We Have Bananas: Decoding with Neural Networks

Tiffany Hamstreet, Ahmet Narman, Junke Yao and Luxi Zhang
(We Have Bananas Team)
Dept. of Bioengineering, Imperial College London, London, UK
e-mails: tah18@ic.ac.uk, an3818@ic.ac.uk, jy2418@ic.ac.uk, lz3518@ic.ac.uk

***Abstract*—A feedforward neural network was used to predict the change in a monkey's hand trajectory from cortical spiking data delivered in 20 ms blocks. Complete predicted trajectories were assembled from each predicted position change and the prior position estimates. Extensive optimisation was conducted using cross validation resulting in a model with 4 hidden layers with exponential rectified linear (ELU) activation and back-propagation with the Adam algorithm. Prediction accuracy was measured with root mean squared error (RMSE) as compared to actual trajectories. The final model placed second in the competition with RMSE of 11.16cm on the testing data.**

I. Introduction

The field of neuroprosthetics relies on our ability to decode motor and sensory information to predict intention and ultimately control a prosthetic. Many studies, similar to this one conducted by Professor Krishna Shenoy at Stanford University, have been conducted with invasive cortical electrodes in animals engaging in kinematic activity (1). Classical approaches to this problem include the population vector method combined with multiple regression (2), principal component analysis (PCA) combined with neural networks, multiple regression alone, and Kalman filtering (1). More recently, machine learning techniques have been applied to this problem. For example, (3) decoded cortical signals from three different brain areas with eleven methods, including modern machine learning techniques such as Long-Short Term Memory (LSTM) recurrent neural networks, feedforward (FNN), and ensemble networks and classical techniques such as Kalman and Weiner filtering. Of these, the three networks mentioned produced much higher prediction accuracies with less training data. With the maximum amount of motor cortex training data tested, the FNN and LSTM networks had R^2 values of approximately 0.85 and 0.83 respectively as compared to 0.72 for the Kalman filter.

A FNN was selected for the current application due to its simple implementation and relatively high prediction accuracy. Several architectures were tested and parameters and hyperparameters were optimised over extensive cross-validation trials using the training data provided. The final model achieved second place in the competition with root mean squared error (RMSE) of 11.16cm.

II. Preliminary work

The data was explored using several methods of visualising neural information including raster plots, peristimulus-time histograms (PSTH) and tuning curves. Figure 1 shows a sample (a) raster plot and (b) PSTH over all 100 trials for a single neuron channel over the 150 degree trajectory angle. The raster plot shows all individual spike times over all trials, and the Gaussian smoothing function fitted to the PSTH provides a continuous estimate of firing rate (4). The PSTH gives the firing rate as spike density over many trials according to Equation 1 where K is the number of trials, t is time and Δt is the size of each time bin in the histogram (5).

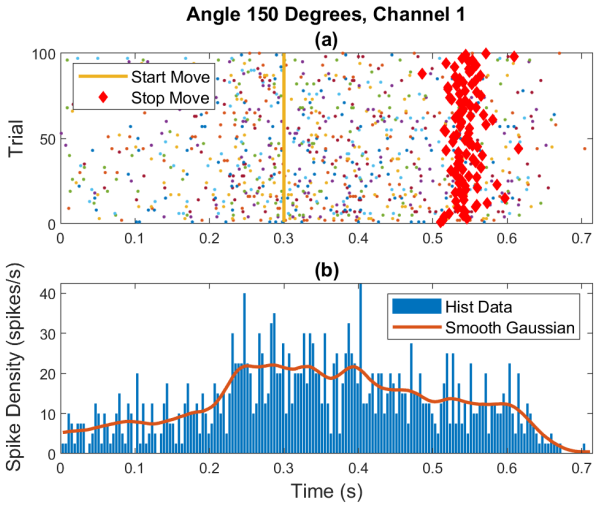

Fig. 1: (a) shows a raster plot and (b) a PSTH for the first neuron channel over all trials for the a trajectory angle of 150 degrees. In (b), each bin contain 4ms of data and Matlabs Gaussian smoothing function, windowed over 5 bins (20ms), provides an estimate of firing rate as spike density over all 100 trials.

$$\rho = \frac{1}{\Delta t}\frac{1}{K} n_K(t; t + \Delta t) \tag{1}$$

There is an anticipatory phenomenon present in Fig. 1 in which spiking activity begins at around 220 ms, prior to the onset of movement at 300 ms shown by the yellow line in (a). Spiking activity also appears to slow in anticipation of the end of movement. While this anticipatory behaviour is observed in many neuron channels, it is not true of all, as evidenced by a plot of the 28th neuron channel for the same trajectory angle with no anticipatory firing, shown in Fig. A7. For further exploration, the spiking activity of the entire population of 98 neurons was plotted in a PSTH for each trajectory. Figure A8 shows that average spiking activity approximately doubles in a short time interval within the first 300 ms, prior to the onset of movement. Using a threshold for the magnitude of change between time steps, the jump was found to occur around 151

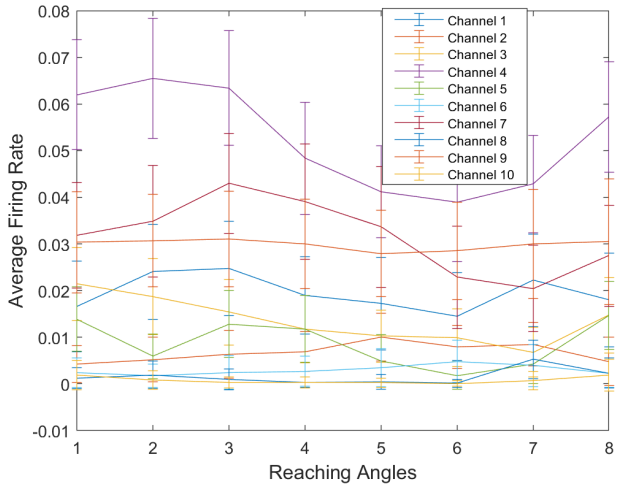

Fig. 2: Tuning curves with standard deviation for all trajectory angles across all trials

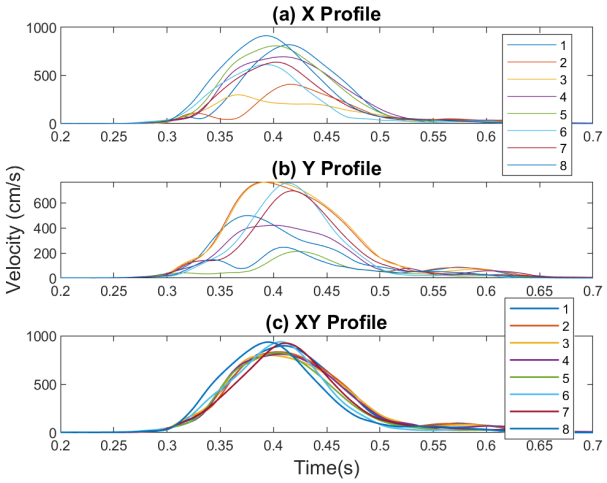

Fig. 3: Mean velocity profiles are shown for all 8 trajectory angles. The x and y component velocities shown in (a) and (b) show Gaussinity with different variance and amplitude, whereas the combined x and y velocity profiles very clearly show Gaussinity with consistent amplitude and variance for all trajectories.

+/- 10 ms for all trajectories. This jump is believed to signify anticipatory/planning activity where the stimulus is present but movement has not begun. As such, spikes prior to this jump are assumed to be random noise or generated by other causes not relevant to the movement.

Directional preference for individual neurons can be visualised with tuning curves which show the neurons average firing rate in the presence of a certain stimulus, in this case a movement angle. Figure 2 shows example tuning curves for 10 neuron channels, calculated over all trials for the duration of movement as described in the pre-competition homework. It is observed that some neural units, such as channel 4, show sensitivities and preferences for different reaching angles, and other neural units, such as channel 6, do not show clear directional preference.

The population vector method was explored during early work, revealing some interesting characteristics in the data. The first step in the population vector method as described by (2) is to use Analysis of Variance (ANOVA) to reduce the dimensionality of the data by identifying neurons whose firing rate does not significantly vary between trajectory angles, and thus does not contribute useful information. At a p-value of 0.05, the following 5 neuron channels were found not to be directionally selective during the duration of movement: channels 9, 15, 38, 44, and 49. Figure A9 shows a smoothed PSTH for each of these neurons over all 8 trajectory angles. It is observed that firing rates for these channels are extremely similar for each of the 8 trajectories during the movement, showing little to no directional selectivity.

Mean velocity profiles were calculated for all trajectory angles as shown in Fig. 3. While x and y velocity profiles in (a) and (b) appear to be Gaussian, of different amplitude and variance, the directional velocity including both x and y components shown in (c) shows clear Gaussinity with very consistent amplitude and variance.

III. FF Neural Network

After exploration of the data and research into several decoding methods, a FNN was selected as our position estimator model. It was based on the FNN model described in (3) and adjusted and optimised to fit our problem. In our task, the first 320 ms of neural data were initially provided followed by 20 ms data blocks until the end of each trial. Therefore, a model was needed to determine the change in hand position over each 20 ms block. The trajectory could then be constructed recursively from each predicted position and the change over the following time block. The FNN architecture, training procedure, and testing procedure were designed with this in mind.

For the processing of each data block, the raw spike train data was first converted into a form that could be used by the FNN more effectively. The data in a window from the current time to W time steps past (200-300 ms) was taken and separated into bins of size B (20-50 ms) in which the spikes were summed. This operation yields a feature vector of size N*(W/B) where N is the number of neurons and each feature is the sum of spiking activity in a specific time bin for a specific neuron. As such, W time steps in total were used to determine the change in x and y positions over each 20 ms block.

In a neural network, the output of each neuron is given by

$$\begin{aligned} \mathbf{Y} &= \mathbf{wX} + \mathbf{b} \quad \text{all layers} \\ \mathbf{O} &= f(\mathbf{Y}) \quad \text{hidden layers} \end{aligned} \tag{2}$$

where $\mathbf{w}$ and $\mathbf{b}$ are weights and bias, $\mathbf{X}$ is input, $\mathbf{Y}$ is initial output, and $\mathbf{O}$ is the final output after a nonlinear activation function is applied. Activation functions allow modelling of nonlinear problems. The final architecture after parameter optimisation is shown in Fig. 4. Feature vectors are fed through

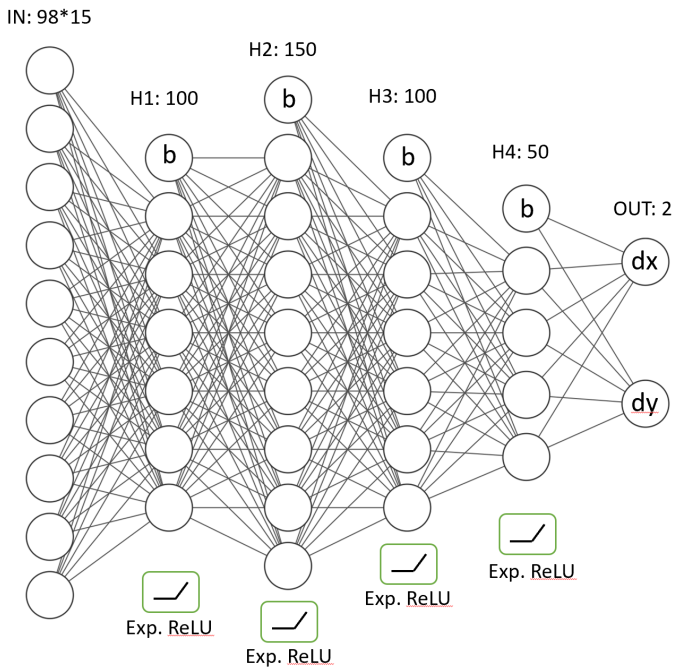

Fig. 4: The final FNN architecture. W=300 ms and B=20ms was used which gave 15 features per neuron. 4 hidden layers had 100, 150, 100 and 50 neurons respectively with the bias terms b. Exponential ReLU was used as the activation function in the hidden layers.

all layers with exponential rectified linear activation functions (ReLU) on hidden layers. Bias terms are fed to the hidden layers and output layer. The final network outputs are the change in x position and y position in 20 ms.

In training, the initial connection weights are randomly assigned small numbers according to a normal distribution with zero mean and standard deviation of 0.1. The training data are fed to the FNN in mini-batches and the corresponding outputs, which are the predicted changes in position, are found. Using the actual position changes as desired outputs, the mean squared error (MSE) was calculated and the desired weight changes were found by backpropagation, including testing of stochastic gradient descent and ADAM optimisation algorithms. The weight changes were normalised for every mini-batch and then updated using the determined learning rate (and ADAM parameters). Training stops after 50 epochs, where the average MSE was found to converge.

In testing, the window of spiking data corresponding to each 20 ms block was converted to the above specified format and fed to the FNN. The resulting position changes were added to the latest position until the end of motion to estimate the overall trajectory.

IV. Optimisation

Optimisation of neural networks includes selection of the fundamental architecture, backpropagation algorithm and activation functions as well as tuning of hyperparameters. Optimisation was completed iteratively, starting with major architecture changes and ending with fine-tuning of hyperparameters. When one parameter was tested, all others were fixed for direct comparison. 5-fold cross validation was used for all preliminary optimisation and 10-fold cross validation was used for final model selection, using RMSE as the measure of performance. All optimised architecture choices and hyperparameters are shown in Table I. Optimisation figures are shown in the appendix, as referenced in the table.

TABLE I: Architectural choices and hyperparameters optimised for FNN where **Opt** indicates the optimised value

Parameter	Description	Ref
Architecture	tested 2, 3, and 4 hidden layers and number of neurons per layer **Opt: 4-layer; 100, 150, 100, 50 neurons**	Table II
Backprop Algorithm	Gradient descent and Adam optimiser **Opt: Adam**	Equ 4
Adam β_1	Adam with variable learning rate, optimised β_1 **Opt:** $\beta_1 = 0.4$	Fig A10 Equ 3
Activation	nonlinear activation functions tested: ReLu, leaky ReLu, Exponential ReLu (ELU) and tanh were tested **Opt: ELU**	Fig A11 Equ 5
Window Size	amount of past data used at each 20 ms time step **Opt: 300 ms**	n/a
Bin Size	Size of data partitions in which the spiking activity was summed **Opt: 20 ms**	Fig A12
Learning Rate	3 types of learning rate were tested: constant, exponential decay , and Adam (variable) **Opt: Adam with 0.01 initial learning rate**	Table III Equ 3,4

A sample of data from the 5-fold cross validation optimisation trials is shown in Table II, where configurations of the architecture with 4 hidden layers are compared. The optimal configuration is highlighted in yellow.

TABLE II: 5-fold cross validation for optimisation of different architectures with 4 hidden layers. Each architecture is represented by 4 numbers and each number represents the number of neurons in the corresponding hidden layer.

Neurons per Hidden Layer	Mean RMSE	Std
100,150,50,50	13.8357	0.7584
100,150,100,50	13.6669	0.4491
100,150,150,50	14.3369	0.2523
70,150,100,50	14.1073	0.9269
150,150,100,50	14.1781	0.7595
150,100,70,50	13.9724	0.5823
120,90,70,50	14.2224	0.4025
100,120,80,50	14.0133	0.3131

In addition to a constant learning rate, an exponentially decaying learning rate was tested with gradient descent backpropagation, given by Equation 3, where γ is the decay coefficient and lr_0 is the initial learning rate.

$$lr = lr_0 \gamma^{\frac{step_{current}}{step_{decay}}} \tag{3}$$

The Adam optimiser performed better than either form of gradient descent learning rate as shown in Table III, where all three learning rate types were optimised and compared.
The Adam optimiser indirectly adjusts the learning rate over time by adjusting the weights during backpropagation as a

TABLE III: 5-fold cross validation for optimisation of learning rate. All learning rate types were optimised for comparison

Learning rate (lr) Method	Mean RMSE	Std
Constant value $lr = 0.002$	13.6384	1.042
Exponential decay $lr = 0.01$	12.4675	0.274
Adam $lr = 0.01$, $\beta_1 = 0.4$, $\beta_2 = 0.999$	11.05598	0.905

function of the mean and variance of the gradient (6). The calculation of mean and variance are given by Equation 4, which involves the hyperparameters β_1 and β_2 where m_t and v_t represent the decaying averages of past gradients and squared gradients respectively, and g_t is the current gradient (6).

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta 1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta 2) {g_t}^2 \end{aligned} \tag{4}$$

The exponential ReLu activation function performed best, closely followed by standard ReLu as shown in Fig. A11. Exponential ReLu is similar to Leaky ReLu in that negative outputs are not truncated at zero as in standard ReLu, but rather are allowed to take on small negative exponential values as given by Equation 5 (7). This calculation involves the hyperparameter α, which was also optimised as shown in Fig. A11.

$$f(x) = \begin{cases} x & if x > 0 \\ \alpha(exp(x) - 1) & if x \leq 0 \end{cases} \tag{5}$$

The final round of optimisation was conducted with 10-fold cross validation, shown in Fig. 5, and involved parameters which scored closely in the previous optimisation trials.

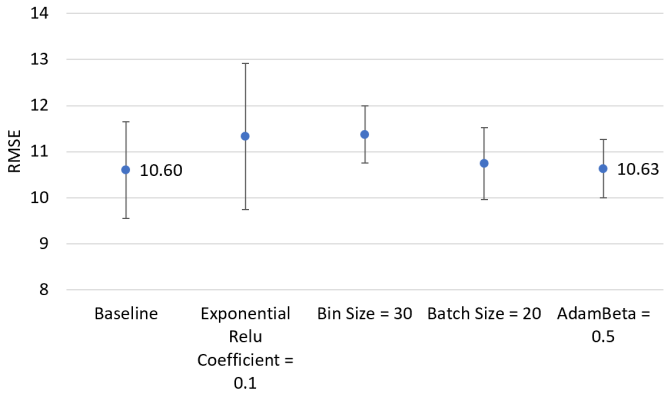

Fig. 5: Each trial varied only one parameter at a time (relative to the baseline) with 10-fold cross validation. This final round of optimisation tested parameters which scored closely in previous trials. The baseline model proved to be best and has the optimised parameters shown in Table I.

This final optimisation resulted in the architecture and parameters shown in Table I.

V. Results

The final model consists of 5 sets of weights, one for each layer of the neural network. Fig. 6 shows the predicted trajectories as trained on 80% of the training data, using

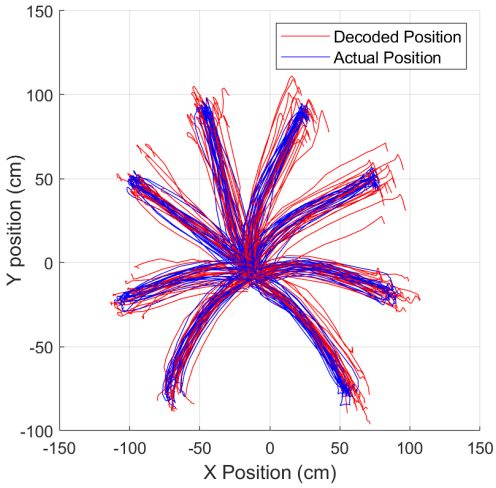

Fig. 6: The optimised parameters shown in Table I were tested on 20% of the training dataset with 80% used for training. RMSE for this run was 10.42cm for the 20 test trajectories. The RMSE on 100% of the test set during the competition was 11.16cm, receiving second place in the competition.

20% for testing. On 20% of the training data, the RMSE was 10.42cm for this trial, and with 10-fold cross validation was $10.6 \pm 1.06cm$. When trained with all training data and tested on all testing data during the competition, RMSE was 11.16cm, receiving second place in the competition.

VI. Contributions

All members contributed equally to code development and report construction.

- Hamstreet: preliminary work, optimisation, writing, editing
- Narman: preliminary work, neural network base code, writing, editing
- Yao: perliminary work, optimisation, writing
- Zhang: preliminary work, optimisation, writing, editing

VII. References

[1] R. P. N. Rao, "Brain-computer interfacing : An introduction," 2013.

[2] A. Georgopoulos, A. Schwartz, and R. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, no. 4771, 1986. [Online]. Available: http://search.proquest.com/docview/213525018/

[3] J. L. Glaser, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording, "Machine learning for neural decoding," 2017. [Online]. Available: https://arxiv.org/abs/1708.00909

[4] R. E. Kass, "Statistical smoothing of neuronal data," vol. 14, no. 1, pp. 5–15, 2003.

[5] Gerstner and Kistler, "Spiking neuron models," 2002, accessed: 2019-21-02. [Online]. Available: http://icwww.epfl.ch/~gerstner/SPNM/node7.html

[6] V. Bushaev, "Adam-latest trends in deep learning optimization," 2018, accessed: 2019-15-03. [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[7] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)." San Juan, Puerto Rico: International Conference on Learning Representations, 2016.

VIII. Appendix

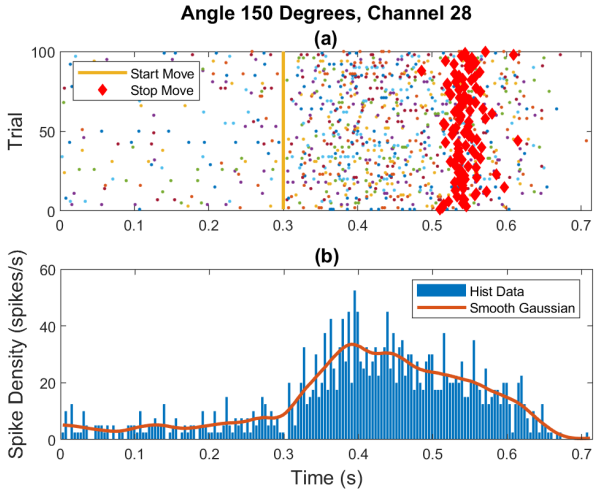

Fig. 7: In contrast with Fig. 1, neuron channel 28 does not show anticipatory behaviour prior to the onset of movement, although the firing rate does appear to decrease in anticipation of the end of movement.

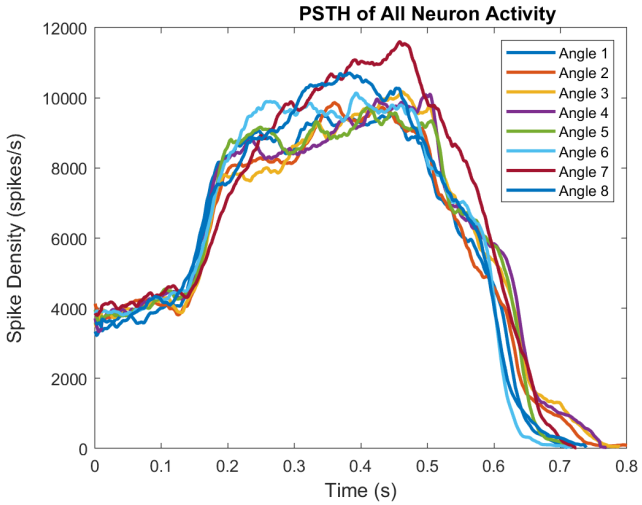

Fig. 8: The PSTH for all neuronal activity during each trajectory angle shows a sharp increase in spiking activity between 141-161 ms. This activity is attributed to anticipatory and planning activity on the part of some neurons (not all, as shown in Fig. A7), where the directional stimulus is present, but activity has not commenced.

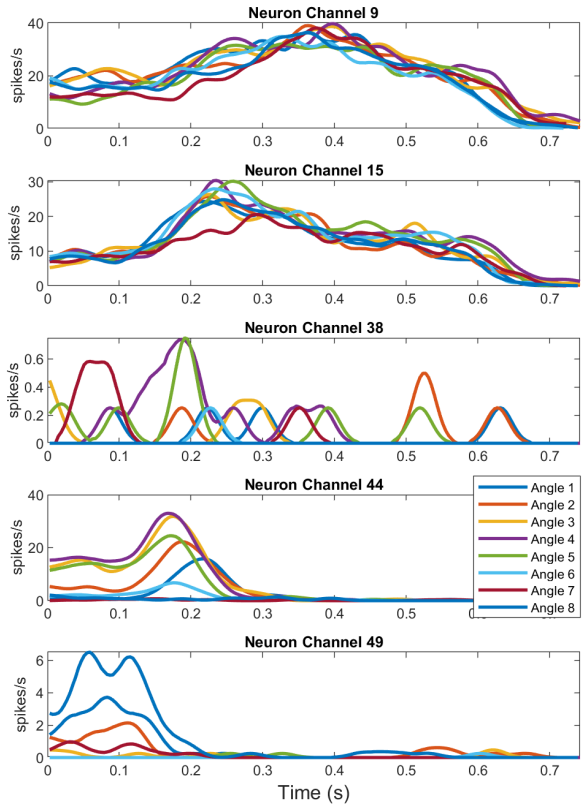

Fig. 9: These 5 neuron channels were rejected in an ANOVA with a p-value of 0.05. These are PSTH plots where only Gaussian smoothed firing rate for each neuron channel over all 100 trials is shown for clarity. During the duration of movement, 300ms to 100ms from the end of each signal, the firing rate of each of these neurons shows little variance between trajectory angles.

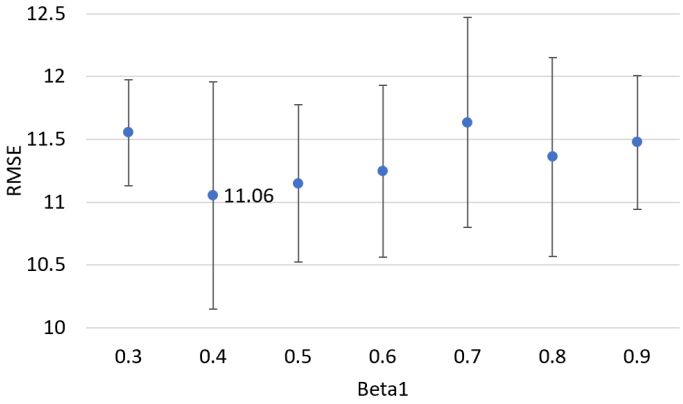

Fig. 10: 5-fold cross validation for optimisation of β_1 parameter for Adam optimiser.

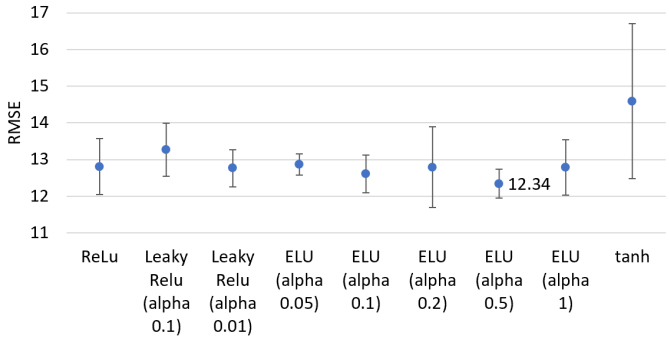

Fig. 11: 5-fold cross validation for optimisation of activation function and coefficients

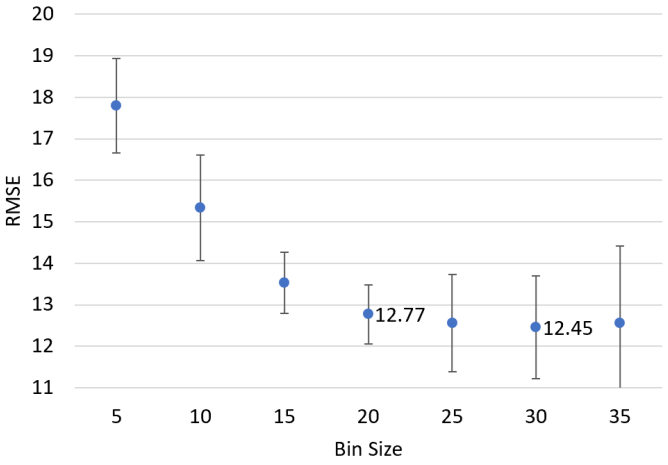

Fig. 12: 5-fold cross validation for optimisation of bin size with gradient descent backpropagation.