FETi data of a single locust. All other training runs used the same hyperparameters. Interestingly, this still resulted in high performing trained models, in contrast to Meruelo et al. (2016)’s work where they used a hybrid of particle-swarm and genetic algorithms to derive the highest performing model for each animal. The consistent performance of RNNs despite choosing hyperparameters once is likely indicative that the processes producing the data for all 5 animals are roughly the same: the hyperparameters do not have to change to accomodate any marked differences in the data. Significant differences in performance would strongly indicate that FETi dynamics change between animals.

In this work, the significant increase in NMSE score for animal 4 compared to all the other animals was cause for suspecting a difference in FETi dynamics. However, running additional hyperparameter searches on models for animal 4 showed no significant performance improvement. The probable cause of the performance gap between animal 4 and all other models is experimental
noise. It is likely that this noise obscured some of the dynamics of the FETi in the data, resulting in the respective model generalising poorly on the test data.

An interesting result from the experiments was that Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models did not perform as well as RNN models, across all animals. The Normalised Mean-Squared Error (NMSE) was much higher for these models than the RNN models on the test data. Additionally, the spread of errors displayed by the LSTM and GRU models over 20 different trials was far lower than the spread of errors displayed by the RNNs. The lower NMSE scores imply RNNs are better able to model the FETi dynamics. The higher variance of errors implies that the set of RNNs capable of modelling FETi dynamics is much larger than the set of LSTMs/GRUs for the same task, where the latter converge to very similar parameter sets during training, leading to very similar (hence lower variance) NMSEs.

In the literature, memory-augmented recurrent models like LSTMs and GRUs far outperform RNNs on traditionally ‘difficult’ machine learning tasks. These include applications like robot control (Mayer et al. (2008)), sign language translation (Huang et al. (2018)) and composing blues music (Eck and Schmidhuber (2002)). This is because these tasks require models which can ‘remember’ information over long many timesteps. In a similar vein to the vanishing gradient problem, Bengio et al. (1994) showed that RNNs are incapable of remembering bits of information over arbitrarily long sequences without stalling the training process. In an attempt to solve this problem, Gers et al. (1999) presented the LSTM, by augmenting the standard RNN with internal structure that learns to ‘remember’ and ‘forget’ information as a type of internal working memory. Similarly, Chung et al. (2015) also presented the GRU model, which like the LSTM, incorporated memory augmentation to the standard RNN. At the time of writing this work, LSTMs and GRUs dominate the machine learning literature for time-series modelling because of their ability to model challenging dynamics with long-term temporal effects.

The poor performance of LSTMs and GRUs in this task can be attributed to a lack of long-term effects in the FETi dynamics. This is corroborated by the result that the FETi dynamics are only dependent on stimuli within a 20ms time delay. Here, the sampling rate of the data is important. If the sampling rate was too high, the 20ms difference between input and output would likely have too many timepoints between them requiring the internal memory structures of LSTMs and GRUs to form the appropriate input-output associations. In this work, with a reduced 500 Hz sampling rate, there are roughly 10 timepoints between input and output, which is evidently a short enough span for the standard RNN to model.

A comparison of RNN performance to that of previous work on the FETi task showed that RNNs significantly outperformed the Linear-Nonlinear-Linear (LNL) model in generalising on unseen test data. Dewhirst et al. (2013) used the LNL model on the FETi task for the easy interpretability of that model class: where each of the linear and nonlinear parts of the model can be analysed separately to derive the implied functionality of the neuron. On the other hand, RNNs are completely black-box in the sense that their functionality isn’t easily interpretable. This is because RNNs don’t contain analogous linear-nonlinear cascades. The RNN contains a very large number of parameters to be able to model tasks of arbitrary complexity, and is highly nonlinear in its temporal dynamics (Lipton et al. (2015)). The RNN model effectively trades interpretability for model complexity, and it is this model complexity which allows it higher performance on the FETi dataGers et al. (1999); Chung et al. (2015).
Interestingly, RNNs display similar performance to Meruelo et al. (2016)’s Time-Delay Neural Network (TDNN) models. In terms of model complexity and interpretability, both classes of models are similar: they are all highly nonlinear, black-box models with low interpretability. This similarity in performance is despite the RNNs having no hyperparameter tuning apart from the first training run. Meruelo et al. (2016) perform a detailed hyperparameter search to find the best performing models for every locust. It is likely that Meruelo et al. (2016)’s TDNN models would also display similar performance without repeated hyperparameter tuning.

One of the dangers with using complex models like RNNs for modelling is overfitting, where the models are complex enough to memorise input-output datapoints in the training data without learning the underlying phenomena producing this data. Overfitting commonly manifests as models which are biased towards the training data, with very poor generalisation ability. As Bengio (2012) suggested, a common practice in the machine learning community is to detect overfitting by comparing training data NMSE with validation data NMSE. This way, overfitting can be detected by low training NMSEs where the models memorise the training data, and high validation NMSEs where the models fail to generalise to the unseen validation data. This work used a 30% heldout validation set to detect overfitting.

For the FETi modelling task, however, it can be argued that training-validation set NMSEs are not a good indicator of overfitting as these are both subsets of the same data with very similar characteristics. There is a possibility that models which are overfitting the training data would also report low NMSE scores on the validation data.

One practice in machine learning literature is to compare model performance metrics like NMSE with those expected from a random classifier (Bengio (2012)). For example a random classifier with N outputs would be expected to be correct (1/N)% of the time. This provides a baseline against which trained classifiers can be compared.

In this work, the baseline idea is used to compare the validation metrics against those of random classifiers. Here, the randomised classifiers were trained on data where the stimulus-response dynamics were obscured in the FETi data by randomising the inputs. The reasoning for this was that if RNNs trained on the FETi data were overfitting, they would report validation scores very similar to the baselines established by the random classifiers. However, the trained model NMSE scores showed significantly better performance by the FETi models than the randomised models. This yields a measure of confidence that the RNN models aren’t overfitting to the data.

### 3.4.2 Evaluating biological plausibility of the models

After validating trained RNNs to check for overfitting, the biological plausibility of the outputs of these models was investigated. With the models verified against overfitting to the the training data, the models were then queried for their response to 5 Hz sinusoidal inputs. These inputs were chosen for biological plausibility because they roughly approximate the stepping motion of a locust’s hind limb in a walking gait (Blackburn et al. (2010)). This is in stark contrast to the Gaussian White Noise (GWN) signals the models were trained on. The hypothesis explored here is that if RNN models can learn the underlying FETi dynamics from GWN during training, they should be able to replicate the output of the FETi for the 5 Hz sinusoid. Additionally, it should
be possible to extract other relevant information about FETi functionality from the RNNs by
querying these for various synthetic input stimuli.

Comparing the RNN outputs to FETi responses to 5 Hz sinusoids showed NMSE scores in the
30% to 40% range. When compared to the test data NMSE scores (17% to 27%), it seems
that model predictions are inconsistent with FETi responses for a walking input. Comparing
the NMSE scores on the walking stimulus against the scores reported for the TDNN and LNL
models revealed that the RNNs underperformed on the walking stimulus test compared to these
models. However, plotting the RNN predictions against FETi output reveals the sources of error
(Figure 3.8). The figure reveals that the RNN output is consistent across time where the FETi
responses vary, as can be expected of a noisy biological system. It is apparent that the model
fit is close, and that within each peak is variation from high-frequency noise. The presence of
high-frequency noise in the output is unsurprising. As Boyd and Martin (1956) showed, it is
possible for electrodes to record spontaneous electrical activity, especially in the non-spiking,
subthreshold regime.

High-frequency noise notwithstanding, the RNN models seemed to display half-wave rectifier like
behaviour. This corroborates Newland and Kondoh (1997a)’s conclusions from their work on
modelling FETi dynamics. Intuitively, claiming that the RNN behaves as a half-wave rectifier is
a sensible statement, as the leg’s reflex response is for the extensor muscle to respond strongly to
leg flexion and either weakly or not at all to leg extension (Burrows (1996)). However, plotting
the frequency spectra of FETi and RNN responses to 5 Hz sinusoids and comparing these to the
output spectra of an ideal half-wave rectifier showed slight differences.

For the 5 Hz input, FETi showed frequency content at multiples of 5 Hz. When simulated
with multiple sinusoids with different frequencies of X Hz, the RNNs displayed similar spectra
at multiples of X Hz. This property is also displayed by the half-wave rectifier. The difference,
however is that the half-wave rectifier doesn’t have output frequency content at multiples of X
Hz, it has these at multiples of 2X Hz, where X is the frequency of an input sinusoid. This
analysis shows that FETi’s behaviour closely approximates, but isn’t identical to a half-wave
rectifier.

The pre-modelling data exploration stage showed an approximate time delay of 20 milliseconds
between peak flexion and the corresponding peak FETi response. A cross-correlation analysis
from the model responses to sinusoids of various frequencies (Figure 3.10), corroborated this
observation. Additionally, the results from the models showed that this delay is consistent
across locusts, and over a range of different frequencies. A biological implication of this result is
that any movement of the locust hind leg requires approximately 20 milliseconds to propagate
through the appropriate neuronal circuitry before a reflex response is calculated for the extensor
muscle. Additionally, the frequency-range consistency shows that the informational pathways
do not change with frequency.

Newland and Kondoh (1997a) provided a thorough analysis of FETi dynamics by interpreting
the patterns of peaks and valleys in the first- and second-order coefficients of Wiener models
trained on FETi data. Jing et al. (2012) provide an impressive analysis of interpreting biological
behaviour from the kernel coefficients of Wiener models. Unfortunately, because of their inherent
complexity and nonlinear structure, RNNs are not equally open to interpretation. However, as
this work shows, they are significantly better estimators than more interpretable models like the LNL model. The tradeoff here is between interpretability and accuracy of estimators.

### 3.5 Conclusion

This chapter showed that RNNs are well suited to the task of modelling the FETi motor neuron. Comparing the results of the RNN models with previously used methods showed that RNNs significantly outperform Dewhirst et al. (2013)’s LNL and perform similarly to Meruelo et al. (2016)’s TDNN when trained and tested on synthetic stimuli. Querying the RNNs with more natural 5 Hz stimuli showed poor generalisation when compared to TDNNs and LNLs. However, a visual analysis showed that the RNN output for 5 Hz stimuli closely matched corresponding FETi responses, with the error stemming from high-frequency noise and natural variation as an be expected from a biological system. In short, the results of the modelling study was in agreement with what is known about FETi behaviour: that it reacts positively in a reflex response to flexion.

Like all previous known approaches studying FETi behaviour, this work took the approach of building models from synthetic GWN stimuli, and querying the models for expected neuronal behaviour. Unfortunately, the RNN models were not as interpretable as the Wiener first- and second-order Wiener kernels trained by Newland and Kondoh (1997a). This work circumvented the interpretability problem by querying the RNNs with stimuli of interest and analysing the responses; implicitly relying on the generalisation performance of the RNNs to unseen stimuli. Another shortcoming of the approach taken in this chapter is that the RNN training is framed as a regression problem, where continuous-valued stimuli are mapped to continuous-valued neuronal responses. Conversely, most of the neurons worth studying are spiking neurons with discrete responses to continuous stimuli. Later chapters explore the modelling of spiking neurons using RNNs, establishing metrics for model generalisation and analysing the RNNs by querying these with novel stimuli.
Chapter 4

Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

4.1 Introduction

Sensory neuroscience encompasses a wide body of work aimed at exploring the functionality of neurons underpinning sensory systems such as vision, olfaction, hearing and proprioception. How do these sensory systems encode information about the world in a concise enough way to allow an animal to sense, survive and thrive? The literature on sensory neuroscience contains work dedicated to producing quantitative and computational descriptions of sensory systems aimed at answering such questions (Dayan et al. (2003)).

Recently, there has been a rapid growth of computational power and an accompanied rise in the amount of data collected from increasingly high-resolution neurophysiological experiments (Devous Sr et al. (1998); Zhu et al. (2017)). This has inspired researchers to use sophisticated computational modelling strategies in studying complex neuronal systems. Sensory neurons are particularly amenable to computational modelling because their inputs can be tightly constrained, and their outputs easily observed as action potentials (Wu et al. (2006)). These input-output data can be used to derive statistical models of sensory neuronal behaviour. Additionally, because sensory neurons lie in the periphery of neuronal circuits, these are less susceptible to nonlinear behaviour than more central neuronal systems (Carandini et al. (2005)) which generally process more abstract, semantic information (Kowalski et al. (1996)), potentially making the modelling task easier.

The classical approach to studying sensory systems is to organise a series of experiments: each designed to query a sensory neuron with a specific hypothesis about how that neuron encodes and processes input stimuli (Wu et al. (2006)). On the other hand, System identification approaches
Chapter 4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

aim to construct statistical models of the neuron’s underlying dynamics which can be used to describe the neurons response to any given stimulus (Ljung (1987); Garg et al. (2017)). Generally, the modelling procedure involves observing the inputs and outputs of a black-box system, before using statistical techniques to identify a functional relationship between stimulus and response from the system. This way, system identification provides methods for deriving the functional characteristics of the neuron from data. This chapter presents a case for modelling proprioceptive sensory afferents using system identification.

Researchers use a wide range of modelling approaches to study sensory neurons each with different assumptions about the underlying nature of neuronal encoding. Of these, models using Wiener/Volterra series are the most prolific (Marmarelis (2004); Newland and Kondoh (1997a); Wu et al. (2006); Jing et al. (2012)). The first-order kernel of a Wiener series is used to derive a linear approximation to neuronal functionality when trained on responses to Gaussian white noise (GWN) stimuli (Marmarelis (2004); Jing et al. (2012)). Adding higher order kernels to the model expands this approximation to include nonlinear neuronal dynamics (Marmarelis (2004); Newland and Kondoh (1997a)). Second-order Wiener models have been used to study the nonlinear properties of a range of sensory systems including auditory (Lewis and van Dijk (2004)), visual (Anzai et al. (2001)) and proprioceptive afferents (Kondoh et al. (1995)).

While Wiener kernels are powerful function estimators (Marmarelis (2004); Newland and Kondoh (1997a)) and provide easily interpretable descriptions of the linear and nonlinear characteristics of a system (Newland and Kondoh (1997a)), these are particularly susceptible to estimation errors where there is a lot of background noise in the data (Dewhirst et al. (2013)). Additionally, Wiener-like identification models do not always converge to a good solution for complex stimulus-response datasets characterised by high levels of nonlinearity (Tötterman and Toivonen (2009); Palm and Poggio (1977a)). This is because it is difficult to estimate Wiener kernels beyond second order without collecting data that is exponential in the order of the kernels (Tötterman and Toivonen (2009)). These characteristics are especially troublesome in modern neuroscience, where there is an increased adoption of very high resolution techniques (Devous Sr et al. (1998); Zhu et al. (2017)) for experiments, and the data gathered from natural neuronal signalling can be noisy.

Recurrent Neural Networks (RNNs) have proven highly successful in modelling and system identification from large, complex datasets. In time-series modelling, RNNs have seen wide adoption with state-of-the-art results in academic research and industrial applications including Natural Language Processing (Goldberg and Levy (2014)), Speech Recognition (Hinton et al. (2012)), Machine Translation (Bahdanau et al. (2014)), and Natural Language Generation (Van Den Oord et al. (2016)). The reason for this success is that RNNs can be built with an arbitrarily large number of parameters, which can be fitted to large time-series datasets with very high accuracy (LeCun et al. (2015)). Modifying the capacity of an RNN to encompass greater levels of nonlinearity is simply a matter of stacking layers in a network, with no changes to the underlying learning algorithm (LeCun et al. (2015)). The demonstrated ability of RNNs in modelling complex processes from large datasets implies that these could be powerful tools for modelling and system identification of sensory neurons.

Chapter 3 showed the efficacy of RNNs in modelling the graded potentials of the locust fast extensor tibiae (FETi) motor neuron in response to Gaussian white noise (GWN) and walking
stimuli. In Chapter 3, RNNs were proven to be significantly better estimators of FETi dynamics than Linear-Nonlinear-Linear (LNL) models, and equivalent in performance to the more recent Time-Delay Neural Networks (TDNNs) which were also used for modelling FETi. The FETi modelling task in Chapter 3 was framed as a continuous-valued regression problem because the postsynaptic FETi recordings in this work were graded/continuous-valued responses to locust tibial movement. However, an overwhelming majority of neurons in any neuronal system is discrete-valued (Dayan et al. (2003)). These neurons propagate information in the form of discrete action potentials (spikes), where information can be encoded either in the rate of spiking or in spikes themselves (Brette (2015)).

This chapter explores the efficacy of RNNs in modelling the dynamics of spiking neurons. To do this, the functionality of proprioceptive sensory afferents sensing movements of the locust hind leg are investigated. These neurons are responsible for sensing leg movements and transducing these into an effective encoding which is relayed to neuronal circuits producing adaptive behaviour like reflex movements (Kondoh et al. (1995)). Kondoh et al. (1995) used Wiener kernels to analyse the functional properties of these sensory afferents. In their work, Kondoh et al. (1995) provided an excellent analysis of the functional characteristics of the afferents, as implied by the Wiener kernels. However, their work only used second-order Wiener kernel expansions, potentially restricting the capacity of these models in capturing more complex dynamics.

Chapter 3 showed that RNNs are not as easy to analyse than the more interpretable models like Linear-Nonlinear-Linear (LNL) filters or Wiener-Volterra kernel expansions (Wu et al. (2006)). However, in their ability to capture all of the complexity inherent in SETi data, RNNs may provide better generalisation (LeCun et al. (2015); Goodfellow et al. (2016)). That is, they may be able to provide higher fidelity responses to novel stimuli than the Wiener models. The generalisation ability of RNNs on unseen GWN data was proven in Chapter 3, where RNNs significantly outperformed LNL models. However, the performance of RNNs lagged behind TDNN and LNL models for novel stimuli that emulated natural walking.

The aim of this chapter is to validate the efficacy of RNNs in modelling the neuronal code of spiking sensory afferents. The entire procedure for modelling the data from sensory afferents is presented: preprocessing the data, training the RNNs on this data, validating the RNNs using held-out test sets and examining the biological implications from the trained RNNs. The biological implications from the RNNs are then compared with the work of Kondoh et al. (1995) on locust proprioceptive afferents to either corroborate or contradict the RNNs results.

4.2 Methods

4.2.1 Data Extraction

The data used for the experiments in this chapter was obtained by Kondoh et al. (1995) in their work on examining the functionality of proprioceptive sensory neurons in the locust hind leg. The following procedure was used for recording the data:

First, adult desert locusts (Schistocerca gregaria) were mounted ventral-side-up in modelling clay. The hind leg was rotated through 90° and fixed to the clay with the anterior face up at a
femorotibial angle of 60°. This configuration of leg placement allows the apodeme of the locust femorotibial chordotonal organ (FeCO) to be in the middle of its linear range of motion (Field and Burrows (1982)).

The FeCO apodeme was exposed by opening a window in the cuticle of the distal anterior femur (Figure 4.1). The apodeme was grasped with fine forceps attached to a vibrator. The apodeme was also cut distal to the forceps. Separating the apodeme from its anchor on the locust tibia allows the vibrator to be the only source of sensory input into the FeCO organ, via the apodeme.

A small window was cut in the cuticle of the ventral thorax, exposing the meso- and metathoracic ganglia. The exposed ganglia were supported on a wax-coated silver platform, and continuously perfused with a saline solution at 20° to 25°C. Nerve 5B1, which contains the axons of the sensory afferents, was treated with a 0.1% (wt/vol) solution of protease for 1 min before recording. The afferents were penetrated intracellularly in their axons using electrodes filled with a 2M potassium acetate solution. Electrode resistances were 50 – 80 MΩ.

The forceps holding the FeCO apodeme were moved with Gaussian White Noise (GWN) stimuli produced by filtering a pseudorandom binary sequence bandlimited to 200Hz. This filtered sequence was further low-pass filtered to generate three different white noise signals with cutoff frequencies of 27Hz, 58Hz, and 117Hz. The rising phase of the signal corresponded to flexion of
the femorotibial joint, and the falling phases to an extension. The dynamic range of the stimuli were in the range of 0°, corresponding to full flexion of the femorotibial joint, to 120°, the limit of linearity between tibial angle and apodeme displacement (Burrows (1996)).

GWN stimuli and the corresponding spikes from the FcCO afferents were sampled using a Digital-Analog Converter sampling at a 10kHz frequency. 35 different recordings were stored in this way, from 13 different neurons. Each recording was of a neuron being stimulated with a GWN of one of 27Hz, 58Hz, or 117Hz for approximately 60 seconds.

4.2.2 Data Examination

An immediate observation from the recordings is that spikes are not discrete events. Figure 4.2 (b) displays the typical characteristics of neuronal spiking: a rising phase, with a depolarising peak, followed by a falling phase of repolarisation with a small undershoot from hyperpolarisation. These phases last approximately 10 ms and are consistent across spikes. In short, each spike is continuous-valued, and extends across a few timepoints in the recording.

The continuous-valued spikes in these recordings can be binarised to discrete 0/1 events for every timepoint. Characterising the data in this way allows the use of a suite of modelling techniques called binary logistic regression. Additionally, it is assumed that the informational content for this problem is contained within the underlying neuronal dynamics which generate the spikes, and not the spikes themselves. By modelling the output as a set of discrete events, the neuronal modeling problem is converted from a real-valued regression task to a binary classification problem: Given an input stimulus, should there be a spike at the output or not? A method for discretising real-valued spikes to discrete outputs is discussed later in this section.

Another observation is that some of the spikes are inconsistent with others. Each of the recordings obtained from the various locusts contains spikes of varying amplitudes (Figure 4.2 (b)). Furthermore, some of the spikes frequently co-occur with very small intervals (5 - 10 ms) between them, relative to the duration of each spike (10ms) (Figure 4.2 (b)). The lack of a refractory
period between these spikes suggests that these spikes do not originate from the same neuron (Dayan et al. (2003)).

It is likely that a single recording is a collection of various spikes from neighbouring neurons. Here, the recording is likely a linear summation of signals from a cluster of neighbouring neurons, including the sensory afferent under investigation. Spiking from neighbouring neurons serves no useful purpose to this investigation and can only confuse the data, hindering any further modelling work. Later, this section discusses a method for separating any background spiking activity from the spikes of the sensory afferents under investigation.

Another pathological feature present in a subset of the recordings was a slow, time-varying drift (Figure 4.7). Drifts of this nature are not a problem if the outputs are discretised to a binary form, but these can confuse the discretisation process itself. Any algorithmic process of identifying spikes by their amplitude can be damaged by inconsistencies in the absolute values of the spikes caused by these drifts. This section later presents a method for detrending and removing these drifts using techniques demonstrated by Santos et al. (2017).

Finally, the frequency content of the input stimuli was examined. The aim was to identify any spurious frequency content within and outside the bandwidth of the original GWN stimuli. None of the recordings displayed any pathological content within the GWN bandwidth. The frequency content was approximately uniformly distributed within this bandwidth, as expected from a GWN signal. Outside of the input bandwidth, there are smaller energies from high-frequency noise. These artefacts are removed by low-pass filtering and downsampling the data.

### 4.2.3 Data Preprocessing

Figure 4.7 illustrates a typical example of slow-moving drift present in some of the sample recordings obtained. This kind of drift can hinder downstream analysis like spike identification, thereby blocking any downstream modelling of the data. Preliminary experiments showed that bandpass filters were ineffective at removing the drift without also destroying important frequency components which form important identifying features for spikes in the recordings.

In their work on predicting neuronal connectivity patterns for the Fast Extensor Tibiae (FETi) neuron, Santos et al. (2017) used the Singular Spectrum Analysis (SSA) method to remove nonlinear trends in their data. In short, SSA breaks down a time series as a linear weighted sum of a set of component time series, one of which is likely to be the drift we want to remove. Reconstructing the original time series without the slower drift components effectively maintains the important spike features in the recordings (Figure 4.7 (a) and (b)).

Spikes in the afferent recordings can be identified from typical characteristics (rising phase, falling phase, undershoot, refractory period). These spikes last approximately 10ms each, and extend across multiple timepoints when sampled at a frequency of 10 kHz. This work takes a spike-base approach to neuronal modelling, assuming that the information about neuronal dynamics is contained within the occurrence (or absence) of individual spikes to a given stimulus. Any other information, including spike-rates is ignored in this case.

A sensible approach to modelling is therefore to discard irrelevant temporal information about the spikes, and retain the fact that a spike occurs (or not). Spikes in the recording are therfore
Figure 4.3: Illustration of the procedure used for binarising spikes. (a) A continuous-valued signal (blue) is thresholded, in this case to values above 0. Consecutive groups of timepoints with values above the threshold are grouped together (green). (b) The maximal value of each group is taken as the discretised value of where a spike occurs. (c) Binarising a continuous-valued electrode recording (above) in this way creates a discretised spike-train (below).

Spikes were binarised by first deriving a threshold for each recording. Any signals beyond this threshold allows easy identification of spikes, separating these from unimportant noise artefacts which typically have much smaller amplitudes. The timepoints between each upward-cross and downward-cross of the threshold are then collected (Figure 4.3 (a)). Here, each of these sets of timepoints represents a spike, but must be narrowed down to a single point where the spike is discretised to. In this case, the timepoint with the maximal potential is assigned a 1 for spike occurrence. All others are assigned a 0. The result of this procedure is shown in (Figure 4.3 (c)).
Figure 4.4: Clustering spikes by distinguishing features. (a) Each spike can be characterised by its amplitude and the time before the previous spike. (b) Normalising these numerical features in the range \([0, 1]\) and plotting each spike as a point in a 2D graph shows that these can be clustered with algorithms like K-means clustering. Clusters with the largest relative amplitude were chosen for discretisation and modelling.

and shows that discrete spike events coincide with each point of maximal amplitude across all spikes of interest.

One concern is that recordings from the afferent may contain spikes from neighbouring neurons. A closer look at suspect spikes indicates that they have two distinguishable features: smaller
amplitudes and smaller intervals with their previous spikes (spiking within the refractory period of the main spike) (Figure 4.4 (a)). These two features were used to identify the errant spikes.

First, all spikes in the recording were spikes were plotted in this two-dimensional feature space spanned by two axes: spike amplitude and time interval between previous spike (Figure 4.4 (a)). In (Figure 4.4 (b)) every spike is represented as a point in this space by computing the amplitude of the spike at its discrete timepoint, and the time between the current discrete spike and the previous discrete spike. This representation allows easy identification of clusters of similar spikes. The occurrence of distinct clusters is a strong indication of the recordings containing signals from multiple spike sources. For each of the recordings, only the cluster with the largest amplitude was chosen for binarisation and further modelling, as these are the likeliest source of spikes from the sensory afferent due to proximity with the electrode.

Recordings in the dataset lasted between 60-120 seconds, and was sampled at 10kHz. There were therefore approximately 600,000 – 1,200,000 timepoints in every recording. However, the GWN input signals have a maximal bandwidth of 117Hz, implying that the important frequency components of the recordings were contained within a minimum sampling rate of 234Hz, according to the Nyquist-Shannon sampling theorem (Shannon (1949)). Only $234 \times 60 = 14040$ timepoints were therefore needed to fully model the input-output characteristics in the data. Modelling the data at a higher sampling rate than 234Hz risked introducing higher frequency noise in the data. Additionally, a higher sampling rate than this simply introduced more data with no informative content. To mitigate the risk of completely removing content which may serve a useful purpose in the modelling tasks, the data was downsampled from 10,000Hz to a conservative sampling rate of 1,000 Hz. This was done by passing the data through a low-pass anti-aliasing filter with a 1000Hz bandwidth, before downsampling the data to 1000Hz.

Finally, the data was split into overlapping chunks using time-windows (Figure 4.5). Intuitively, a long time window encompasses longer term phenomena like neuronal memory, where a certain input many timepoints ago affects the output at the current timepoint. A shorter time window will disregard these effects. Using long time windows can make the modelling more accurate, but only upto a point where the memory effects last. Beyond this simply introduces noise and confusion into the model. A 180ms window was empirically found to work best for the sensory afferent modelling. Successive chunks of data were chosen to overlap each other by 20ms (Figure 4.5).

4.2.4 Modelling with Recurrent Neural Networks

Before beginning any modelling, the data from each of the recordings was split into three parts: training set, validation set and test set each with a 60%-10%-30% split, respectively. The largest proportion of the data (60%) was kept as a training set, which was used to train the Recurrent Neural Network (RNN) models. 10% of the data was a validation set, which was used to tune the hyperparameters of the RNN. The final 30% of the data was a test set to evaluate the quality of the trained models.

RNNs are a family of neural network models which are general enough to model any kind of time-series data. RNNs and their memory augmented counterparts: Long Short-Term Memory
(LSTM) and Gated Recurrent Unit (GRU) models, embed no prior assumptions about any data into their structure, apart from assuming temporal continuity between successive points in a time-series. The ability to model the underlying functional characteristics of a stimulus-response time series makes RNNs ideal for the sensory afferent modelling task.

The RNNs can be tuned to a given task by adjusting a set of model hyperparameters. For any given RNN, hyperparameters include: number of hidden layers, number of hidden neurons in the hidden layer, activation functions for each layer, etc. In this work, RNNs were tuned for the sensory afferent task by running a random search algorithm (Bergstra and Bengio (2012)) for hyperparameter combinations which yield the lowest errors on the validation sets of a preselected subset of afferent recordings. The best hyperparameter combination found by random search was held fixed for training all RNN models on all the sensory afferent recordings. By preselecting hyperparameters in this way, the search space of RNN models was reduced to the ones which perform best in modelling the sensory afferent data (see Appendix 2).

Upon narrowing the space of RNNs, the parameters of each RNN were optimised on the training set from each of the recordings. In this case, the RNN parameters are the feedforward and recurrent weights between successive layers of neurons in the RNNs artificial neural network. Optimisation was done using the Adam algorithm for Stochastic Gradient Descent (SGD) (Kingma and Ba (2014)). This procedure uses a cost function to compute the cost for a specific set of

**Figure 4.5:** Illustration of the windowing procedure used to split a stimulus-discretised spike-train dataset into chunks of data. Overlapping windows of 180ms are used to split each chunk. Successive chunks are not entirely independent, with 20ms of overlap between them.
Chapter 4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

53

Figure 4.6: Examples of receiver operating characteristic (ROC) curves plotting the proportion of true positives in a prediction set to the proportion of false positives in the prediction set as the probability threshold for two predicting classifiers is varied from 0 to 1. (a) An example of a ‘good’ classifier where the true positive rate stays close to 1 as the threshold is varied. (b) A relatively worse classifier which has lower true positive rates as the probability threshold is varied. In both cases the dashed line indicating equal true-positive rates and false-positive rates is indicative of the worst type of classifier which is no different from random guessing. The area under the ROC curves (AUC) can be considered as a scalar metric of the quality of a classifier. Here, a perfect classifier has an AUC of 1 and a random-guessing classifier has an AUC of 0.5.

model parameters and a subset of the data. Gradients of the parameters with respect to the cost function are then computed. This allows the algorithm to find optimal, cost-minimising parameters of the model using iterative gradient descent.

This work uses binary cross-entropy as a cost function. Simply stated, cross-entropy is a measure of the difference between two probability distributions $p(x|s_t)$ and $q(x|s_t)$. In this case, $p(x|s_t)$ is the true probability of a spike $x$ from a sensory afferent, given a certain stimulus $s_t$. On the other hand, $q(x|s_t)$ is the estimated probability of a spike given stimulus $s_t$, as predicted by an RNN model. Cross entropy is minimal when $p(x|s_t) = q(x|s_t)$ for all stimuli $s_t$. Using SGD to minimise cross-entropy with respect to the RNNs parameters implicitly reduces the difference between an RNNs internal probability model for spiking and the distribution of the sensory afferent dynamics on which it is trained.

4.2.5 Model analysis

To measure the quality of the train models the area under receiver operating characteristic (AUC) metric is used, as is common for comparing logistic regression classifiers (Bradley (1997)). The final models output the probability of a spike given a stimulus $s_t$ for every timepoint $t$. As before, this is denoted by the probability function $p(x|s_t)$. Correspondingly, the probability of non-spiking for a given stimulus $s_t$ can also be derived: $(p(\sim x|s_t) = 1 - p(x|s_t))$. The problem here is that spikes from the test set are discrete binary events. On the other hand, the trained models emit a continuous probability value between 0 and 1 for spiking. One method for gauging how well this probability model measures up against the real spiking data is to threshold the probability outputs. All outputs above this threshold are assigned a spike event and all probabilities below the threshold is assigned a no-spike event.
How can such a threshold be chosen a-priori? The common technique used by practitioners is to vary the threshold between 0 and 1. For every threshold in this range, the true positive rate (TPR) and false positive rate (FPR) can be measured. TPR is the ratio of true spike events predicted by the probability (true positives) model to the total number of spikes in the data. FPR is the ratio of no-spike events that the model predicts as spikes (false positives) to the total number of no-spike events in the data. For a perfect predictor, TPR is 1 and does not vary as the threshold increases from 0 to 1. As the threshold is changed, TPR and FPR can be plotted against each other on a graph (Figure 4.6). This graph is called the receiver operating characteristic (ROC) of a classifier. The ROC provides a visual illustration of the quality of a classifier, depicting how the classifiers predicted true positives vary with false positives for various thresholds. However, easily comparing models against each other requires a scalar measure of quality as opposed to a graphical one. For this, the area under the receiver operating characteristic (AUC) can be used. Because an ideal predictor has a TPR of 1, it should also have an AUC of 1. Conversely, a predictor which randomly predicts spike/no-spike events will have equal values of TPR and FPR as the threshold varies, and an AUC of 0.5 (Figure 4.6).

For the trained RNN models in this work, the AUC metrics are measured against the test data for each recording. This is done to avoid the danger of the models overfitting to their training set, where the model memorises the input-output characteristics of samples in the training data, and fails to generalise to novel stimuli. The data in the test set, by definition, contains data the model hasn’t been trained on. Test data is therefore a good test for the models generalisation onto novel stimuli. If the models learn the functional characteristics of the sensory afferent data they are trained on, they should exhibit test data AUCs close to 1.

Despite demonstrating the performance of the RNN models on unseen test data, there is a risk that the models are overfitting the data by simply ‘memorising’ one-to-one stimulus-response relationships. In such a situation, the models should display the defining trait of overfitting: high training data performance with low test data performance. However, because both test and training data are drawn from the same distribution, it is likely that memorised input-output samples are present in the test data too. This would lead to a high test-error performance, falsely implying good generalisation from the models.

To counteract this phenomenon, models were trained 50 times on ‘randomised’ versions of each of the afferent recordings. Here, for each recording, the GWN stimuli are randomised (at the same frequency bandwidth of the original GWN stimuli), with the output spikes held constant. The aim of this is to obscure the underlying dynamics producing the data. Training models on randomised afferent dynamics helps establish a baseline for classifiers which overfit to the training data. For each recording, the spread of test data AUCs across the 50 models is compared to the AUC of the model trained on the original recording.

### 4.3 Results

#### 4.3.1 Data Preprocessing

An examination of the data noted slow time-varying drifts in a subset of the recordings. Figure 4.7 illustrates a typical example of the change in amplitude with time caused by such drifts. The
Figure 4.7: Illustrations of Singular Spectrum Analysis (SSA) used to detrend time series. (a) A repeated sinusoidal signal (top) is trended by adding an $x^2$ component (middle). SSA is used to detrend the drifting signal, recovering the original structure with much smaller drift magnitudes (bottom). (b) Using SSA removes similar drift in recordings of the sensory afferents, retaining the spiking structure.
previous section discussed a method for removing errant spikes from neighbouring neurons by clustering in amplitude and time before previous spike. If the spike amplitudes for spikes from the afferent arent consistent in time, there is a danger that the clustering procedure can remove informative spikes, which can hinder further modelling and analysis. It follows that the signal requires detrending to remove any drifts before spike select. Figure 4.7 (b) illustrates the result of using Singular Spectrum Analysis (SSA) to remove slow drift components from an afferent recording.

The strength of SSA is that any time series can be expressed in a vector space as a sum of principle components (Golyandina and Usevich (2010)). Expressed this way, the basis components that constitute the signal can be found algorithmically. Essentially, the SSA procedure sees any time series as a weighted linear sum of orthogonal component time series, where each weight is a measure of the components importance to the time series. In this case, arranging the linear sum without the drift components leaves behind the core detrended time series.

Figure 4.7 (a) shows a cursory experiment with using SSA to detrend a simple time series. First, a time series is constructed by summing two phase-shifted sinusoids of different frequencies: $5 \ast (\cos(4\pi X) + \sin(8\pi X))$. Next, $X^2$ is added to this time series to simulate a slow drift. Using SSA to decompose the time series and reconstitute the signal without the slow drift component shows a detrended signal (Figure 4.7 (a)).

A major concern was that the binarised spiketrain may contain spikes recorded from neighbouring neurons, which dont represent the functional dynamics of the subject afferent(s). Figure 4.4 shows how spikes can be clustered based on two features: spike amplitude, and time between previous spike. For the modelling experiments, only spikes from the cluster with the largest amplitude were chosen. The remaining for filtered out from the discretised spike train. The argument for choosing the cluster with the largest amplitude is that spike voltages from the subject afferent are the least attenuated when recorded by the electrodes due to their proximity to the electrode compared to neighbouring neurons.

After detrending and clustering the neuronal outputs, these were binarised into a discrete spike-train. Figure 4.4 illustrates the discretisation procedure, where consecutive timepoints which span a spike are grouped together, and the timepoint at the maximal depolarisation is chosen as the spike point. Here, the timepoint is assigned an output value of 1, regardless of the amplitude or duration of the spike. All non-spike timepoints are assigned a value of 0.

### 4.3.2 Model building and analysis

On preliminary modelling experiments, the RNNs converged to good estimators on some recordings, but were very poor estimators on others. Incidentally, the datasets with poorly converging estimators also had far fewer spikes to learn from. Because the number of spikes in the data were many orders smaller than the number of no-spikes in the data, the cost of false negatives from the estimators was negligible to the gain of true negatives. Unsurprisingly, the RNNs converged to estimators which guessed no-spike even if a spike occurred. Figure 4.8 plots the ratio of spike to no-spike events for the various recordings. The number of no-spikes far outweighs the number of spikes in the recordings.
To counteract this imbalance effect in the data, a reweighting scheme was used on the cross-entropy loss function. Assigning higher weights to the false negative terms penalises the model disproportionately more for a false negative than a false positive (Lin et al. (2018)). Intuitively, this reweighting scheme should encourage the model to err on the side of spike events in balance with predicting no-spike events. Figure 4.9(a) shows a comparison of test-set errors for select recordings with and without this imbalance-weighted augmentation of the cost function. Using a reweighting scheme vastly improved the quality of the final estimators.

Training the models saw a sharp decrease in cross-entropy for successive training epochs. Figure 4.10 illustrates how the average cross-entropy loss on the training set decreases during training. Upon initialisation of parameters, the RNNs are essentially random classifiers and simply randomly guess spike/no-spike at the output. On training the RNNs to tune the model parameters to reduce the cross-entropy loss, the models become better predictors of the probability of spiking/non-spiking given an input stimulus. This phenomenon is evidenced in Figure 4.10, where the training cross-entropy loss starts high on model initialisation, and rapidly decreases as the parameters settle to good estimators of the training data. Tuning the model parameters to minimise the weighted cross-entropy loss function through successive iterations sees them becoming better estimators on the training data.

Despite converging to good estimators of the training data, there is a danger that RNNs can overfit to the training set. Here, the concern is that RNNs memorise the training samples without learning the underlying neuronal dynamics which produced these stimulus-response samples. To examine how well the RNNs generalise outside the training set, the models performance was examined on unseen test data. By comparing spiking from the afferent in this test data to the RNNs responses for the same stimuli, a measure of the models generalisation can be derived.
Figure 4.9: Test data spike trains with respective predictions from data trained with (above) cross-entropy loss. (below) weighted cross-entropy loss. The models trained with cross-entropy loss overfit to predicting 'no-spike' for every timepoint, eliciting no spiking for any stimuli. Models trained with weighted cross-entropy are penalised higher for false negatives and therefore learn to predict 'spikes' for the stimuli most likely to elicit these from the afferent.
Figure 4.10: Illustrations of the change in cross-entropy on training and validation data for successive training iterations (epochs). (above) The training and validation loss for a cross-entropy loss function. (below) The training and validation loss for a weighted cross-entropy loss function. In both cases, successive training iterations see the training loss decrease rapidly. The losses on validation data correspondingly decrease, despite not being used for training. For the weighted cross-entropy loss case, validation losses saturate due to the increased false-negative penalties.
Chapter 4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

Figure 4.11: Final area under receiver operating characteristic (AUC) performance scores on test data for models trained on all 35 afferents. For comparison, typical scores for a random, uninformative classifier should be 0.5. AUC scores for a perfect classifier should be 1. This figure shows that all models form good estimators of their respective afferents.

Figure 4.9 shows spiking from a single afferent stimulated by GWN from the test data. Overlayed on these is the respective models prediction of where spiking should be. Note that model predictions are continuous-valued. For every timepoint in the time series, the model predicts a \([0, 1]\) probability of emitting a spike at that timepoint. To obtain the spike train illustrated in Figure 4.9, the probabilities output by the RNNs can be thresholded and timepoints where the probability exceeds this threshold can be assigned as spikes.

Discretising the RNNs probability outputs in this way requires choosing an adequate threshold. However, the best threshold for a model is not known a-priori. In this work, the threshold is varied between 0 and 1. For each threshold level, the true positive rate (TPR) and false positive rate (FPR) are computed. Graphing these values together yields the receiver operating characteristic (ROC) (Figure 4.6). To yield a scalar, numerical measure of performance, the area under the receiver operating characteristic curve (AUC) is calculated. Figure 4.11 plots the AUC scores for RNN models trained on sensory afferents. These AUC scores are computed on the unseen test data for each model. The AUC scores for these models are predominantly high (>70%) indicating that the models are very good estimators of their respective afferent dynamics.

Figure 4.12 plots the spread of test-set AUCs for models trained on randomised input stimuli. Here, the AUCs are roughly centered around 0.5, as is typical of a random classifier on any binary classification task. A z-test for statistical significance between the AUCs of models trained on randomised stimuli and models trained on true stimuli revealed p-values less than 5%. Models trained on the real recordings therefore display significantly higher test AUCs 4.11, showing that the RNN models learn good estimators of the underlying dynamics of the sensory afferents and that the models aren’t simply memorising the training data.
4.3.3 Spike-triggered average

After confirming the efficacy of trained RNN models on unseen test data, new GWN stimuli were generated. These were used to query the trained models, effectively querying the afferent models for their responses to novel stimuli. Each model was queried with GWN signals from the same frequency bandwidth of the original input stimuli in the recordings used for training. The stimuli preceding each spike elicited by the model by 100ms were collected. These stimuli were averaged together to derive the spike triggered average (STA) as an estimate of the linear receptive field of the afferents (Lee and Schetzen (1965)).

Figure 4.13 illustrates single examples of the variety of STAs observed from the RNN models. The position-, velocity- or acceleration-sensitivity of these STAs can be estimated using the patterns they exhibit. In short, STAs can be monophasic, biphasic or triphasic, each indicating position-, velocity- or acceleration-sensitivity respectively. Additionally, STAs can indicate position-, velocity- or acceleration-sensitivity in one of two directions, given the polarity of the STA (Jing et al. (2012)).

Figure 4.13 (a) illustrates a monophasic STA which responds to flexion of the tibia. Figure 4.13 (b) illustrates biphasic STAs indicating velocity-sensitivity with two polarities (both of flexion- and extension- sensitivity). Likewise, Figure 4.13 (c) illustrates a triphasic STA indicating acceleration-sensitivity with two polarities (both of flexion- and extension- sensitivity).

The STAs of all RNNs trained on afferent data show a variety of different receptive fields as implied by their STAs. Figure 4.14 (a) illustrates the collection of all STAs collected from all models. Here, a mixture of different STAs can be observed. Separating out the STAs by the number of peaks shows a clearer picture of the variety of STAs collected from the models trained. The least frequently observed STAs are position-sensitive (Figure 4.14 (b)). Among the position-sensitive afferents, only flexion sensitivity was observed. As Figure 4.14 (b) shows, the position-sensitive parts of the receptive fields begin approximately 40-50 ms before a spike.
Figure 4.13: Illustrative examples of the types of spike-triggered averaged (STA) stimuli observed from the models. These are illustrated as normalised tibial angle between [-1, 1] plotted against time. Positive tibial angles correspond to tibial flexion. (a) A monophasic (single peak) STA indicating position sensitivity. (b) Biphasic STAs indicating velocity with two different polarities in the left and right graphs. (c) Triphasic STAs indicating acceleration sensitivity with two different polarities in the left and right graphs, respectively.
Figure 4.14: All spike-triggered averages (STA) from the RNN models plotted together. (a) All the STAs from the models. (b) Position-sensitive STAs. (c) Velocity-sensitive STAs of opposite polarities in the left and right graphs. (d) Acceleration-sensitive STAs of opposite polarities in the left and right graphs. (e) STAs with no clear sensitivity observable from their patterns.
and last approximately 30 ms with a 10-20 ms delay before a spike is produced. Note that the
frequency with which neurons of certain sensitivities were observed reveals nothing about the
relative distribution of these sensitivities in the general locust population.

Figure 4.14 (c) and (d) illustrate all velocity- and acceleration-sensitive STAs respectively. Models which display these sensitivities display two types of polarities each. Velocity-sensitive STAs begin approximately 20-50 ms before a spike and have durations between 40-50 ms, with at most a 20 ms delay before a spike is elicited. Similarly, acceleration-sensitive STAs begin approximately 30 ms before a spike with relatively smaller (5-10 ms) delays before a spike is produced. Figure 4.14 (e) illustrates the remaining STAs whose patterns have more than three phases and do not fit any of position-, velocity- or acceleration- sensitivity. In short, the various STAs of trained RNN models depict one of position-, velocity-, acceleration- or unknown sensitivity. Counting the two polarities for flexion- and extension-sensitivity, a total of six different configurations are observed.

It should be noted that the STAs derived in this set of experiments are not purely position-, velocity- or acceleration- sensitive. In so much as these STAs are described by the number of peaks present, a predominant number of STAs have ambiguous STA descriptions. Figure 4.15 plots an illustrative example of such ambiguous STAs, which can best be described as a mixture of position-, velocity-, or acceleration- sensitivity. When describing the occurrences of STAs with a particular sensitivity (Figure 4.14), the predominant sensitivity was chosen for each STA, based on the relative size of the largest peaks.

Interestingly, increasing the frequency bandwidth changes the STA of the models. Increasing the bandwidth from 27 Hz to 58 Hz showed three types of transitions: 2 instances of position-sensitivity transitioning to acceleration-sensitivity; 6 instances from velocity-sensitivity to acceleration-sensitivity and 3 instances which remained velocity-sensitive at both input frequencies.

Similarly, increasing the frequency bandwidth from 58 Hz to 117 Hz showed two transition types: 2 instances of retained velocity-sensitivity in both bandwidths; and 5 instances of retained acceleration-sensitivity at both bandwidths.

This result suggests that afferent receptive fields are sensitive to lower derivatives i.e. position or velocity at lower frequencies and transition to higher derivatives i.e. velocity or acceleration at higher frequencies. The result also show that if the afferents are acceleration-sensitive at lower frequencies, they remain acceleration sensitive at higher frequencies. In short, no transitions from a higher derivative to a lower derivative was observed with increasing frequency.

4.3.4 Predicting lower frequency afferent data using higher frequency models

So far, the results of using RNNs to model afferent responses to Gaussian white noise (GWN) stimuli have shown that the RNNs form good estimators of the afferent dynamics at various bandwidths of white noise frequency. Additionally, the linear receptive fields of these afferents can be derived by taking the spike-triggered average (STA) from trained models. An interesting observation was that the STAs of models change with white noise frequency for the same afferent.
Chapter 4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

Figure 4.15: Examples of STAs with ambiguous sensitivity. (Above) Two predominant peaks followed by a smaller peak at 40 ms, indicating a predominant velocity component with a small sensitivity to acceleration. (Below) A larger negative (extension) peak indicating position sensitivity surrounded by two relatively smaller peaks indicating sensitivity to all of position, velocity and acceleration; with predominant position sensitivity.

These results imply that an afferent’s receptive field may change depending on the input frequency of the GWN stimulus.

Figure 4.16 illustrates the spread of area under receiver operating characteristic (AUC) scores of all models trained on afferent data with higher frequency inputs predicting data from the same afferent at a lower frequency. The results show that models trained on higher frequency input GWN are generally poor estimators of afferent dynamics at lower frequency inputs. Most of the AUCs are close to 0.5, indicating uninformative classifiers with equal rates of true- and false-positives, no different from random classifiers. There is one outlier with a significantly higher AUC (0.92). Figures 4.17 (a), (b), and (c) show that in the high AUC cases, the STA of the higher frequency models is identical to the STA of the lower frequency model, explaining why
Chapter 4 Locust neuronal system identification with Recurrent Neural Networks: Proprioceptive Sensory Afferents

Figure 4.16: Spread of AUCs for the four observed STA transitions. Here, models trained on higher-frequency afferent data are used for predicting the responses to lower-frequency inputs from the same afferent. The models’ responses are compared with the real afferent responses to derive an AUC score. For most of the transition types, the scores are centered around 0.5 indicating poor estimators. There are outliers in the velocity-to-velocity transition types and acceleration-to-acceleration transition types. Additionally, the velocity-to-acceleration transition types have a large variance of estimator AUCs with a large inter-quartile range and a higher median than the other transition types.

Given that the RNN models are very good estimators of afferent dynamics on their own datasets, the poor performance when using these to test lower-frequency data of the same afferents suggests a dramatic shift in neuronal dynamics as input frequency varies from 27Hz to 58Hz to 117Hz. This is further evidenced by the shifts in STA which either remain at the same sensitivity and change polarities, or change to lower-derivative dynamics e.g. velocity to acceleration.

Given that there are 5 observed cases of velocity-sensitive afferents remaining velocity-sensitive at higher frequencies and 4 cases of acceleration-sensitive afferents retaining acceleration-sensitivity at higher frequencies, why are there poor AUCs in these cases? Of these 9 neurons which retain sensitivity at low and high frequencies, 7 switch polarities explaining the poor estimation scores. The two which do remain the same are the outliers observed in Figure 4.17. In fact, the group which display the highest median AUC scores are the afferents which transition from velocity-sensitive to acceleration-sensitive. In this case, the high-performing estimators have STAs where the predominant peaks line up almost perfectly (Figure 4.17 (c) ), explaining their similar functionality with correspondingly higher AUCs.
Figure 4.17: STAs of cases where models trained on higher frequency inputs show high AUC scores when predicting the afferent responses to lower frequency inputs. In all cases, the model STAs on lower frequency inputs are identical to model STAs on their original high frequency input data. (a) Acceleration-acceleration transition STAs with an AUC of 0.71, here all of the STAs are similar, with the low-frequency model STA lagging less than the higher frequency model STAs. (b) Velocity-velocity transition STAs with an AUC of 0.92. Here all STAs are nearly identical. (c) Velocity-acceleration sensitive STAs with an AUC of 0.72. Here, the STA of the high-frequency model’s prediction shares a large extension (negative) peak with the STA of the low frequency model.

4.4 Discussion

Before beginning any modelling experiments, the data recorded from sensory afferents were examined for any confounding factors. Here, data examination was a crucial step to reveal undesirable qualities in the data like missing datapoints or sampling errors. Electrophysiological recordings are particularly susceptible to such noise and corruption (Newland and Kondoh (1997a)). For example, recordings taken intracellularly can pick up spikes and noise from neighbouring neurons. Additionally, experimental errors such as slipped or drifting electrodes can skew the data collected from patch-clamp recordings (Heinemann and Conti (1992)). Where models are used to learn the underlying dynamics which produce the data in these recordings, such confounding factors can lead to incorrect conclusions from modelling experiments. A visual examination of the afferent recordings quickly revealed where such confounding factors were present.

Data examination can also inform any modelling work. Spotting useful trends, patterns or features in the data can inform the choice of algorithms, parameters and hyperparameters in downstream modelling experiments. The afferent recordings considered in this work were simply described: input Gaussian white noise (GWN) stimuli with corresponding outputs from afferents
in response to these stimuli. Both inputs and outputs were time-series data in that temporal sequences of stimuli produced corresponding temporal sequences of afferent responses. Modelling these would therefore require algorithms capable of exploiting features embedded in temporal sequences.

A crucial property of the afferent recordings is that both stimuli and responses were continuous valued. Here, the inputs were continuous values between minimum and maximum extremes of tibial extension or flexion. The outputs were continuous voltages of membrane potentials as recorded by the electrode within the neuron. The most straightforward method of modelling this data would be to adopt the approach in Chapter 3, where RNNs were used to predict continuous-valued outputs for a given stimulus. However, this approach would be erroneous as the important information contained in the afferent recordings is not in the continuous-valued output potentials, but the occurrence (or lack) of spikes for a set of input stimuli. The output spike potentials were therefore discretised using the method outlined in Section 2.

In choosing which spike coding scheme to model, there are two overarching philosophies in the literature: spike-based vs spike-rate coding (Brette (2015)). The former posits that the informational content of a spike train is contained entirely within the occurrence (or not) of a spike in response to a particular stimulus (Dayan et al. (2003)). The latter, a more classical view of neuronal information coding, proposes that the firing rate of a neuron conveys the informational content of that neuron's processing (Dayan et al. (2003)). Although there are a number of studies examining which of these two philosophies best describe neuronal encoding schemes, the matter is still unsettled (Brette (2015)). This chapter set out to prove that RNNs are capable of modelling the occurrence (or lack) of spikes from sensory afferents given an input stimulus. Therefore, this work takes a spike-based view of the afferent encoding: modelling the dynamic outputs of locust sensory afferents as spike/no-spike in response to particular stimuli.

Another problem noticed during the data examination was the likelihood of spikes from neighbouring neurons being recorded. Although the afferent recordings were intracellular, there was evidence for errant spiking activity, including differing spike amplitudes and very short inter-spike times, well within what would be within the expected refractory period of a spike. There is a vast corpus of work on separating spikes from potentially different sources in extracellular recordings (Lewicki (1998)). In short, the majority of these methods use features which can easily distinguish between different spike sources. In this chapter, clustering spikes by amplitude and inter-spike times proved useful. Upon separating spikes in the afferent recordings into clusters, the cluster with the greatest amplitude was chosen for further modelling experiments. Because the magnitude of spike amplitude is a measure of proximity to the recording electrode, this scheme was most likely to extract spikes from the afferent closest to the electrode.

Measuring the amplitude of spikes for clustering assumed a stationary signal recorded by the afferents. However, a visual inspection of some of the afferent recordings showed non-stationary behaviour in the form of slow time-varying trends in the signal, likely due to small movements of the electrode for the duration of the recording. The presence of these spurious trends made it difficult to measure the relative magnitude of spike amplitudes, resulting in spikes being erroneously clustered. Santos et al. (2017)’s work proved a singular spectrum analysis (SSA) to be successful at detrending neuronal recordings. Following their results, this work successfully used the SSA scheme to detrend afferent recordings before clustering and discretising spikes.
There is a large discrepancy between the 10000 Hz sampling rate used for the afferent recordings and the 27/58/117 Hz GWN bandwidth of the stimulus inputs. From a modelling perspective, the important informational content is contained within the GWN frequency bandwidth. Incorporating extended frequencies beyond this range simply serves to introduce high-frequency noise which can confound the data and reduce the efficacy of modelling. Additionally, a higher sampling frequency simply means additional data, increasing the computational demand to derive an accurate model. The data was therefore downsampled 10x to 1000 Hz before any modelling experiments. The choice of 1000 Hz safely contained the frequency content of the input stimuli, which should be entirely contained within a minimum sampling frequency of 234 Hz suggested by the Nyquist-Shannon sampling theorem (Shannon (1949)). Weber et al. (2017) showed empirical evidence that downsampling data beyond the Shannon frequency erases important information required for modelling. In this chapter, a safety factor of approximately 4x more than the minimum sampling frequency was empirically found to be computationally efficient, without erasing important information.

The downsampling was done before discretising the afferent outputs to discrete spike-trains. This was done to ensure that the timepoints attributed to a spike were well aligned with the occurrence of spiking in the downsampled version of the original signal. However, the membrane potentials in the original recordings were found to contain membrane potentials with spectral content above the 500 Hz upper band assumed by the 1000 Hz downsampling. The recordings were therefore passed through a 500 Hz low-pass filter before downsampling to 1000 Hz to avoid spurious aliasing effects in the spike discretisation stage (Mitra and Kuo (2006)).

4.4.1 Model building and analysis

The work in Chapter 3 showed that RNNs make significantly better estimators than Wiener models and Linear-Nonlinear-Linear models in modelling the dynamics of the fast extensor tibiae (FETi) motor neuron. Additionally, this work showed that RNNs form marginally better estimators than memory-augmented variants of the RNNs at the FETi modelling task. In the machine learning literature, these memory augmented variants of RNNs: the long-short term memory (LSTM) and gated recurrent unit (GRU) models, have been shown to be much more effective at modelling large, complex time series, especially if these time series have memory effects, where the output at a given timepoint is dependent on an input or sequence of inputs many timepoints in the past (Bengio et al. (1994); Gers et al. (1999)).

Bengio et al. (1994) showed that using RNNs on complex datasets with memory effects can lead to vanishing gradients: a phenomena where any learning signal present in the data degrades over many timesteps. The gradients of this learning signal with respect to the parameters of the RNN vanish to zero at exponential rates, which can stall training. Gers et al. (1999) empirically demonstrated that memory-augmented LSTMs were far better at modelling data with memory effects, as the memory augmentations were capable of learning to memorise longer-range data without erasing the learning signal. However, Chapter 3 showed that RNNs were better than LSTMs in modelling the dynamics of a motor neuron without long ranging memory effects. Like the FETi motor neuron, the sensory afferents investigated in this chapter are also unlikely to have long-range memory effects, as these are most likely to respond to immediate sensory stimuli (Wu et al. (2006)). This was the justification for choosing RNNs for the afferent modelling task. If it
was found that RNNs were not able to adequately model the afferent data, this would increase the possibility of memory effects in the data, and LSTMs would be a natural alternative. However, RNNs were shown to converge to good estimators of the data in preliminary experiments, and these were chosen for further modelling work.

Chapter 3 showed that RNNs form good estimators of neuronal processes which map Gaussian white noise (GWN) input stimuli, to continuous-valued output responses from the neuron. However, a large proportion of the neurons in any nervous system communicate through discrete spikes (Dayan et al. (2003); Burrows (1996)). Bolstered by the success of RNNs at the FETi modelling task, this chapter set out to prove that RNNs are capable of modelling the dynamics of spiking neurons too. Doing this required an alternative to the mean squared error loss function used to train RNNs in Chapter 3, as this assumed continuous-valued as opposed to discrete outputs.

Instead of framing modelling as a per-timepoint, continuous-valued regression task, this chapter posed the afferent modelling problem as a binary regression task. In the machine learning literature, binary outputs are framed in terms of emission probabilities. Here, the output of a time series is expressed as the probability of emitting a 1 or 0 at every timestep. From a probabilistic perspective, Bishop (2006) explained that a loss function can be expressed in terms of the cross-entropy between the model probability and the data output. Furthermore, LeCun et al. (2015) explained that the cross-entropy loss is a special case of the Kulback-Liebler divergence between two probability distributions. In short, binary-cross entropy is a measure of the divergence between the emission probability distribution output by the model and the binary distribution of the data outputs. Tweaking the RNN parameters to reduce this divergence is functionally equivalent to reducing the disparity between the models spike producing distribution and the latent probabilistic process which produced the spike train in the recordings: i.e. the underlying functional characteristics of the sensory afferents (Bishop (2006); LeCun et al. (2015)). In this chapter, the cross-entropy function is used to train RNNs on sensory afferent data instead of the mean-squared error function used for modelling FETi in Chapter 3.

Framing model training as a problem of reducing a binary cross-entropy loss necessarily requires the RNN models to output a probability of a spike in response to a given stimulus. Essentially, the models output values continuous in the range \([0, 1]\), despite the requirement for modelling spike outputs as a discrete process. Here, the outputs can be thresholded where probabilities above a certain threshold can be assigned as a spike. Likewise, probabilities below the threshold can be designated no-spike.

Unfortunately, the best threshold for a given model is not known a-priori. In the machine learning literature, a common method for reporting the quality of a classifier is to vary the threshold at various points in the range \([0, 1]\) and report how the true positive rate (TPR) and false positive rate (FPR) vary with the threshold (Hanley and McNeil (1982)). For a perfect classifier, the TPR will be 1.0 for all thresholds. Conversely, for an imperfect classifier, the TPR will be equal to the FPR as the threshold is varied. Instead of reporting the numerical values for TPR and FPR as the threshold is varied, these values can easily be plotted in a 2-D graph (Figure 4.6) called a Receiver Operating Characteristic (ROC) curve.

Additionally, a concise numerical measure of model quality can be derived from a models ROC. The illustration in Figure 4.6 illustrates that a low-quality classifier has a lower area under the
curve than a high-quality classifier. In the machine learning literature, this Area Under receiver operating Characteristic (AUC) is used as a surrogate measure of model performance, especially in situations which require a comparison of various models for a classification task (Hanley and McNeil (1982)). In this work, AUC was used as a measure of trained model performance. Trained models showed relatively high AUC scores (> 70%), significantly higher ($p < 0.05$, z-test) than models trained on randomised data which demonstrated lowed AUCs centered around 50%. In comparison, a perfect classifier would achieve AUC scores of 100%. Measuring the quality of trained models using the AUC metric is effective in comparing between the theoretical worst-case and best-case performance of 50% AUC and 100% AUC, respectively. To compare, Kondoh et al. (1995) framed the afferent modelling problem as a continuous valued task, modelling the graded afferent responses as-is, and reporting on mean-squared error (MSE) as a measure of performance.

4.4.2 Biological interpretation of RNNs

Confident that the trained RNN models form good estimators of afferent dynamics, the next series of experiments attempted to use these RNNs to derive concise descriptions of afferent characteristics. Kondoh et al. (1995) used the first- and second-order Wiener kernels to describe the afferent receptive field. This method of using Wiener kernels to characterise neuronal dynamics is prolific in the literature (Marmarelis (2004); Marmarelis and Marmarelis (1978)). A strength of Wiener type models is that the series expansion coefficients form good estimators of first-, second-, and higher-order behaviours of a neuron stimulated by Gaussian white noise (GWN). Recurrent Neural Networks (RNNs), in contrast are relatively uninterpretable because these trade off model interpretability for model complexity and the ability to accurately model complex dynamics in very large datasets (LeCun et al. (2015)).

Freshly generated GWN was used to query the trained RNN models. The neuronal responses immediately preceding every spike in the output spiketrain were collected and averaged. The resulting spike-triggered averages (STA) were used to characterise first-order afferent dynamics. Mathematically, a STA is the average of the stimuli immediately preceding every spike in a spike-train. The STA derived from a GWN stimulus is an unbiased estimator of a neuron's receptive field (Paninski (2003), Sharpee et al. (2004)). Unlike traditional approaches which use steps, ramps and sinusoids, GWN stimulates the entire receptive field of the neuron (Dayan et al. (2003)). In the literature, the use of STAs forms the core of a body of work aimed at using information-theoretic approaches to neuronal functional characterisation (Pillow and Simoncelli (2006); Davidson et al. (2007); Farina et al. (2005)). STA is especially widely used in investigations of neurons in the visual cortex (Dayan et al. (2003)) because it is particularly easy to stimulate these by presenting GWN stimuli to the visual field in the form of checkerboard patterns with randomised pixels, and averaging the stimuli which cause strong spiking.

Lee and Schetzen (1965) showed that the STA is functionally equivalent to the first-order kernel of Wiener series expansion. The STAs derived in this work can therefore be compared with Kondoh et al. (1995)'s work on characterising first-order dynamics of sensory afferents. Jing et al. (2012) showed that various patterns of first-order Wiener kernels can be interpreted as position-, velocity- and acceleration- sensitivity in a neuron's receptive field. The equivalence of
first-order Wiener kernels and STAs means these patterns can be extended to describe the STAs derived from RNN models.

Examining the STAs of RNNs trained on each recording showed a variety of sensitivities to all of position, velocity and acceleration. Additionally, each of these sensitivities were divided into two polarities e.g. flexion-sensitive position as opposed to extension-sensitive position. This broad range of afferent receptive fields was reported by Matheson and Field (1990) and Kondoh et al. (1995).

Afferent dynamics were characterised as one of position-, velocity- or acceleration-sensitive as in Matheson and Field (1990)s work. In fact, the STA results show that the afferent sensitivities are a graded combination of all these sensitivities. This result was also noted by Kondoh et al. (1995) who observed a mixture of sensitivities in their characterisation of afferents from first-order Wiener kernels. This result shows that sensory afferents in the locust FeCO encode sensitivities to all of position, velocity and acceleration from the locust tibia. The varying extents of sensitivity from each afferent serves as redundancy in a system of afferents all sensitive to different parts of the same input in parallel (Kondoh et al. (1995); Burrows (1996)).

An interesting result is that a given afferents STA sensitivities change depending on the input stimulus frequency. At a 27 Hz GWN stimulus frequency, most of the afferents were either position- or velocity-sensitive. As GWN frequencies increase to 58 Hz or 117 Hz, these sensitivities either remain the same or change. If sensitivities remain the same, a large proportion of the afferents switch polarity for example from extension velocity sensitivity to flexion velocity sensitivity. If sensitivities change, these transition to higher differentials e.g. position to velocity or velocity to acceleration. This phenomenon of shifting frequency was also noted by Kondoh et al. (1995).

Using models trained on higher frequency recordings to predict the responses of lower frequency recordings from a given afferent showed a mixture of results. Higher-frequency models yielding high AUC in predicting lower-frequency responses were only for the afferents whose sensitivities did not change with frequency. Models trained on higher frequencies which changed sensitivities to a higher differential generally yielded lower AUCs in predicting lower frequency responses. The lower AUCs persisted even if the sensitivity remained unchanged, but the polarity switched at a higher frequency. These results show that models trained on recordings from the same afferent are highly frequency dependent. These models do not form good estimators of lower frequency responses from the same afferent unless the receptive field stays the same. Given that the RNN models form good estimators of afferent dynamics within their specific frequency bandwidth but not in lower frequency ranges, the results in this work show that the dynamics of sensory afferents in the locust FeCO are largely dependent on input frequency.

### 4.5 Conclusion

This chapter showed that RNNs perform well at the task of modelling the FETi motor neuron. Compared to the work in Chapter 3, which used RNNs to model the graded action potentials of the FETi motor neuron, there were several technical challenges to overcome to obtain successful models for the spiking sensory afferent modelling task. The first of these challenges was one of
separating spikes from different sources, which was solved by clustering spikes based on distinguishing features observed during a visual examination of the dataset. The next was to detrend data collected from potentially drifting electrodes, which was solved using spectrum analysis methods in the same vein as used by REF. The next challenge was overcoming the imbalance in data between spiking and non-spiking events using a weighted cross-entropy scheme to penalise false negatives. Finally, it was found that the models failed to converge to good estimators for a wide range of hyperparameter combinations, which was solved by using random search to find the best performing hyperparameter combination for the task.

As opposed to the work on afferents by Kondoh et al. (1995), this chapter modelled spikes as discrete events. This reframe of the modelling problem from continuous-valued to discrete/categorical-valued required choosing a metric other than the mean squared error (MSE). Model performance was reported using the area under receiver operating characteristic (AUC) metric. Additionally, baselines for comparing the quality of the test set AUC performance were established by training RNNs on data with randomised GWN inputs. Although not seen in the literature, this method for achieving comparative baselines were useful for demonstrating that the trained RNNs were not overfitting to the GWN data, especially with the lack of performance comparisons to models used in Kondoh et al. (1995)'s prior work.

Finally, the trained RNNs were used to find the spike-triggered average (STA) of the neurons by querying the models with fresh GWN stimuli. The results of the STA analysis showed close parallels with Kondoh et al. (1995)'s work, but the question of how the models perform when queried with more natural signals remains unsolved. The next chapter explores building RNNs for modelling spiking neurons using synthetic GWN stimuli, and simulating neuronal behaviour by querying the trained RNNs with natural stimuli.
Chapter 5

Locust neuronal system identification with Recurrent Neural Networks: the Slow Extenor Tibiae motor neuron

5.1 Introduction

In the context of neuroscience, system identification approaches can produce quantitative models of neuronal function (Marmarelis (2004); Dayan et al. (2003)). Generally this involves deriving mathematical or statistical models of data collected from neuroscience experiments, a tradition dating back to the work of Lapicque (1907), who introduced the integrate-and-fire model of a spiking neuron. Once derived, models can be used to explore the dynamics of a neuronal system, to confirm or deny hypotheses of the systems function, or to predict the systems behaviour to external stimulus or manipulation (Lamb and Calabrese (2012); Marder and Taylor (2011a)).

In choosing the right modelling tools for a particular system, practitioners face two important considerations. First is the need for model interpretability. Interpretable models express dynamics in terms of mathematical concepts in terms of which the system can be understood (Gerstner et al. (2012)). The Hodgkin-Huxley model (Hodgkin and Huxley (1952)) is a famous example of such a model, expressing the dynamics of spike initiation and propagation in neurons using nonlinear differential equations. The second consideration is the need for model fidelity. This is a requirement for models which faithfully model the nuances and complexities of the underlying neuronal system. High-fidelity models generally have a large number of tunable parameters which are statistically fit to stimulus-response data (Nelles (2013); Marmarelis (2004)). Unlike interpretable mathematical models, these can be black-box where the internal workings of the model are unknown.

For modelling neurons with complex dynamics, interpretability and fidelity create competing design pressures. Interpretable models are generally highly simplified to provide analytically
tractable approximations of the neuron (Gerstner et al. (2012)). Such simplification trims away at complexity in neuron dynamics, usually to derive concise mathematical representations of the dynamics. Conversely, high-fidelity models attempt to capture all of the neuron dynamics without any simplification. Accurately capturing the nuances of a neuronal system comes at the cost of added model complexity, which makes such models less interpretable (Zhang and Zhu (2018); LeCun et al. (2015)).

There have been a number of attempts at efficient tradeoffs between interpretable and high-fidelity models. Hunter and Korenberg (1986) showed that the kernels of cascaded linear-nonlinear models can be used to model nonlinearities in a number of biological systems. Similarly, Jing et al. (2012) showed how the first- and second-order kernels of a Wiener/Volterra series expansion can be used to identify position-, velocity- or acceleration-sensitivity for neurons controlling the movements of the locust hind leg. While Wiener/Volterra models are powerful function estimators (Marmarelis (2004)) and provide easily interpretable descriptions of the linear characteristics of a system (Newland and Kondoh (1997a)), they are particularly susceptible to estimation errors where there is a lot of background noise in the data (Dewhirst et al. (2013)). Additionally, Wiener-like identification models do not always converge to a good solution for very large stimulus-response datasets (Töitterman and Toivonen (2009); Palm and Poggio (1977a)). These characteristics are especially troublesome in modern neuroscience, where there is an increased adoption of very high resolution techniques (Devous Sr et al. (1998); Zhu et al. (2017)) for experiments. With increased resolution come ever larger and complex dataset. Although a number of neurological investigation models based on Wiener/Volterra kernel expansions have a good balance of model fidelity and interpretability, these models have their limitations (Marmarelis (2004); Nelles (2013); Newland and Kondoh (1997a); Dewhirst et al. (2013); Meruelo et al. (2016)).

Outside of neuroscience, Recurrent Neural Networks (RNNs) have proven highly successful in modelling large, complex datasets. In time-series modelling, RNNs have seen wide adoption with state-of-the-art results in academic research and industrial applications including Natural Language Processing (Goldberg and Levy (2014)), Speech Recognition c3n22), Machine Translation (Bahdanau et al. (2014)), and Natural Language Generation (Van Den Oord et al. (2016)). The reason for this success is that RNNs can be built with an arbitrarily large number of parameters, which can be fit to large time-series datasets with very high accuracy (LeCun et al. (2015)). The potential for applying RNNs to large, complex neuroscience datasets is promising. Unfortunately, the large parameter count in RNNs comes at the expense of interpretability.

A solution to the interpretability problem is simulation (Wu et al. (2006)). Models trained on a neuronal system can be queried with a variety of input stimuli. If these models are high-fidelity representations of the neuron dynamics, the responses produced should be representative of the systems response to these inputs. Simulation can thus be used to explore various hypotheses about the system. The prospect of gaining an understanding about complex neuronal systems through simulation has inspired academic interest in large-scale simulation (Lang et al. (2011); Markram (2006)).

Chapter 3 explored using RNNs for modelling the dynamics of the locust Fast Extensor Tibiae (FETi) motor neuron. The results of that study showed promise, where RNNs and another type of artificial neural network (TDNN) significantly outperformed Linear-Nonlinear-Linear (LNL)
cascades, a more interpretable variety of function estimator. Additionally, Chapter 4 showed that RNNs also excelled in modelling the spiking responses of locust sensory afferents when these were stimulated with Gaussian white noise (GWN) of various frequency bandwidths. A spike-triggered average (STA) analysis of the RNN models corroborated Kondoh et al. (1995)’s prior observations on afferent dynamics.

This chapter investigates the dynamics of the Slow Extensor Tibiae (SETi) motor neuron controlling movements of the locust hind leg. Like the proprioceptive afferents, SETi is a spiking neuron. The techniques developed in Chapter 4 can therefore be applied here. SETi dynamics have already been extensively studied using Wiener models (Newland and Kondoh (1997a)). Despite the comprehensive study of SETi, the study was limited to using first- and second-order kernel expansions of the their models, possibly restricting the capacity of these models to capture more complex dynamics. This presents an excellent opportunity to use a higher fidelity model like RNNs for system identification of SETi.

This study proposes that RNNs may be better candidates for capturing the full complexities of SETis neuronal dynamics. In terms of interpretability, RNNs are difficult to analyse compared to the kernel expansion coefficients of the Wiener model. However, with their ability to capture all of the complexity inherent in SETi data, RNNs may provide better generalisation. That is, they may be able to provide higher fidelity responses to novel stimuli than the Wiener models. Greater insight into SETi dynamics may be derived by simulating SETi using an RNN model trained on some stimulus-response data derived from that neuron. The implied SETi dynamics derived from a process of simulation can then be compared to the findings of earlier works to either verify or refute the efficacy of RNNs in neuronal modelling.

The ability to simulate the dynamics of a black-box system using derived statistical models is one of the benefits afforded by the system identification approach. Training models on SETi data derived using a rich stimulus set like GWN means the models can generalise to simpler, more natural stimuli (Dewhirst et al. (2013)). The responses of the models to these stimuli can yield insight into SETi behaviour. Surprising or anomalous responses from the model can then be investigated to determine either how the model may be improved, or what the implications are for SETi dynamics (David et al. (2004)). Finally, using the simulation approach can be used to reveal novel principles of sensory coding, even in the absence of prior theoretical models. In this work, GWN-trained SETi models are tested for their generalisation ability using synthetic stimuli. Once metrics for these ability have been established and tested against appropriate baselines, the models are simulated using a variety of signals to derive the implied SETi dynamics in response to these stimuli.

One of the central questions of this chapter is the difference between SETi dynamics when data is collected in open-loop vs closed-loop. As far as is known, most of the literature on locust neurophysiology reports results from experiments carried out in open-loop, where neuronal feedback is removed (Meruelo et al. (2016); Dewhirst et al. (2013); Newland and Kondoh (1997a)). Removing feedback loops mitigates the potentially complex effects introduced by the interplay between input stimulus and neuronal feedback on the system under experiment (Roth et al. (2014)). This chapter argues that RNNs can sacrifice interpretability for the ability to model increased complexity. This provides an opportunity to close the neuronal feedback loop and compare RNN models trained on both open- and closed-loop setups.
Chapter 5Locust neuronal system identification with Recurrent Neural Networks: the Slow Extenor Tibiae motor neuron

The aims of this chapter are: 1) To use the techniques developed in Chapter 4 to train and evaluate RNN models on the SETi motor neuron. The models are trained on synthetic Gaussian white noise (GWN) data and evaluated on natural stimuli emulating locust walking gaits. 2) To explore SETi functionality implied by the trained RNNs by querying these with various different stimuli. The results of these are analysed and compared with the results of prior work on SETi by Newland and Kondoh (1997a). 3) To simulate and compare the implied dynamics from RNNs trained on open- and closed-loop data.

The first section presents the methods used to extract SETi data in both open- and closed-loop setups. Additionally, this section briefly describes building the RNN models and justifies various design choices for the SETi modelling case. The second section presents the results of a series of experiments. The first set of experiments is aimed at proving that the RNNs derived from SETi data are good estimators of SETi dynamics. The next set of experiments explores the implied SETi dynamics from the models by investigating the spike-triggered average (STA) of RNNs. Finally, the results of exploring SETi dynamics through simulation are described. Throughout this section, the results of models trained on open-loop data are compared with results from models trained on closed-loop data. The third section presents a narrative discussion explaining the results of the SETi modelling experiments. Context is provided using external literature, where applicable. Finally, this chapter concludes with a summary of what was discovered, including the strengths and weaknesses of using RNNs for neuronal modelling in general.

5.2 Methods

5.2.1 Data recording

The data used in this chapter was extracted by Devon Lewis using the procedure outlined below:

First, adult desert locusts (Schistocerca gregaria) were fixed ventral side up in modelling clay. The prothoracic, mesothoracic and right metathoracic leg were secured in place. The left metathoracic femur was fixed 45° from body on frontal plane with the tibia fixed at 45° on the transverse plane, allowing access to the apodeme of the FeCO. The femoro-tibial (FT) joint was fixed at 60° (Figure 5.1 (a)). Modelling clay was also placed over the head during experiments to remove any visual stimuli. Data was recorded in two conditions: open- and closed-loop. Under open-loop conditions, the limb remained in a fixed position. Under closed-loop conditions, the tibia was unfixed, allowing free movement of the knee after forceps were attached to the FeCO apodeme.

A window was removed from the lateral surface of the distal femur to access the apodeme of the femoral chordotonal organ (FeCO) (Figure 5.1 (c)). Air sacs were removed, taking care not to damage the accessory flexor muscle (Figure 5.1 (d)). Forceps, mounted to an LDS V101 permanent magnet shaker, were used to secure and control movements of the FeCO apodeme (Figure 5.1 (b)).

A window was also removed from the ventral surface of the thoracic cuticle to access the metathoracic ganglion (Figure 5.2 (d)). Air sacs were removed, and extracellular nerve recordings were taken with two 0.37mm silver electrodes under nerve 3B, between the metathoracic ganglion and the left metathoracic leg. A mixture of petroleum jelly and liquid paraffin was used to insulate
the silver electrodes from the haemolymph. The exposed ganglion was superfused with oxygenated saline (140mM NaCl; 10mM KCl; 4mM CaCl₂; 4mM NaHCO₃; 6mM NaH₂PO₄·2H₂O) at 3.25cm³min⁻¹ for the duration of the experiments.

50 second GWN displacement signals were generated in MATLAB with a mean of 0 and a standard deviation of 1 (Figure 5.3). Signals were bandlimited between 0 and 27Hz with a 5th order Butterworth low pass filter, converted to analog and amplified to generate a maximum FeCO displacement of ±0.47mm, replicating a ±43.5° change in FT angle about its central position of 60° (Figure 5.4). Displacement amplitude was calibrated and monitored using a Keyence LK-G32/37 laser displacement sensor system. Displacements were driven with 30 second intervals to limit habituation during the experiment while ensuring results were collected before fatigue and deterioration of the neuromuscular system.

Electrophysiological recordings of SET₁ were obtained under open- and closed-loop conditions. In open-loop conditions, the leg was held fixed with modelling clay during FeCO stimulation. In closed-loop conditions, the leg was allowed to move during FeCO stimulation. The order of recording conditions were randomised to remove any effect of fatigue, deterioration or adaptation throughout experiments. Analogue extracellular nerve signals were band-limited (5-100Hz) with a 50Hz notch filter. Signals were then amplified, digitised and recorded (fs=10kHz) using spike2 software (version 7.05b)
Figure 5.2: (A) Pathways of the four-identified motor neurones innervating the ETi muscle, showing extensions from the ventral cortex of the metathoracic ganglion through nerve 3, 4 and 5 towards the ETi muscle. Including Fast Extensor Tibiae Motor Neuron (FETi); Slow Extensor Tibiae Motor Neuron (SETi); Common Inhibitory Neuron (CI1) and Dorsal Unpaired Median Neuron (DUMETi). Diagram adapted from Burrows (1996). (B) Intact ventral surface of the thorax. (C) Exposed metathoracic ganglion (yellow). (D) Nerves 3 and 5 shown following the removal of the left main trachea, false coloured in blue and red respectively.

Figure 5.3: Example of a Gaussian white noise signal used to drive displacements of the FeCO apodeme.
Figure 5.4: Displacement of the FeCO apodeme as femoro-tibial joint increases from full flexion to full extension of the leg (Dewhirst et al. (2013); Burrows (1996)).

Figure 5.5: (a) A typical spike train recording from SETi. (b) A closeup of the spike train shows spikes of different shapes in the recording. (c) A further closeup shows a burst of continuous electrical activity within 7 ms, with no refractory period.
Chapter 5 Locust neuronal system identification with Recurrent Neural Networks: the Slow Extensor Tibiae motor neuron

5.2.2 Data preprocessing

An immediate observation of the neurons output shows low power noise artifacts amongst distinctly visible spikes (Figure 5.5 (a)). Because the neurons outputs are encoded entirely by the higher amplitude spikes, the noise provides no information, and only serves to muddle any modelling on the data. Fortunately, these artifacts can easily be removed by simply thresholding the output. Here, this threshold was was set lower than the range of spike amplitudes, but higher than noise range.

Another important observation is the variation in amplitude between spikes in the output recording (Figure 5.5 (b)). Additionally, some spikes appear within 1-2 milliseconds of a previous spike, well within the apparent refractory period of that spike (Figure 5.5 (c)). It is likely that spikes from nearby sources were recorded in addition to spikes from the SETi. These errant signals could also be high-frequency noise which can’t simply be thresholded out because they have large amplitudes of roughly equal magnitude to the SETi spikes. If this is the case, spikes from SETi must be separated from these errant background spikes or noise before any modelling and analysis are performed on the data. A method for separating SETi spikes from non-SETi spikes is described shortly.

**Figure 5.6:** (a) An input signal (blue) is thresholded above a predefined value (green). (b) Consecutive timepoints above the threshold are grouped, and inflection point for each group is identified as the spike. (c) Using this procedure on continuous-valued spiking outputs discretises a spike train (top) to a boolean output of ‘spike’ or ‘no-spike’ (bottom)
Recordings from each locust range from 30-60 seconds. Because each of these is recorded at a sampling frequency of 10000 Hz, there are roughly 300,000-600,000 points in each recording. To use this data for modelling, continuous-valued spikes in the output recordings were binarised to either 0 or 1. Here, 1 indicates the presence of a spike at a given discrete timepoint, and 0 indicates the absence of a spike at a time point. The procedure for binarising spikes began by grouping all consecutively positive values in the output (Figure 5.6 (a)). The inflection point of each group was taken where the gradient was zero. For a spike, the inflexion point corresponds to maximal depolarisation (Figure 5.6 (b)). For groups with multiple inflexions, the inflexion with the greatest magnitude was chosen. Finally, the time point at the occurrence of the chosen inflexion was assigned a value of 1, denoting a spike (Figure 5.6 (c)). Binarising continuous-valued spikes in this way converted the modelling task from a real-valued regression problem to a binary classification problem. After discretising the spikes, the datasets were decimated from 10000 Hz to 1000 Hz.

Here, the distinction between a regression problem and classification problem is important. Regression models map inputs to a continuous-valued output, whereas classification models map inputs to a set of discrete classes (Bishop (2006)). If the neuron encoded any information in the shape or amplitude of its spikes, regression modelling would be the ideal solution. For example, if the rate-coding properties of the SETi neuron were under study, framing the modelling as regression would be practical. However, this study considers the spike-coding properties of the SETi neuron. In this scenario, the neuron encodes its computational properties as the presence or absence of spikes, better framed as a binary classification problem of spike vs no-spike. Binarising the output recordings was, therefore, a precursor to the classification modelling task.

Next, each binarised spike train was filtered to make sure it did not include spikes from neighbouring neurons. First, each real-valued spike was cast into a point denoted by the minimum and maximum amplitude of the spike. Plotting all of the spikes in this way revealed a two-dimensional plot where separate clusters of points became clearly visible (Figure 5.7). To select the clusters most likely to be from SETi, the spike-triggered averages (STAs) of each cluster were compared. Clusters with STAs which show activation in response to slow flexion were selected, as these were most likely to belong to the SETi motor neuron (Newland and Kondoh (1997a)). Spikes from clusters not selected by this method were discarded.

Finally, the data was split into overlapping chunks using time-windows (Figure 5.8). Intuitively, longer time-window include a longer time span of data and thus encompass longer range phenomena in the underlying neuronal dynamics like neuronal memory. A shorter time-window would clip these effects. In this way, using longer time-windows has the potential to make the modelling more accurate, but only up to the point where memory effects last. Any longer and the larger time window simply introduces noise and confusion into the model. There is, therefore, a tradeoff between a short time window, which yields more data points; against a longer time window, which includes long-range nonlinear neuron behaviour, but yields fewer data points and can be potentially noisy. The windows were sampled with overlaps because these create more samples than without overlaps (Figure 5.8). The windowed samples were shuffled before training the models to maintain independence between batches of samples (Bishop (2006); Goodfellow et al. (2016)). In the following experiments, a 400ms windows with 200ms of overlap between successive windows was empirically found to yield the best model performance.
Chapter 5 Locust neuronal system identification with Recurrent Neural Networks: the Slow Extensor Tibiae motor neuron

**Figure 5.7:** Each point represents a spike represented by two features: minimum and maximum normalised spike amplitude. This figure shows three distinct clusters of spikes in this features space. Clusters were selected based on their spike-triggered averages (STAs).

**Figure 5.8:** The stimulus-reponse data is separated into independent chunks of 400ms each. This illustration shows the effect of windowing on a SETi response spike train. Here, successive windows overlap by 200ms.
5.2.3 Model Training

Before training models on the SETi recordings, the data was split into two mutually exclusive groups: training and validation. 90% of the GWN recordings were held as training data used to train the models. The remaining 10% was used as validation data. Keeping the validation data disjoint from the training data is important to ensure the trained models do not overfit on the training data (LeCun et al. (2015)). Overfitting implies poor model generalisation. The most obvious signs of such overfitting are when models achieve high accuracy on training data, but much lower accuracy on validation data (Bengio (2012)). There are heuristics for combating this kind of overfitting, all of which involve tuning model hyperparameters. The validation data was therefore used to find model hyperparameters which empirically provided the least amount of overfitting. These hyperparameters were chosen using the random search algorithm (Bergstra and Bengio (2012)), and were held constant for all models trained on the different SETi recordings (see Appendix 3).
Besides being stimulated with GWN inputs, each SETi neuron was also stimulated by directly moving the FeCO apodeme with a set of three different signals resembling locust walking gaits (Angarita-Jaimes et al. (2012)) (Figure 5.9). These walking recordings were used for testing the efficacy of the final trained models. These were purposefully kept separate from training and validation data, as models can overfit on the test data if used for hyperparameter tuning (Bengio (2012)). Because the trained models were in no circumstance exposed to this test data during hyperparameter or parameter tuning, model performance on the test data is a good indicator of the models ability to generalise onto unseen data. Models which perform well on the test set are highly likely to have learned the underlying neuronal dynamics of SETi from neuronal recordings alone.

To ensure that test set scores were not generated by chance, the models were repeatedly trained on randomly generated GWN signals with the same input frequency characteristics as the original GWN inputs in the recordings. As before, the models were trained to match the binary spikes from the original recordings, given the randomly generated GWN inputs. If the models trained on these random input-output mappings still perform well on the test data, this indicates a pathological condition where the model isn’t learning any underlying neuron dynamics and is generating high test-data accuracies by pure chance. Each of the models trained on a SETi recording were additionally trained on 20 randomly generated recordings. The test set performance of the SETi-trained vs random-trained models were compared.

Finally, training was done by optimising RNN weight parameters to minimise a binary cross-entropy cost function, as is typical for a binary classification task (LeCun et al. (2015)). Here, the cost was calculated as the sum of the models binary cross-entropy for every time point in the recording. By tuning the RNN weights to minimise this cost, the model learns to match the temporal dynamics of the training data. There is, however, a danger that these learned dynamics fail to generalise outside of the training set. This is the overfitting problem which was mitigated by using a validation set for tuning hyperparameters and a test set to compare the final model performance. The model hyperparameters in this task were derived using random search (Bergstra and Bengio (2012)) on the validation data from one animal. Once derived, the same hyperparameter set was used in training RNN models on data from all animals.

5.2.4 Model Evaluation

The final trained models output a $y_t \epsilon [0, 1]$ value for every time point, given an input $x_t$, for timepoints $t = (0, 1, \ldots, N)$. These outputs can be interpreted as the probability of a spike at a time point, given the stimulus at every preceding time point. In evaluating how well the models perform against the ground truth test data, these probabilities must be converted to spikes before a like-for-like comparison of spikes in the data vs corresponding spikes predicted by the model. The number of true positives and false positives in the prediction can be calculated and presented as a measure of model quality.

Converting spike probabilities to spike certainties involves thresholding the model probabilities. However, the best thresholds are unknown beforehand. To circumvent choosing a certain threshold for model evaluation, the threshold can be varied from 0 – 1 and the model can be evaluated
at every threshold in this range. For every chosen threshold, the true positive rate can be plotted against the false positive rate. The graph generated by this procedure is called the Receiver Operating Characteristic (ROC) curve. The area under the ROC curve (AUC) provides a quantitative description of model performance and is widely used to evaluate models on the binary classification task (Bradley (1997)).

The AUC scores for each trained model were noted for each of the different walking test recordings. Here, AUC scores near 1.0 imply far greater true positives than false positives, denoting excellent model performance with perfect generalisation. Conversely, AUC scores of 0.5 denote models with equal amounts of true positives as false positives, denoting poor model performance no better than random guessing. AUC scores of models trained on the SETi recordings were compared with models trained on randomly generated GWN recordings.

5.3 Results

In running gradient descent optimisation on the weight parameters of the Recurrent Neural Network (RNN) models, the change in cross-entropy losses per iteration was plotted for both the training and validation data (Figure 5.10). Here, both losses showed an appreciable decline in a few iterations, showing that the models rapidly learned from the training data.

Validation data losses were marginally higher than training data losses, indicating a slight amount of overfitting. Although the discrepancy between training-loss and validation-loss can be symptomatic of the models failing to learn the underlying dynamics of SETi (Bengio (2012)), here...
they were deemed to be within acceptable bounds. This is because the loss statistics in these experiments came from the empirically best-performing hyperparameters. Further tweaking certain hyperparameters such as dropout rates and early-stopping for regularisation led to further increase in the discrepancy between training and validation losses, with additionally higher validation losses in the trained models.

Next, the trained models were queried with walking stimuli from the test data of each recording. The models predictions were compared to the true output from the neuron using the Area Under the Receiver Operating Characteristic (AUC) metric (Figure 5.11). For all walking types, models trained on SETi recordings where the leg was fixed (open-loop) showed lower median test data performance than models trained on data where the leg was free to move (closed-loop).

Figure 5.11: Spread of AUC scores for models tested on each of the three walking gaits. (a) Models trained on SETi data from closed-loop experiments. (b) Models trained on SETi data from open-loop experiments. Open-loop models show higher interquartile ranges. Closed-loop models show lower interquartile range and higher median scores.
Additionally, the open-loop models showed a wider interquartile range in test-data performance than models trained on closed-loop data. With higher median and lower interquartile ranges in AUC scores for all walking tests, models trained on closed-loop data appear to form better performing and more reliable estimators. However, a Friedman test for statistical significance revealed no difference between models trained on open-loop data to models trained on closed-loop data ($p > 0.025$).

Figure 5.12 displays the spread of test-set AUCs for models trained multiple times on randomised output data. Here, the AUCs were roughly centred around 0.5, with relatively small variance, as is typical of a random classifier on a binary classification task. For all models, comparing the true-data AUC scores with random-data shows that true-data scores are higher. Using a t-test to determine whether the SETi-trained models were significantly different from the random classifiers revealed that all but 4 of the trained models had p-values less than 0.05, indicating significantly different performance. All the 4 models not significantly different from random classifiers were trained on open-loop data. This result shows that most of the models have learned good estimators of the underlying dynamics of SETi neurons from stimulus-response data, and that the models are not overfitting the training data.

To comment on the statistical testing methods used in comparing open-loop to closed-loop models and models trained on random data against models trained on nonrandom data: in the former test, both open- and closed- loop models showed skewed AUC distributions for each of the three walking gaits. Thus a nonparametric (non-gaussian assumption), multi-way test was required to test the null hypothesis that open-loop and closed-loop models were not significantly different from each other. This function was naturally performed by the Friedman test (Friedman (1940); Rice (2006)). In the latter case, AUCs for models trained on randomised data were approximately normally distributed, allowing a t-test to be used (Rice (2006)). In this case, all randomised AUCs from across the three walking gaits were pooled into the same population, under the
assumption that no informative signal was drawn from the type of walking input. Models trained on true SETi data were compared against this random population in independent trials. In both cases, significance levels of 5% were used.

5.3.1 Spike Analysis

After confirming the efficacy of trained models on simulated walking test data, new GWN stimuli were generated. These were used to query the trained models, effectively querying the SETi models for their responses to novel stimuli. For a single model, plotting all stimuli which elicit a spike at the output reveals a common pattern (Figure 5.13 (a)). Between 10 - 60 ms before a spike, most of the stimuli have a single dominant flexion peak. The 10ms between peak flexion and spike response is likely a signal propagation delay. There is high variance in the pre-spike input signals beyond approximately 60ms before the spike, indicating very low correlation between stimulus and response beyond 60ms before a spike. The mean of these responses reveals the spike-triggered average (STA), an illustration of the receptive field of the RNN. The single dominant peak of the STA suggests that the RNN and the SETi it is modelling are both position-sensitive (Jing et al. (2012)).
Plotting the STAs of all models together reveals a high degree of similarity between them (Figure 5.13 (b)). All STAs have a similar shape, with a single dominant peak suggesting position-sensitivity. For most of the STAs, the dominant peak preceds the spike by approximately 60 ms. Additionally, the time between peak flexion and spike is roughly 10ms for all STAs. The low variance in STAs between SETi models suggests a functional similarity between animals.

Figure 5.14 illustrates the STAs derived from models trained on open-loop SETi data vs models trained on closed-loop SETi data. Here, the mean STAs are largely the same apart from a slight difference in amplitude for the first large peak. A common feature in all the STAs displayed here are the smaller oscillations following the initial peak. It should be noted that these are artefacts from averaging the GWN stimuli which are bandlimited to 30Hz. As Figure 5.13 (a) shows, after the first peak, there is high variance between the spike-preceding stimuli. Averaging these high-variance signals should yield a neutral mean. But because these signals have an upper limit on frequency, the result of their mean yields an oscillation.
5.3.2 Model Prediction

Using models trained on the GWN data from one animal to predict the walking SETi responses of all other animals in the study reveals an interesting pattern (Figure 5.15). When the range of AUC scores of each model in predicting the responses of other animals were plotted, in 21 of the 25 trained models, the mean AUC scores are greater than 0.7. For all models but one, lower quartiles are greater than 0.6 indicating that the models are good estimators of SETi dynamics for at least 75% of the other animals. The high (above 0.7) mean AUC and correspondingly high lower quartiles of AUC scores indicate that models trained on one animal can be consistently high-fidelity estimators of SETi dynamics in other animals.
An interesting question is how well models trained on closed-loop SETi data predict the responses of open-loop SETi data or vice-versa. Figure 5.16 (a) shows the spread of AUC scores for models trained on closed-loop data being used to predict SETi data obtained in both open and closed loop. Here, the spread of both is almost identical with predictions for closed-loop data displaying a slightly higher median AUC than predictions for open-loop data. Predictions of open-loop data display marginally higher interquartile range indicating a slightly higher variance of predictions when closed-loop models are used to predict open-loop data. Interestingly, this pattern is also observed when open-loop models are used for predictions (Figure 5.16 (b)). Here too, the mean AUC scores are marginally higher for predictions on closed-loop data.

The only difference in the two cases is that models trained on open-loop data have more anomalies when predicting data from either open- or closed-loop. These anomalies are all below 0.55 AUC, indicating very poor prediction quality, identical to the output of a random classifier. Despite these anomalies, at least 75% of the predictions are above 0.7 for the closed-loop predictions and above 0.65 for the open-loop predictions. Additionally, the interquartile ranges are marginally larger in the open-loop case, indicating that models trained on open-loop data form slightly higher variance estimators when used to predict the dynamics of SETi in other animals.

5.3.3 Simulation Analysis

Assuming the RNN models generalise well on the SETi data and form good estimators of SETi dynamics, these models can be used to simulate SETi responses to various stimuli. Because the spike modelling task was framed as a binary cross-entropy problem, the models output the continuous-valued probability of a spike at every timestep, given the stimulus at preceding timesteps. Additionally, an examination of the spike-triggered average (STA) obtained by querying the models with gaussian white noise stimuli showed that the receptive field of the SETi is roughly 60ms long and is sensitive to flexion. In this section, simulation is used to further explore the implied dynamics of SETi from trained RNN models.

First, the models were queried with simple sinusoids of approximately 5Hz to simulate a walking gait (Figure 5.16). The resulting patterns imply a strong response to flexion, an unsurprising characteristic given the flexion-sensitive receptive field of SETi neurons. However, without prior knowledge of SETi’s flexion-sensitivity these may also imply phase-shifted sensitivity to SETi extension. To clarify which one of extension or flexion most elicits a spiking response from the models, these can be queried with half-wave rectified sinusoids biased to either one of the flexion or extension regimes.

Querying the models with half-wave rectified sinusoids in the flexion regime shows a sustained high spiking probability for the duration of the flexion (Figure 5.18 (a)). Decreasing the amplitude of the input sinusoid shows a gradual reduction in the spiking probability until the amplitude is under approximately 0.2 of the normalised tibial angular range in flexion (Figure 5.18 (c)) below which the probability of a spike drops dramatically. This result supports the claim of SETi being flexion-sensitive, as was discovered by analysing the STA of the models. Additionally, this result shows that the flexion sensitivity only occurs when the tibia is flexed above a threshold. The range of this threshold was empirically found to be consistent across all the SETi models, and corresponds to a tibial angle of approximately 62 - 63°.
Figure 5.17: Typical response from RNN models trained on SETi data on a sinusoidal stimulus. (Above) The output probability of spiking from the model. (Below) The input stimulus expressed as the tibial angle normalised to [-1, 1]. Positive stimulus values are tibial flexion and negative values are extension.

An interesting observation from Figure 5.18 (a) is that the high spiking probability begins much before peak flexion. In fact, this high spike probability region begins immediately after the start of a half-wave. This result contrasts the 10ms delay implied by the models’ STA. Observing the model’s response to half-wave rectified sinusoids in the extension regime sheds further light on this phenomenon. Here, single peaks of high probability can be observed immediately after the return of each extension to zero (Figure 5.19 (a)). Gradually raising the lower limit of the extension regime shows a rapid decrease in the magnitude of this probability peak (Figure 5.19 (c)), indicating that the models are excited when the tibial angle approaches flexion, even if the tibia does not enter the flexion regime. This pattern persists when the limit of the rectifier is raised into the flexion regime(Figure 5.19 (b)). Decreasing the frequency of the extension half-waves below approximately 10 Hz shows a sharp decline in the probability of spiking upon the tibia returning to zero(Figure 5.19 (d)). This indicates that the models are velocity sensitive to rapid movements towards the extension regime. This explains the apparently premature spiking probability in Figure 5.18 (a), where the sudden velocity change at the onset of a flexion half-wave elicits spiking from the model.

Querying the models for responses to step functions confirms the position- and velocity- sensitivity of the models observed so far. A step function from 0 to full flexion shows an initial burst of high probability as expected from the sudden change in stimulus velocity, followed by a sustained high probability of spiking as expected from the models’ position-sensitivity (Figure 5.20 (b)). Conversely, a step function to full flexion shows no change in probability output from the model (Figure 5.20 (b)).
Figure 5.18: Typical responses from RNN models trained on SETi data on a half-wave rectified sinusoid biased toward flexion. (a) With an input sinusoidal amplitude of 1. (b) An input amplitude of 0.1 with dramatically reduced probability of spiking. (c) An illustration of how the maximum spiking probability varies with the amplitude of the half-wave rectified sinusoid.

5.4 Discussion

5.4.1 Model Building

RNNs are very general learning algorithms designed for modelling time series data (LeCun et al. (2015); Goodfellow et al. (2016)). In the machine learning literature, the memory-augmented variants of RNNs have been successful at a number of seemingly difficult tasks involving modelling time series data modelling. The successes of RNNs include high-fidelity speech recognition systems (Hinton et al. (2012)), models which achieve state-of-the-art performance in language translation (Bahdanau et al. (2014)) and models which accurately model the human voice for text-to-speech systems (Van Den Oord et al. (2016)).

In investigating the efficacy of RNNs for neuronal time-series modelling, chapter 3 measured the performance of simple RNNs with two memory-augmented recurrent models: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) for modelling the presynaptic responses of the fast extensor tibiae (FETi) motor neuron. Of these, the memoryless RNN proved the most successful, achieving the lowest mean-squared errors on predicting the neuronal responses for unseen test data. This is in contrast to trends in the machine learning literature, which favour LSTMs and GRUs over the simple RNN (Hinton et al. (2012)). Chapter 3 showed that the
Chapter 5 Locust neuronal system identification with Recurrent Neural Networks: the Slow Extensor Tibiae motor neuron

Figure 5.19: Typical responses from RNN models trained on SETi data on a half-wave rectified sinusoid biased toward flexion. (a) Returning the tibial angle to neutral after extension results in high spiking probability immediately after the return. (b) The spiking probability pattern persists even if the return is into extension. (c) An illustration of how the output spiking probability varies with where the tibia is returned to after extension. Here, 0 corresponds to a return to a neutral tibial angle and 1 corresponds to pure tibial extension. (d) An illustration of how the output spiking probability varies with the velocity of return from extension. Here, the spiking probability is plotted against frequency of the half-wave rectified stimuli.

Figure 5.20: Typical model responses to a step stimuli: (a) A step towards full flexion shows a sustained output of high spiking probability. (b) A step toward full extension shows no response from the model.
neuronal dynamics of FETi were relatively simple, with predominantly linear dynamics. The type of data explored in the machine learning literature on the other hand are far more complex, nonlinear and include long-ranging memory effects which can cause instabilities if modelled using simple RNNs.

In particular, Bengio et al. (1994) showed that using RNNs on complex datasets can lead to vanishing gradients: a phenomena where any learning signal present in the data degrades over many timesteps, stalling model training. Gers et al. (1999) also empirically showed that memory-augmented LSTMs are far better at modelling such complex data than RNNs. In contrast, the FETi neuron dynamics were shown to be relatively short term: an input stimulus produced a response within 20ms. With short-term dynamics, RNNs are unlikely to suffer from the vanishing gradient problem. Bolstered by their success at the FETi modelling task, RNNs were also chosen for the SETi modelling task.

Because the SETi neuron is a counterpart to the FETi neuron (Burrows (1996)), the quality of their dynamics were assumed to be similar: both with relatively short-term stimulus-to-response dynamics in the order between tens to hundreds of milliseconds. A major difference between this study and that done in chapter 3 in the nature of the neuronal output. In contrast to the continuous-valued FETi outputs studied in chapter 3, this chapter considers binary-valued spiking outputs from the SETi. The difference in output types means that applying the mean-squared error loss function to train RNNs as discussed in chapter 3 would not work for this task.

In the machine learning literature, binary outputs are framed in terms of emission probabilities. Here, the output of a timeseries is expressed as the probability of emitting a 1 or 0 at every timestep. From a probabilistic perspective, Bishop (2006) explained that a loss function can be expressed in terms of the cross-entropy between the model probability and the data output. Furthermore, LeCun et al. (2015) explained that the cross-entropy loss is a special case of the Kullback-Liebler divergence between two probability distributions. In short, binary-cross entropy is a measure of the divergence between the emission probability distribution output by the model and the binary distribution of the data outputs. Tweaking model parameters to reduce this divergence is therefore equivalent to reducing the disparity between the models dynamics and the dynamics underpinning the processes which produced the data (Bishop (2006); LeCun et al. (2015)). In this study, the cross-entropy function was used to express the discrepancy between RNN predictions and SETi output.

A major obstacle to SETi modelling was an imbalance in the data. Here, the number of time-points with spikes far outnumbered the number of timepoints with no spikes. Japkowicz et al. (2000) showed that in similar cases of data imbalance, models can overfit to a situation where the predictions are no spike for every timestamp. This is because the number of spikes in the data are so outnumbered, that the models cross-entropy error is minimised if it simply predicts no spike at every timestep. In this situation there is no effective learning signal from misclassifying spikes as no-spikes as the cost of false negatives is relatively miniscule. The overfitting phenomenon was observed very early in the preliminary stages of the SETi modelling experiments.

There are a number of works in the literature dedicated to solving the data imbalance problem (Japkowicz et al. (2000); He and Garcia (2008)). Of these, the cost-function reweighting scheme described by Panchapagesan et al. (2016) was chosen for this chapter. With this scheme, the cross-entropy loss function was reweighted to penalise false negatives far more compared to false
This meant that models settling into a state of predicting no spike at every timestep were largely penalised for any false negatives where no spikes were predicted at points where spikes occurred. Using this reweighting scheme was shown to work best for the experiments in this study, where the models were forced to learn the underlying spiking dynamics rather than settle into the lazy solution of predicting no spike everywhere. The reweighting scheme vastly improved model accuracy, and forced the models to better generalise to predicting unseen test data.

Another difference between this study and Chapter 3 is the metric used to describe generalisation performance. Chapter 3 used the mean-squared error between model prediction and test data output, the standard performance metric on continuous-valued data. Although the binary cross-entropy loss could be used to describe the discrepancy between model outputs and the outputs in unseen test data, simply stating a scalar cross-entropy value doesn't provide an intuitive explanation of whether that value implies a good or a bad model. Instead, a more meaningful metric could state the proportion of true positives and false positives predicted by the model for a given test dataset. Given an input from the test data, this metric would provide a measure of how well the model predictions for spiking lines up with actual spikes elicited from the SETi by that input.

A problem with providing such a metric is that it is impossible to measure the number of true- and false- positives for spiking from the models because the models output a continuous-valued probability and not a discrete spike at every timestep. It is possible to threshold this probability at a given value between 0 - 1 and accept any timepoints where model outputs are greater than this threshold as spikes. However, the best value for such a threshold is not known beforehand.

An interesting approach in the literature is to plot how the proportion of true positives varies against the proportion of false positives for all thresholds between 0 - 1 (Bradley (1997)). This
procedure results in a two-dimensional graph called the receiver operating characteristic (ROC) curve. In machine learning, ROC curves are used extensively to report on the quality of modelling algorithms (Bradley (1997)). In particular, the area under the ROC curve (area under curve or AUC) is commonly used as a scalar description of the quality of a learning algorithm (Hanley and McNeil (1982); Bradley (1997)). Regardless of algorithm or application, an AUC of 0.5 implies an equal number of true-positives to false-positives from the model. This implies a poor estimator that is no different from random guessing (Bradley (1997)). On the other hand, an AUC of 1 implies a constant true-positive rate of 1, regardless of probability threshold, which is indicative of a perfect classifier (Figure 5.21). In this study, AUC scores were computed from model predictions on unseen test data as a measure of model quality.

RNN models were trained on the test data of each of the 25 SETi datasets collected, and the spread of their AUC scores for each of 3 different test datasets were reported. To provide a baseline against which these scores could be evaluated, a set of experiments were run where RNNs were trained on randomised gaussian white noise (GWN) inputs mapping to SETi data outputs. Because these models were trained to model random dynamics, the AUCs of these models on the test data are good measures for a random classifier. As expected, the AUCs for the randomised models were shown to be centered around 0.5 with tight variances, and the scores of the real RNN models on the test data were significantly higher. Establishing AUC baselines with random classifiers showed that the RNNs trained on SETi data were significantly better than random classifiers.

5.4.2 Biological Interpretation

Confident that the RNNs were learning the underlying dynamics of SETi from data and forming good estimators of SETi function, the next set of experiments was to use these RNNs to derive and characterise SETi behaviour. There has already been work on characterising SETi dynamics using first- and second- order Wiener kernels (Newland and Kondoh (1997a); Dewhirst et al. (2013)). Further work on neuronal modelling of locust motor neurons has shown that Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs) can create higher fidelity estimators of neuronal data than Wiener kernels (Chapter 3, Meruelo et al. (2016)). However, compared to Wiener kernels, ANNs and RNNs trade interpretability for fidelity in that these models can be far more numerous in their parameters in order to enable better modelling of complex time series datasets. The following series of experiments aim to leverage the higher fidelity of RNNs to simulate SETi dynamics on various different input stimuli. The argument in favour of this is that if the characteristics of SETi can be uncovered through simulation, there is greater credibility for using less interpretable models like RNNs on complex neuronal datasets. The benefit of having a body of previous studies on SETi is that the results unveiled from the RNNs can be compared and corroborated against the results of these studies.

First, the spike-triggered averages (STA) of the models were examined. Mathematically, a STA is the average of the stimuli immediately preceding every spike in a spike-train. The STA derived from a GWN stimulus is an unbiased estimator of a neurons receptive field (Paninski (2003); Sharpee et al. (2004)). Unlike traditional approaches which use steps, ramps and sinusoids, GWN stimulates the entire receptive field of the neuron (Dayan et al. (2003)). In the literature, the use of STAs forms the core of a body of work aimed at using information-theoretic approaches
Chapter 5 Locust neuronal system identification with Recurrent Neural Networks: the Slow Extensor Tibiae motor neuron

to neuronal functional characterisation (Pillow and Simoncelli (2006); Davidson et al. (2007); Farina et al. (2005)). STA is especially widely used in investigations of neurons in the visual cortex (Dayan et al. (2003)) because it is particularly easy to stimulate these by presenting GWN stimuli to the visual field in the form of checkerboard patterns with randomised pixels, and averaging the stimuli which cause strong spiking.

Interestingly, the STA is equivalent to the first-order kernel of a Wiener series expansion (Lee and Schetzen (1965)). STAs derived from the RNN models can therefore be compared with the result of Newland and Kondoh (1997a)s work on characterising SETi using Wiener kernels. Jing et al. (2012) showed that various patterns of first-order Wiener kernels can be interpreted as position-, velocity- and acceleration- sensitivity in a neurons receptive field. Querying the RNN models with GWN stimuli and averaging the stimuli immediately preceding spikes from the model showed an STA pattern with a single dominant peak, implying that the SETi is position-sensitive to flexion. The STA also showed that the stimuli eliciting spiking from the models begin approximately 60ms before a spike and end approximately 10ms before a spike. These results, including flexion- and position- sensitivity and stimulus timing corroborates the work of Newland and Kondoh (1997a), who showed that SETi response most to position-dependent synaptic inputs from the locust femoro-chordotonal organ (FeCO), with very similar stimulus characteristics. This conclusion also aligns with the work of Field and Coles (1994) who showed that tibial flexion engages the extensor muscles as a reflex response to return the tibia to its initial position.

The consistency of position-sensitive STAs between all models implies a consistent receptive field of SETi for all the locusts tested. Previous work has shown that the system of neurons controlling tibial muscles in a locust have a negative feedback reflex mechanism to maintain resistance in the tibia (Newland and Kondoh (1997a); Field and Coles (1994)). In this context, it makes sense for the SETi to spike reflexively to position-dependent flexion, regardless of individual. The consistency of functionality in motor neurons between individuals was also observed in the work on the fast extensor tibiae (FETi) neuron outlined in Chapter 3. The consistency of STAs across RNNs trained on SETi data from different individuals shows that the flexion-sensitive dynamics implied by the STAs are consistent across the population.

In this study, the results show that RNNs trained on SETi data from one individual are largely able to predict the SETi dynamics of all other individuals in the sample population. 21 of the 25 trained RNNs showed a spread of performance in predicting the test data across the population. These performance scores were significantly better than the baselines established by random classifiers, showing that these RNNs were good estimators of SETi dynamics across the population, despite being trained on data from a single individual. Marder and Taylor (2011b) and Goldman et al. (2001) suggested that models trained on neuronal data from a single individual may not be representative of the system across the population due to the variability and differences between individuals. Although the prediction performance for each RNN in this experiment was higher for the individual it was trained on, the mean performance across all individuals were approximately equal to the own-data performance for each RNN, indicating that individual RNN models for SETi form good estimators of functionality across the population.

Small differences in the spike-coding properties of SETi between individuals can account for the variance in model performance between individuals. Marder and Taylor (2011b) showed that the
synaptic properties of identified neurons vary across individuals, even for systems with the same neural function. Schneidman et al. (2001) empirically showed that in the fly visual system, approximately 70% of the information encoded by identified neurons is consistent across individuals and the remaining 30% varies across the population. For a motor neuron, consistent performance across individuals is to be expected as these neurons serve the same purpose regardless of individual differences (Burrows (1996)). However, spontaneous activity stemming from interneurons upstream of the motor neuron can be a source of variability (Burrows (1996); Büschges et al. (1994)). Yet another source of error is noise from neighbouring neurons, as was discovered in the data exploration stage. However, the noise from neighbouring spikes was mitigated by clustering and removing non SETi spikes using a preliminary STA analysis of the clusters.

Although the STA is a beneficial tool in deriving the linear characteristics of a spiking neuron, these cannot be used to derive any nonlinear characteristics. Newland and Kondoh (1997a) got around this shortcoming by augmenting their analysis with second-order Wiener kernel coefficients. Wiener kernels can theoretically model arbitrary levels of complex nonlinearities through higher-order kernels (Marmarelis (2004)). However, in practice it can be difficult to train these models to converge to good estimators with third- and higher-order kernels on highly nonlinear data (Tötterman and Toivonen (2009); Palm and Poggio (1977a)). On the other hand, training RNNs to model nonlinear data is trivial. In practice simply stacking hidden layers allows RNNs to converge to good estimators (LeCun et al. (2015)). Additionally, expanding RNNs to encompass greater complexity requires no changes to the underlying learning algorithm, and convergence is only a matter of choosing good hyperparameters (LeCun et al. (2015); Curtis and Scheinberg (2017)).

In this study, RNNs were queried for any indication of nonlinear behaviour by querying the models with a range of inputs. The most insightful results were obtained when the RNNs were queried with half-wave rectified sinusoids at various frequencies. Half-wave sinusoids in the flexion regime showed strong spiking responses from the models, in agreement with the flexion-position-sensitivity implied by the models STA. Reducing the magnitude of the input half-waves showed that this activation only occurred past a certain threshold in normalised tibial angle, implying that the neutral tibial angle is 0.1% in the flexion regime for the angular range defined in these experiments.

Interesting nonlinear behaviour was observed when the RNNs were queried with half-wave sinusoids in the extension regime. Here, any return of the tibia from full extension elicited high spiking probabilities from the models. Furthermore, this phenomena was only observed at higher frequencies. These results show that SETi may be sensitive to velocity, a result not indicated by the models STA. Interestingly, Newland and Kondoh (1997a)’s work suggested that SETi is largely position-sensitive with a very small velocity-sensitive component. However the results in this study suggests that SETi shows a strong velocity-sensitive component, particularly when the tibia is rapidly moved from extension towards flexion and suddenly halted. Mechanically, this suggests that abruptly removing the external stimulus as the tibia is rapidly moving toward flexion engages the extensor to halt further movement into flexion.

A strong point to be noted is that the models were queried with frequencies above the cutoff bandwidth of the GWN stimuli they were trained on. Any results for frequencies beyond the
27 Hz GWN frequency cutoff may not be representative of SETi responses, simply because the models were not trained on SETi behaviour beyond this range. The most effective way to corroborate the results of this analysis would be to query SETi with another set of electrophysiological experiments, although this was not done in this work.

For all the simulation experiments described in this chapter, additional tests were performed to measure any potential differences between RNNs trained on open- and closed-loop data. All of the studies in motor neuron function of the locust hind leg use an open-loop experimental setup: where the locust tibia is held fixed, and the apodeme connected to the tibia is stimulated as a proxy for tibial movement (Meruelo et al. (2016); Dewhirst et al. (2013); Newland and Kondoh (1997a)). In the closed-loop setup, the apodeme is also stimulated as a proxy for tibial movement, however the tibia is allowed to move freely, potentially introducing additional dynamics into the setup which can confound analysis.

Comparing the spread of AUC scores attained by the models showed that the closed-loop models attained higher median AUCs with tighter interquartile ranges than the open-loop models. Four of the models trained in closed-loop did not achieve AUC scores significantly different from random classifiers. This result implies that RNNs form better estimators of closed-loop SETi dynamics. Comparing the STAs of the two classes of models showed no qualitative difference. Both open- and closed-loop models demonstrated STAs with similar amplitudes and a large flexion-sensitive peak. Models trained in closed-loop to predict responses of all other SETi data showed higher median AUC scores than open-loop models predicting responses of other SETi data. Comparing the AUCs between the open- and closed-loops showed that neither result was significantly different from the other (Friedman test, \( p > 0.025 \)). Finally, in simulating the RNN dynamics with half-wave rectified sinusoids, there were small differences in the amplitude of output probabilities of all models, regardless of open- or closed-loop, which can be attributed to individual differences. Functionally, the outputs of both open- and closed-loop models were the same. Both showed sustained spiking as a response to input flexion. Additionally, both showed spiking to high-frequency half-wave sinusoids from extension to flexion. Both in linear and nonlinear functional characteristics, no difference was observed between the open- and closed-loop models.

In short, although models trained in closed-loop performed better in most tests for performance compared to open-loop, the difference was marginal and not significant. Additionally, in tests of functionality using STA and simulation, there was no difference in SETi functionality implied by either model. This result suggests that all the previous studies on locust neuronal modelling which used open-loop setups to derive experimental data were justified in their choice.

### 5.5 Conclusion

The results showed that RNNs perform well at the SETi modelling task. The techniques developed in Chapter 4 for modelling spiking neuronal responses as discrete events proved successful at the modelling task in this chapter. The models were trained on data using Gaussian white noise (GWN) synthetic stimuli. Testing the trained models on SETi responses to natural walking stimuli showed that the models form high-fidelity estimators of neuronal function. The evidence for this was presented using AUC metrics, demonstrating significantly better performance than
random classifiers. This result established the generalisation ability of the RNNs, allowing these to be simulated and queried using more interesting signals.

In short, this chapter established that RNN models trained on synthetic SETi data can be used to simulate a range of input stimuli. The corresponding responses can be analysed to gain insight into SETi behaviour. The results of this simulation analysis showed that there is no significant difference between SETi experiments in open-loop vs experiments in closed-loop. Evidence for the models' generalisation ability was provided, lending credence to the validity of the simulation analysis. It would be best to confirm the results of this analysis with further electrophysiological experiments, although this was not done in this work.
Chapter 6

Discussion and Conclusion

The overarching ambition of this thesis as set out in Chapter 1, was to test the efficacy of using Recurrent Neural Networks (RNNs) for the system identification (SI) of neuronal systems. The argument for using RNNs, as laid out in Chapter 1, is that RNNs may be better capable of modelling the underlying dynamics of neuronal systems than more traditional modelling techniques like Wiener/Volterra kernel expansions or Linear-Nonlinear-Linear (LNL) methods. This argument is aided by the ongoing presentation of the capability of RNNs in modelling complex data in a variety of tasks in the machine learning literature (LeCun et al. (2015); Goodfellow et al. (2016)).

The strongest limitation of this argument is that RNNs are capable of modelling very complex datasets, but only with very large datasets. This is a limitation with artificial neural networks (ANNs) in general. In the machine learning literature, the need for large datasets to make ANNs productive relegated these models in the background compared to less powerful techniques like support-vector machines (SVMs) which were more effective than ANNs on smaller datasets (LeCun et al. (2015); Goodfellow et al. (2016)). Only with a recent availability of large amounts of computation and a corresponding rise in the size of available datasets have ANNs become more popular than other techniques, as the structure of these models can easily be extended to learn encompass arbitrary levels of complexity (Goodfellow et al. (2016)).

Given this context, the work in this thesis did not use large neurophysiological datasets of the scale usually presented in the machine learning literature. Instead, RNNs were used in modelling a series of neuronal systems from a much simpler nervous system in the form of locust motor and sensory proprioceptive neurons. The reasons cited for this choice were: the locust nervous system provided a scaled-down version of more complex nervous systems and that a variety of previous studies on the locust nervous system were already available for corroboration. The datasets derived from the neurons studied in this work were arguably simple and not as complex as datasets which could be derived from more complex nervous systems.

Nevertheless, having simpler datasets with varied sources of background work for corroboration meant that the RNN models could be developed and validated easily. This was a bonus given the lack of previous work on using RNNs in neurophysiological modelling. In fact, despite being
trained on small datasets, RNNs showed significant results in modelling and generalisation on these datasets.

The reasoning for testing the efficacy of RNNs in neuronal modelling was laid out in three stages and separated by chapter. Chapter 2 presented a background of the problems associated with training RNNs in the general machine learning context. Chapter 3 compared the quality of RNN models with previously used models on the FETi modelling task. Chapter 4 developed techniques for modelling and evaluating RNN models on spiking neuron data. Chapter 5 extended these techniques for training on synthetic data derived from a spiking neuron; tested the models on natural stimuli to provide measures for the generalisation performance of the models; and used a variety of stimuli to derive insights into neuronal behaviour by querying the trained models.

6.1 The FETi modelling task

The data presented in Chapter 3 was unique in that these were graded responses from the locust Fast Extensor Tibiae (FETi) motor neuron in response to wideband (27Hz) Gaussian white noise (GWN) stimuli. Dewhirst et al. (2013) used the same data to model FETi dynamics and used a Linear-Nonlinear-Linear model for the task. Meruelo et al. (2016) also used this data for modelling FETi dynamics using Time-Delay Neural Networks (TDNNs). Chapter 3 therefore set out to model this FETi data using RNNs, and compared the RNN performance with LNL and TDNN performance. RNN performance was also compared with Long Short-Term Memory models (LSTMs) and Gated Recurrent Unit models (GRUs) to accommodate for long-range memory effects in FETi dynamics. Additionally, Chapter 3 aimed to provide some analysis of the functionality of FETi as implied by the trained RNNs, and compared this with the results of previous studies on FETi.

Here, the modelling was framed as a continuous-valued regression task. The models were trained in mapping GWN inputs to the corresponding graded FETi activations. Preliminary passes at modelling were largely unsuccessful, and successful models were only derived after a thorough search for the right combination of hyperparameters. A similar effect was observed in Meruelo et al. (2016)’s work using TDNNs. The trained models were tested on a 30% held out test set and demonstrated significantly lower normalised mean-square errors (NMSE) than the previously used LNL models. Additionally, the NMSE scores on the test set were similar to TDNNs. Among the trained RNNs, LSTMs and GRUs, the RNNs had the highest performance (lowest NMSEs) across all animals. The insight drawn from this result was that FETi data may not have any long-range memory effects. Testing data for memory effects using RNNs and comparing the modelling results with memory-equipped equivalents like LSTMs is not standard practice in the literature. Relatively poorer performance by LSTMs and GRUs is not necessarily an indication that there is no long-range memory in the dynamics of the system being modelled. However, in the case of a motor neuron, this is a reasonable assumption. Functionally, a motor neuron does not need to exhibit hysteresis, especially in reflex responses, where it only responds to stimuli from the recent past. Of course, if the FETi system did exhibit hysteresis, RNN performance would be expected to be much lower than LSTM and GRU. Although from this standpoint, a lot more data would need to be collected on the hysteresis phenomenon than was available in
these experiments for the LSTM and GRU models to converge to good estimators of the system dynamics.

In modelling FETi responses to 5 Hz stimuli, the RNN models demonstrated much higher NMSEs (lower performance) than LNLs and TDNN models. However, a visual examination of the RNN outputs showed that their responses to 5Hz stimuli closely mirrored true FETi responses, with the exception of high-frequency variation, which was the source of the higher NMSEs. This result reinforced a point that simply observing final metrics from a model may not yield the full picture. Any post-training analysis must be accompanied by a visual analysis of model responses. The dynamics observed from the trained RNNs mirrored the conclusions of Newland and Kondoh (1997a)’s previous work on FETi dynamics.

6.2 The proprioceptive sensory afferent modelling task

Chapter 4 set out to test the efficacy of RNNs in modelling the dynamics of the proprioceptive sensory afferent. Unlike the FETi data in Chapter 3, the data presented in this chapter contained spiking action potentials instead of graded potentials, as can be expected from a majority of neurons in any nervous system. The approach in this chapter took a spike-based as opposed to a spike-rate approach to modelling the sensory afferents, as the spike-rate modelling approach proved unsuccessful in preliminary experiments. As a result, the modelling was framed as a categorical problem of modelling the probability of spiking/non-spiking from the afferent at every discrete timestep, given all previous sensory stimuli. Because there was no similar prior work on using RNNs to model spiking neurophysiological data as a categorical problem, the focus of this chapter was predominantly on the machine learning and data preprocessing aspects of the problem.

In preliminary experiments, getting high performing RNN estimators of the sensory afferent data provided was a difficult challenge. Specifically, the data was noisy, with spikes from neighbouring neurons present in a single afferent’s recordings. Additionally, there were artifacts like slow time-varying drifts which made discretising the spikes for the categorical modelling task difficult. The neighbouring spiking problem was mitigated by clustering spikes based on features which were visually apparent to distinguish the spikes: spike amplitude and time between previous spike. The time-varying drift effect was removed using a singular-spectrum method to remove the low-frequency drift components whilst retaining the high-frequency spikes. In comparison, using low-pass bandwidth filters were less successful as these removed distinguishing features of many spikes in the data, with a consequence of increasing data sparsity.

Another difficulty with deriving good estimators was the data imbalance problem. A very small proportion of the total data resulted in spikes (1% - 5%). This meant that estimators which predicted ‘no-spike’ responses to all stimuli were 95% to 99% accurate in their predictions. This also meant that the frequency of false negatives was high, resulting in low AUC scores. High accuracy and low AUC is a typical indicator of such data imbalance. The data imbalance was solved by a weighted cross-entropy scheme, where the false negative terms were weighted by the inverse proportion of positives in the data, making the cost for missing a positive much larger.
Upon adequately preprocessing the data, and performing hyperparameter search for the best combination of hyperparameters, the RNN models began converging to good estimators on validation data. Keeping with the machine learning literature on binary categorical modelling, area under receiver operating characteristic (AUC) metrics were used to report on the generalisation ability of the models on unseen test data. Unfortunately, there was no previous modelling work that framed modelling the sensory afferents as a categorical problem. Thus, there were no baselines against which AUC metrics could be compared. Kondoh et al. (1995) modelled sensory afferents using Wiener/Volterra methods, but framed their modelling task as a continuous-valued problem.

To establish baselines against which the test set AUC metrics could be compared, RNNs were trained on randomised data where the inputs were freshly generated GWN stimuli, and the original responses from the afferents were retained. By obscuring the afferent dynamics in this way, baselines could be obtained for what were effectively random classifiers. The spread of test-set AUCs for these random classifiers were obtained, demonstrating that the RNNs trained on the original GWN data formed significantly better estimators against the random baselines. The implied sensory afferent functionality from the trained models closely matched the results observed by Kondoh et al. (1995).

### 6.3 The SETi modelling task

The aim of Chapter 5 was to model the spiking dynamics of the Slow Extensor Tibiae (SETi) motor neuron using RNNs. Specifically, this chapter sought to use RNNs to model SETi data using synthetic GWN stimuli, and to test the trained models using more natural stimuli, in this case locust walking gaits. In contrast with the previous chapter on sensory afferents, where generalisation performance was established using a GWN test set, using natural stimuli is arguably a better test of model generalisation. Here, the AUC metrics are an indicator of RNN predictive performance on how SETi may perform in a more natural setting than the synthetic stimuli the models were trained on. As in Chapter 4, baselines were also provided using randomised GWN inputs. The RNNs trained on true GWN data showed significantly higher AUC scores than random classifiers. The results showed that RNNs form good estimators of SETi data when the data imbalance problem is solved (using weighted cross-entropy) and proper model hyperparameters are chosen.

Additionally, this chapter explored the implied dynamics of SETi by simulating the models with simple sinusoidal and half-wave inputs. The results showed that SETi responds strongly to flexion, corroborating Newland and Kondoh (1997a)’s observations of SETi’s reflex response to flexion. Building on this result, the models showed that the probability of spiking increases by increasing the magnitude of the input sinusoids, but up to a limit, where the probability saturates. Additionally, simulating the SETi with extension half wave sinusoids showed that the models’ spiking probability increases strongly when the tibia returns to a neutral position of 60° after a period of extension, a result not observed in prior work. This work also showed that the probability of spiking upon a return to neutral position increases when the frequency of the half-wave sinusoids is increased, where the probability saturates at approximately 27 Hz. It
should be noted that the models were trained on 27 Hz bandlimited GWN and that these results may not be an accurate reflection of SETi behaviour past the 27 Hz frequency range.

Finally, all experiments for modelling were done on data extracted from locusts in open- and closed-loop setups. There is no literature comparing the positives and negatives of either setup for locust neuronal modelling, but the open-loop setup is generally used in all cases Newland and Kondoh (1997a); Dewhirst et al. (2013); Kondoh et al. (1995); Meruelo et al. (2016). The work in this chapter showed that in modelling SETi dynamics there was no significant difference between open- and closed-loop setups.

### 6.4 Conclusion and future work

The results of the chapters presented in thesis conclude that RNNs are successful at the neuronal modelling task. As pointed out before, the data used in this thesis were from three different neuronal systems, but were much smaller in scale than the datasets typically reported in machine learning literature (LeCun et al. (2015); Goodfellow et al. (2016)). Nevertheless, the success of RNNs on the tasks in this chapter lends confidence to their use in the system identification and modelling of more complex datasets like human fMRI scans.

Crucially, both graded potentials and action potentials were modelled in this thesis, using regression and categorical/classification setups respectively. Regardless of the type of neuronal data, the issues highlighted in this work are universally applicable in all cases. In particular, the issues involve: ensuring the data is relatively noise-free and only contains data with the important features of the system to be identified; Ensuring the data is balanced with balanced proportions of positive to negative samples, or using augmented loss functions if acquiring balanced data is impossible; Training the models on data that is likely to stimulate the system within its entire operating range, typically with some synthetic data like Gaussian white noise (GWN); and testing the models with natural stimuli to ensure generalisation or establishing good baselines if no data with natural stimuli is available.

One of the downsides of this thesis is that the implications of neuronal behaviour implied by the RNNs was not tested by going back to the biology and corroborating with neuronal behaviour in-vivo. An immediate future application is to compare interesting results from the model with the actual neuronal system through further neurological experiments. One of the advantages of training RNN models for neuronal system identification is that an initial set of RNN models can be trained on a synthetic, broadband, exploratory stimulus set like GWN. Interesting or anomalous implications from the models can then be further explored by querying the neuronal system and further modelling. Using this procedure, researchers can refine their understanding of a complex neuronal system in the vein of work done in optimal stimulus design (Paninski et al. (2007); Sahani and Linden (2003)).
Appendix A

Appendix A: Hyperparameters for Fast Extensor Tibiae motor neuron experiments

This appendix presents the list of final hyperparameters used to prepare data, initialise models and train models on the data. These hyperparameters were the result of using random search to find optimal hyperparameters on a single dataset.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window length</td>
<td>250 points (0.5s @ 500Hz)</td>
</tr>
<tr>
<td>Window overlap</td>
<td>240 points (0.48s @ 500Hz)</td>
</tr>
<tr>
<td>Training data%</td>
<td>60</td>
</tr>
<tr>
<td>Validation data%</td>
<td>10</td>
</tr>
<tr>
<td>Test data%</td>
<td>30</td>
</tr>
</tbody>
</table>

Table A.1: Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. *Window length* and *Window overlap* were parameters of the windowing scheme discussed in Chapter 3. *Train %, Validation % and Test %* are the proportions of data assigned to training data, validation data and test data respectively.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>RNN</th>
<th>LSTM</th>
<th>GRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hidden layers</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hidden units per layer</td>
<td>32</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Hidden layer activation function</td>
<td>tanh</td>
<td>tanh</td>
<td>tanh</td>
</tr>
<tr>
<td>Output layer activation function</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td>Network initialisation procedure</td>
<td>Glorot</td>
<td>Glorot</td>
<td>Glorot</td>
</tr>
<tr>
<td>Dropout probability</td>
<td>0.1</td>
<td>0.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table A.2: Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on FETi data. *Number of hidden layers* denotes the number of intermediate layers between input and output for the networks. *Hidden units per layer* denotes the number of neurons in the intermediate layer(s). *Output layer activation function* denotes the nonlinear activation function used in the hidden layer(s). *Network initialisation procedure* denotes the network initialisation procedure (see Chapter 3). *Dropout probability* denotes the extent of dropout applied to the network as regularisation.
Appendix A: Hyperparameters for Fast Extensor Tibiae motor neuron experiments

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>RNN</th>
<th>LSTM</th>
<th>GRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>1e-3</td>
<td>1e-3</td>
<td>1e-3</td>
</tr>
<tr>
<td>Alpha</td>
<td>0.7</td>
<td>0.65</td>
<td>0.7</td>
</tr>
<tr>
<td>Number of training epochs</td>
<td>30</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>Minibatch size</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Table A.3: Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. Learning rate denotes the size of the gradient step per iteration used during optimisation. Alpha is an Adam-specific parameter used to denote the momentum used during gradient descent. Number of training epochs is the number of epochs used in training; where each epoch is a complete iteration through the dataset. Minibatch size denotes the size of each minibatch used to estimate stochastic gradient.
Appendix B

Appendix B: Hyperparameters for sensory afferent experiments

This appendix presents the list of final hyperparameters used to prepare data, initialise models and train models on the data. These hyperparameters were the result of using random search to find optimal hyperparameters on a single dataset.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window length</td>
<td>180 points (0.18s @ 1000Hz)</td>
</tr>
<tr>
<td>Window overlap</td>
<td>20 points (0.02s @ 1000Hz)</td>
</tr>
<tr>
<td>Threshold for grouping spike points</td>
<td>0</td>
</tr>
<tr>
<td>Training data%</td>
<td>60</td>
</tr>
<tr>
<td>Validation data%</td>
<td>10</td>
</tr>
<tr>
<td>Test data%</td>
<td>30</td>
</tr>
</tbody>
</table>

Table B.1: Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. Window length and Window overlap were parameters of the windowing scheme discussed in Chapter 3. Threshold for grouping spike points denotes the threshold used for the normalised output amplitude beyond which points were grouped together as a single spike for discretisation. Train %, Validation % and Test % are the proportions of data assigned to training data, validation data and test data respectively.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hidden layers</td>
<td>1</td>
</tr>
<tr>
<td>Hidden units per layer</td>
<td>64</td>
</tr>
<tr>
<td>Hidden layer activation function</td>
<td>tanh</td>
</tr>
<tr>
<td>Output layer activation function</td>
<td>logistic sigmoid</td>
</tr>
<tr>
<td>Network initialisation procedure</td>
<td>Glorot</td>
</tr>
<tr>
<td>Dropout probability</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table B.2: Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on sensory afferent data. Number of hidden layers denotes the number of intermediate layers between input and output for the networks. Hidden units per layer denotes the number of neurons in the intermediate layer(s). Output layer activation function denotes the nonlinear activation function used in the hidden layer(s). Network initialisation procedure denotes the network initialisation procedure (see Chapter 3). Dropout probability denotes the extent of dropout applied to the network as regularisation.


<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>Alpha</td>
<td>0.5</td>
</tr>
<tr>
<td>Number of training epochs</td>
<td>40</td>
</tr>
<tr>
<td>Minibatch size</td>
<td>8</td>
</tr>
</tbody>
</table>

Table B.3: Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. *Learning rate* denotes the size of the gradient step per iteration used during optimisation. *Alpha* is an Adam-specific parameter used to denote the momentum used during gradient descent. *Number of training epochs* is the number of epochs used in training; where each epoch is a complete iteration through the dataset. *Minibatch size* denotes the size of each minibatch used to evaluate stochastic gradients at every iteration.
Appendix C

Appendix C: Hyperparameters for SETi experiments

This appendix presents the list of final hyperparameters used to prepare data, initialise models and train models on the data. These hyperparameters were the result of using random search to find optimal hyperparameters on a single dataset.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window length</td>
<td>400 points (0.4s @ 1000Hz)</td>
</tr>
<tr>
<td>Window overlap</td>
<td>200 points (0.2s @ 1000Hz)</td>
</tr>
<tr>
<td>Threshold for grouping spike points</td>
<td>0.</td>
</tr>
<tr>
<td>Training data%</td>
<td>90</td>
</tr>
<tr>
<td>Validation data%</td>
<td>10</td>
</tr>
<tr>
<td>Test data%</td>
<td>0</td>
</tr>
</tbody>
</table>

Table C.1: Data hyperparameters: These hyperparameters were used to prepare the data before training any models on the data. Window length and Window overlap were parameters of the windowing scheme discussed in Chapter 3. Threshold for grouping spike points denotes the threshold used for the normalised output amplitude beyond which points were grouped together as a single spike for discretisation. Train %, Validation % and Test % are the proportions of data assigned to training data, validation data and test data respectively. The test data was held at 0% as the models were tested against natural stimuli from walking gaits.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hidden layers</td>
<td>1</td>
</tr>
<tr>
<td>Hidden units per layer</td>
<td>32</td>
</tr>
<tr>
<td>Hidden layer activation function</td>
<td>tanh</td>
</tr>
<tr>
<td>Output layer activation function</td>
<td>logistic sigmoid</td>
</tr>
<tr>
<td>Network initialisation procedure</td>
<td>Glorot</td>
</tr>
<tr>
<td>Dropout probability</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table C.2: Model hyperparameters: These hyperparameters were used to determine the structure of the RNN, GRU and LSTM models trained on SETi data. Number of hidden layers denotes the number of intermediate layers between input and output for the networks. Hidden units per layer denotes the number of neurons in the intermediate layer(s). Output layer activation function denotes the nonlinear activation function used in the hidden layer(s). Network initialisation procedure denotes the network initialisation procedure (see Chapter 3). Dropout probability denotes the extent of dropout applied to the network as regularisation.
<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>Alpha</td>
<td>0.65</td>
</tr>
<tr>
<td>Number of training epochs</td>
<td>20</td>
</tr>
<tr>
<td>Minibatch size</td>
<td>8</td>
</tr>
</tbody>
</table>

Table C.3: Optimiser hyperparameters: These hyperparameters were used for training the models using stochastic gradient descent. In all cases, the Adam algorithm was used. Learning rate denotes the size of the gradient step per iteration used during optimisation. Alpha is an Adam-specific parameter used to denote the momentum used during gradient descent. Number of training epochs is the number of epochs used in training; where each epoch is a complete iteration through the dataset. Minibatch size denotes the size of each minibatch used to evaluate stochastic gradients at every iteration.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


