Behavioral Classification of Sequential Neural Activity Using Time Varying Recurrent Neural Networks

Yongxu Zhang
Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
`zhangyongxu@ufl.edu`

Shreya Saxena
Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
`shreya.saxena@ufl.edu`

Abstract

Shifts in data distribution across time can strongly affect early classification of time-series data. When decoding behavior from neural activity, early detection of behavior may help in devising corrective neural stimulation before the onset of behavior. Recurrent Neural Networks (RNNs) are common models to model sequence data. However, standard RNNs are not able to handle data with temporal distribution shifts to guarantee robust classification across time. To enable the network to utilize all temporal features of the neural input data, and to enhance the memory of an RNN, we propose a novel approach: RNNs with time-varying weights, here termed Time-Varying RNNs (TV-RNNs). These models are able to not only predict the class of the time-sequence correctly but also lead to accurate classification earlier in the sequence than standard RNNs. In this work, we focus on early robust sequential classification of brain-wide neural activity across time using TV-RNNs as subjects perform a motor task.

1 Introduction

Robust classification of behavior from multi-regional sequential neural data has attracted more and more attention [1, 2]. Temporal neural activity can be classified sequentially in time, which has the potential for early detection of behavior. However, classifying the entire sequence and having a running estimate of the accuracy are often competing goals, especially with significant changes in the distribution of the data across time. Here, we investigate the accurate classification of behavior from neural time-series as early and as reliably as possible. Specifically, we would like to predict the behavior before it happens, while having robust classification across time despite temporal distributional shifts in the data [3][4].

Recurrent Neural Networks (RNNs) are designed for time-series data: they take in sequential inputs and predict the class of the sequence using recurrent hidden states that are able to retain a memory of previous inputs. However, standard RNNs are static in nature, and thus do not perform well on data with long time-series and shifting data statistics. Instead, they are good at predicting from temporal data correctly at the end of the sequence. To help the network utilize all temporal features of the input and to enhance the memory of an RNN, here we propose a novel approach: RNNs with time-varying weights, termed Time-Varying RNNs (TV-RNNs). These models are able to not only predict the class of the sequence correctly, but also lead to accurate classification earlier in the sequence than standard RNNs. In this work, with TV-RNNs, we focus on the early sequential classification of brain-wide neural activity across time, as subjects perform a motor task (Figure 1A). Two different datasets are used: (1) simulated data with chirp signals to simulate distributional shifts in the data, and (2) widefield calcium imaging that records the neural activity across mouse dorsal cortex while

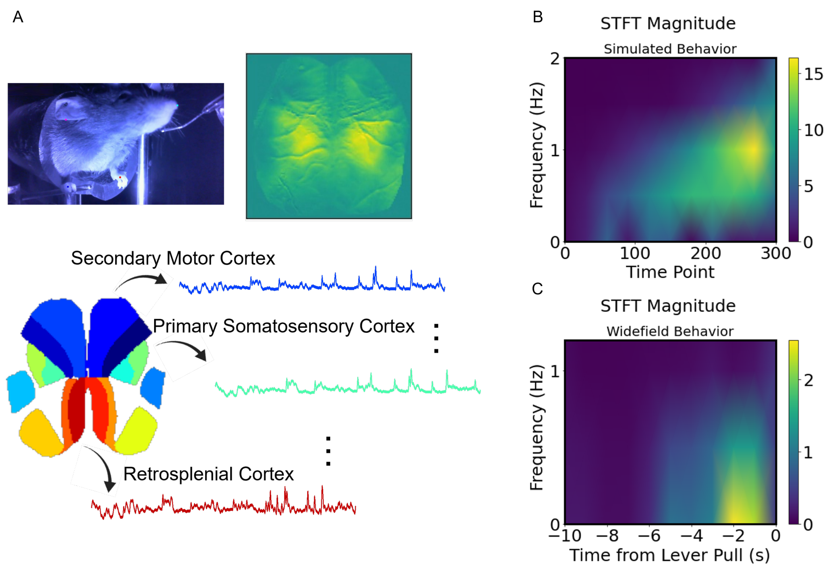

Figure 1: (A) Mice were trained to pull a lever for water reward. Widefield calcium imaging activity was recorded from multiple regions. Short Time Fourier Transform (STFT) magnitude of behavior signal in (B) simulated data and (C) widefield dataset.

subjects perform a behavior, here, a 'lever pull' task, further experimental details are provided in [5]. We first examine the data distribution across time: the data distribution in the behavior trials changes across the trial, as quantified by visualizing the short-term Fourier transform (STFT) (Figure 1B,C). Note that the STFT of the control signals are close to 0 for all frequencies and magnitudes (not shown). Here, we trained TV-RNNs to perform binary classification of behavior, here, lever pull. As a comparison, we show the results while using two different loss functions to train standard RNNs using backpropagation-thru-time (BPTT): (a) the loss at the end of time sequence, which is a common strategy to train RNNs for classification, and (b) the loss at all time steps, which concentrates on not only the classification of the entire sequence but also at each time point in the sequence. As an evaluation metric, we consider the 'temporal accuracy' of an RNN, which quantifies the performance of the model at each timestep, and we aim to enable RNNs to perform classification as early and as accurately as possible. We show that (a) TV-RNNs outperform standard RNNs, (b) we are able to understand the classification mechanisms of TV-RNNs.

2 Related Work

2.1 Behavioral classification using neural activity

Previous research has focused on behavioral decoding by splitting data into multiple windows and independently applying separate classifiers to each window of data. For example, Soon et al. use a support vector machine (SVM) to decode the decision made by humans up to 10 seconds before awareness [1]; in our previous work, we have also succeeded in behavioral decoding using a sliding window approach [5][6]. However, the temporal information hidden in the time series data is not adequately utilized in these models because each classifier is independent. Thus, RNNs become a more effective model because they are able to handle the neural data while taking into account the temporal features.

2.2 Sequential classification

Classification of sequence data has attracted extensive attention and can be applied in many areas, e.g., genomic analysis, information retrieval, and health informatics [7]. Information about the data is stored across the sequence; in our work, we consider temporal sequences, the features for predicting the behavior are not only distributed across time of the brain but also across different regions. By using and storing information across the sequence, RNNs are able to convert their representations across time to adapt to the task, thus, they perform well in classifying sequential data [8]. For example, in [9], the presence of heart disease can be detected by RNNs using electrocardiogram data, while in [10], text sequences are classified by RNNs. Research has been lagging on performing early classification on temporal data, although this is attractive since we would like to predict the class of a time series as soon as possible in order to be able to act on it. Xing et al., explored the minimal prediction length for neural networks to classify the time-series data accurately in [3]. Mori et al., optimized the early index and accuracy of a network at the same time in [4]. Here, we use a time-varying approach to perform early and sequential classification of behavior using neural data.

2.3 Time varying models

Models with time varying parameters are efficient in dealing with temporal tasks, they have unique parameters to utilize specific information at different times. For instance, switching linear dynamical systems (SLDS) and recurrent SLDS are designed to parse data sequences into coherent discrete units which help to capture distinct dynamics in different time periods of time-series data. Moreover, time varying parameter regression models have shown their utility in a range of applications such as economics [11][12]. Classification can also be improved by applying different parameters temporally, e.g., Yang et al., used multiple CNNs in parallel across time to classify time varying signals [13], and Wang et al., show that time varying parameters outperform common machine learning approaches in the classification of EEG signals [14]. However, these methods lack connections between different parts of the models: temporal information that may be crucial for classification is not transmitted across time. On the contrary, the proposed TV-RNNs have explicit hidden states storing and transmitting temporal information across the entire sequence.

3 Methods

3.1 Time-Varying Recurrent Neural Networks

In order to capture the specific temporal features of the input, we design an RNN with time-varying weights including input weights W_x^t, recurrent weights W_h^t, output weights W_y^t, and biases b_h^t, b_y^t.

$$h_t = \tanh(W_h^t h_{t-1} + W_x^t x_t + b_h^t) \ \ \forall t \in [1, T] \tag{1}$$

$$y_t = \sigma(W_y^t h_t + b_y^t) \ \ \forall t \in [1, T] \tag{2}$$

$$W_{x,h,y}^t = W_{x,h,y}^k \ \ \forall t \in [(k-1)w, kw], k \in [1, \frac{T}{w}] \tag{3}$$

$$c = \begin{cases} 0, & \text{if } y_t < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

where w is the window size of RNNs. To avoid overfitting, the inputs in each time window w are fed into RNNs with a fixed set of parameters. Thus, T/w sets of weights are used for the whole sequence of inputs. As baseline models, we train standard RNNs, which have fixed weights for the entire sequence. One type of standard RNNs is trained by using the loss at the end of the sequence (RNN-S1), and the other type of standard RNNs is trained by using the sum of the loss at every time point (RNN-S2). For trained TV-RNNs, We use the weights of standard RNNs trained with the loss at the end of the sequence to be the initialization of all sets of weights in TV-RNNs, in order to keep a relatively high classification accuracy at the end of the sequence (see Appendix A.3 for more details). We chose the number of hidden units as 64, and we trained all networks for 1000 epochs using Adam at a learning rate of 0.0001. These hyperparameters were determined using cross-validation on a sample session of the dataset. We applied 5-fold cross-validation to all of our experiments. We used Pytorch to train our models. We performed all tasks with NVIDIA A100 GPUs. The code is avaliable at `https://github.com/saxenalab-neuro/TV_RNN`.

3.2 Accuracy Quantification

Temporal accuracy In order to describe the performance of temporal decoding and exploring early classification, we use temporal accuracy $Accuracy_t$, which depicts the classification accuracy at each time point t.

Area under accuracy curve (AUAC) We calculated the area under the accuracy curve in different time windows, above chance level. This quantifies the overall decoding ability of the classifier.

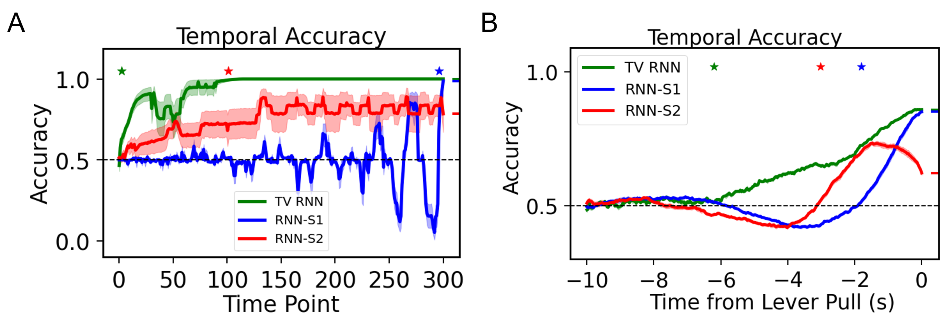

Figure 2: Temporal classification accuracy curve of standard RNNs and TV-RNNs using (A) simulated data and (B) widefield data . The stars on top represent the earliest decoding time for each model (see Section 3.2), and the bars on the right side reflect the final classification accuracy of the sequence. Note that chance accuracy level is 0.5 for both datasets.

Earliest decoding time We would like to classify the sequence as early and as accurately as possible. In order to explore the ability of RNNs in early classification, we introduce a metric, earliest decoding time, to measure early classification. This is the earliest time after which we obtain consistent and significant decoding till behavior onset. Significance was determined using a one-tailed t-test at a significance level of $p < 0.05$ (after multiple hypothesis correction using the Benjamini-Hochberg procedure [15]). This represents the earliest time that the behavior can be reliably decoded.

4 Results

4.1 Classification Accuracy Using TV-RNNs

Classification of simulated data We consider the temporal classification accuracy of a simulated dataset using standard RNNs as compared with TV-RNNs. We first use a common strategy, i.e., BPTT with a binary cross-entropy loss at the end of the sequence, to train a standard RNN to classify the behavior (RNN-S1; see Appendix A.3). Figure 2A (blue curve) shows that the temporal classification accuracy using this strategy only starts to increase above chance level after 150 time points. This trend matches the signal statistics in Figure 1B: after 150 time points, the simulated behavior signal has a higher frequency and magnitude, the data distribution changes drastically. The alternative strategy to train Standard RNNs (RNN-S2), BPTT with the sum of the binary cross-entropy loss over time, leads to a low final accuracy (Figure 2A, red curve). Consequently, a single set of weights in the standard RNNs does not seem to be able to guarantee early and accurate classification. On the other hand, TV-RNNs (Figure 2A, green curve) can not only predict the class of the sequence early in the sequence, but maintain a high classification accuracy throughout the trial.

Classification of behavior from widefield neural activity The experimental dataset was recorded from mice trained to perform a self-initiated lever-pull task with a water reward. At the same time, neural activity was recorded from the dorsal cortex of the mice in the form of widefield calcium imaging activity. We use localized semi-nonnegative matrix factorization (LocaNMF) to process the widefield data and feed the LocaNMF components into the models [16] (see Appendix A.1 and [5] for more details). We train the three types of networks to classify the widefield calcium imaging dataset of 300 time points, i.e., from 10 seconds before the behavior (lever pull) to the time that the behavior happens (see Section 3 and Appendix A.3), in order to quantify the earliest behavioral decoding time and the temporal performance of decoding with real data. Figure A.3C shows how we decide the window size of TV RNNs with widefield dataset We use an example session of one mouse which has a large number of trials, i.e., 378 trials to set the window length between 6 and 100, and compute the AUAC and earliest decoding time. The purple curve shows the AUAC of different w and the brown curve shows the earliest decoding time. Finally, we set the TV-RNN w to 30, which implies that every 30 time points (1 second) will lead to a switch in the weights, because it has the largest AUAC and the earliest decoding time (Figure A.3C). Figure 2B shows the temporal classification accuracy with combined trials (2447 'behavior' trials). We see that the time around lever pull has the highest accuracy in both standard RNNs with S1 training strategy and TV-RNNs. Using standard

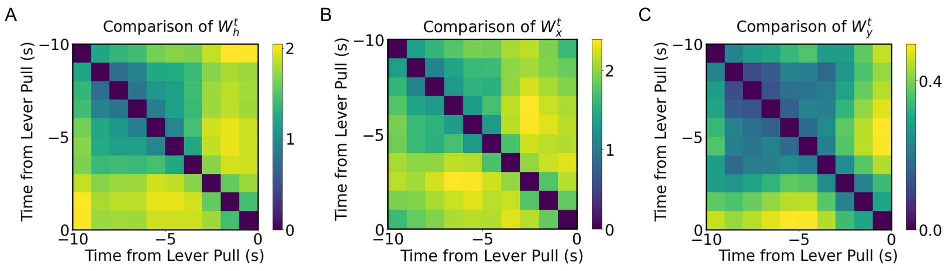

Figure 3: Euclidean distance between (A) W_h^t, (B) W_x^t, and (C) W_y^t of TV-RNN at different times in the trial.

RNNs, the behavior can be classified significantly above chance up to around several seconds prior to the lever pull, i.e., around 2 seconds in S1 and around 3 seconds in S2. This also illustrates that S2 performs better than S1 in early classification but worse in final classification accuracy, i.e., around 0.85 in S1 and 0.62 in S2. We also show that TV-RNNs significantly outperform standard RNNs in most time points in Figure 2B, the earliest decoding time of TV-RNNs can reach around 6 seconds before the lever pull, and the sequential classification accuracy is around 0.86. Next, we evaluate the session-by-session accuracy of the TV-RNNs. We first compute the AUAC in Figure A.2A for selected sessions (with $\#trials \geq 40$). We consistently see that TV-RNNs achieve a higher accuracy than standard RNNs, presumably because TV-RNNs explicitly take into account more temporal structure with time-varying weights in the model, and allow for a monotonically increasing classification accuracy. Finally, we show the earliest decoding time in Figure A.2B, where we see that TV-RNNs outperform standard RNNs in most sessions of the example mouse.

4.2 Model Analysis

In order to understand why TV-RNNs are more efficient and analyze the difference between TV-RNNs and standard RNNs, we compare the learnt time-varying weights across the trial. We measure the Euclidean distance between the same type of time-varying weights at different times. In Figure 3A, B, and C, the color-map illustrates the Euclidean distance, with brighter colors representing a larger difference. We see that the weights at the time closer to the behavior are very different from the weights at the earlier timepoints. Furthermore, the recurrent weights W_h^t in Figure 3A show more changes than the other weights, which may be necessary here to exploit the dynamic nature of the temporal features. Moreover, the sharpest changes in the input and recurrent weights are at around 3 seconds before the behavior, which matches the changes in the signal statistics in the STFT (Figure 1C). The output weights W_y^t do not vary much (note the difference in the color axis), which reveals that the divergence between the trajectories of the two classes already exists in RNN layers, consequently, the output weights can distinguish two classes without many changes. In order to examine the effect of TV RNN weights on classification performance, we calculate the earliest decoding time and AUAC while removing a single set of weights from trained TV RNNs with 10 windows ($T = 300$, $w = 30$) (Appendix Table 1). After removing each of the first 5 sets of weights, the earliest decoding time and the AUAC only change slightly compared with the trained TV RNNs ('All' in Appendix Table 1). Similarly, removal of the last two sets of weights also do not have a large effect on the performance metrics. However, when the parameters for $k = 6$ to $k = 8$ are removed, the earliest decoding time and AUAC varied more. These time periods correspond to the large shifts in the data statistics. Lastly, in order to understand how the exact output of RNNs changes with shifting data distributions, we visualize the trained network activity succinctly: we plot the output of the networks ($y(t)$) in Appendix A.4.

5 Conclusion

In this work, we used RNNs to explore robust early sequential classification with brain-wide neural activity when the data distribution shifts across time. We show that TV-RNNs are able to achieve temporal robust classification earlier than standard RNNs and have higher accuracy. In the future, we aim at robust behavioral decoding by using different types of neural activity.

Acknowledgement

This research was partly funded by the UF AI Catalyst Fund. We thank Catalin Mitelut and Timothy Murphy for providing the widefield dataset.

References

[1] C. S. Soon, M. Brass, H.-J. Heinze, and J.-D. Haynes, "Unconscious determinants of free decisions in the human brain," *Nature neuroscience*, vol. 11, no. 5, pp. 543–545, 2008.

[2] E. Batty, M. Whiteway, S. Saxena, D. Biderman, T. Abe, S. Musall, W. Gillis, J. Markowitz, A. Churchland, J. P. Cunningham *et al.*, "Behavenet: nonlinear embedding and bayesian neural decoding of behavioral videos," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[3] Z. Xing, J. Pei, and S. Y. Philip, "Early classification on time series," *Knowledge and information systems*, vol. 31, no. 1, pp. 105–127, 2012.

[4] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano, "Early classification of time series by simultaneously optimizing the accuracy and earliness," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4569–4578, 2017.

[5] C. Mitelut, Y. Zhang, Y. Sekino, J. D. Boyd, F. Bollanos, N. V. Swindale, G. Silasi, S. Saxena, and T. H. Murphy, "Mesoscale cortex-wide neural dynamics predict goal-directed, but not random actions in mice several seconds prior to movement," *Elife*, vol. 11, p. e76506, 2022.

[6] Y. Zhang, C. Mitelut, G. Silasi, F. Bolanos, N. Swindale, T. Murphy, and S. Saxena, "Uncovering the effect of different brain regions on behavioral classification using recurrent neural networks," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2021, pp. 6602–6607.

[7] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[8] M. Farrell, S. Recanatesi, T. Moore, G. Lajoie, and E. Shea-Brown, "Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion," *Nature Machine Intelligence*, pp. 1–10, 2022.

[9] J. Van Der Westhuizen and J. Lasenby, "Techniques for visualizing lstms applied to electrocardiograms," *arXiv preprint arXiv:1705.08153*, 2017.

[10] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[11] A. D'Agostino, L. Gambetti, and D. Giannone, "Macroeconomic forecasting and structural change," *Journal of applied econometrics*, vol. 28, no. 1, pp. 82–101, 2013.

[12] N. Hauzenberger, F. Huber, G. Koop, and L. Onorante, "Fast and flexible bayesian inference in time-varying parameter regression models," *Journal of Business & Economic Statistics*, vol. 40, no. 4, pp. 1904–1918, 2022.

[13] R. Yang, X. Zha, K. Liu, and S. Xu, "A cnn model embedded with local feature knowledge and its application to time-varying signal classification," *Neural Networks*, vol. 142, pp. 564–572, 2021.

[14] Q. Wang, H.-L. Wei, L. Wang, and S. Xu, "A novel time-varying modeling and signal processing approach for epileptic seizure detection and classification," *Neural Computing and Applications*, vol. 33, no. 11, pp. 5525–5541, 2021.

[15] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal statistical society: series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[16] S. Saxena, I. Kinsella, S. Musall, S. H. Kim, J. Meszaros, D. N. Thibodeaux, C. Kim, J. Cunningham, E. M. Hillman, A. Churchland *et al.*, "Localized semi-nonnegative matrix factorization (locanmf) of widefield calcium imaging data," *PLoS computational biology*, vol. 16, no. 4, p. e1007791, 2020.

[17] D. Xiao, M. P. Vanni, C. C. Mitelut, A. W. Chan, J. M. LeDue, Y. Xie, A. C. Chen, N. V. Swindale, and T. H. Murphy, "Mapping cortical mesoscopic networks of single spiking cortical or sub-cortical neurons," *Elife*, vol. 6, p. e19976, 2017.

[18] Q. Wang, S.-L. Ding, Y. Li, J. Royall, D. Feng, P. Lesnar, N. Graddis, M. Naeemi, B. Facer, A. Ho *et al.*, "The allen mouse brain common coordinate framework: a 3d reference atlas," *Cell*, vol. 181, no. 4, pp. 936–953, 2020.

[19] S. Musall, M. T. Kaufman, A. L. Juavinett, S. Gluf, and A. K. Churchland, "Single-trial neural dynamics are dominated by richly varied movements," *Nature neuroscience*, vol. 22, no. 10, pp. 1677–1686, 2019.

[20] D. Zoltowski, J. Pillow, and S. Linderman, "A general recurrent state space framework for modeling neural dynamics during decision-making," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 680–11 691.

A Appendix

A.1 Experimental details

Widefield experiments record large-scale neural activity from the mouse dorsal cortex through widefield calcium imaging. We analyze widefield neural activity while mice engage in a task. In the experiment, head-fixed water-deprived mice were trained to pull a lever and hold it at an angle (for $>$ 100ms) in order to receive a water supplement [5]. Rewarded lever pulls were identified online (using a lever analog signal). Widefield calcium imaging was recorded from the mouse dorsal cortex [17]. We identify the 'behavior' trials as trials that were tracked in real time to provide water reward, with the trial centered around the initiation of the lever pull behavior. As control trials, we take a random sequence from the task with the same number of time points as 'behavior'. Thus, the 'behavior' trials have a clear behavior initiated at the middle of the trial, unlike the 'control' trials. In order to further eliminate the influence of multiple instances of lever pulls occurring during a 'behavior' trial, we manually selected trials such that only one instance of lever pull is located at the middle of each 'behavior' trial. The neural activity is sampled at 30 time points per second, and each trial in this dataset contains 1800 time points (60 seconds). We spatially align the imaged neural activity with the Allen mouse brain coordinate framework [18] using affine transformations, as previously performed in [19]. We then apply Localized semi-nonnegative matrix factorization (LocaNMF) in [16] on widefield calcium imaging data and take 16 components as identified by LocaNMF, which form our input signals, with each input dimension corresponding to one brain region. In this work, we focus on the signals around the lever pull, i.e., from 10 seconds before lever pull to 0 seconds after lever pull, because 'behavior' trials and 'control' trials are easier to classify during these periods [5][6].

A.2 Simulated data

The simulated behavior data consists of 10 chirp signals with 300 time points in each. Each of 10 signals is multiplied by a distinct coefficient selected within a range of $(1, 4)$, and the amplitude linearly increases across time points in each signal. We then add Gaussian noise, and simulate 2000 trials. The simulated 'control' signals are shuffled 'behavior' signals across time.

A.3 Model training

A.3.1 Standard RNNs

We build a classification model with time-series neural data $x \in \mathbb{R}^{R\times T}$ from R different brain regions and T time points as the input, with the outputs as the different classes of behavior. Here, we implement a hidden recurrent layer with the $tanh$ activation function, and a dense layer at the output with the $sigmoid$ activation function σ to predict the binary class. Following are the equations of the standard RNN networks.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h) \;\; \forall t \in [1, T] \tag{5}$$

$$y_t = \sigma(W_y h_t + b_y) \;\; \forall t \in [1, T] \tag{6}$$

$$c = \begin{cases} 0, & \text{if } y_t < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

where x_t is the neural data from all R regions at time point t, $h_t \in \mathbb{R}^{N\times 1}$ is the value for the N hidden units at time point t, $W_x \in \mathbb{R}^{N\times R}$ is the input weight matrix, $W_h \in \mathbb{R}^{N\times N}$ contains the recurrent weights for the hidden layer, and $W_y \in \mathbb{R}^{1\times N}$ represents the output weight matrix. y_t is the output of dense layer. We use backpropagation-thru-time (BPTT) to train the RNNs. We use two commonly used loss functions to train the standard RNNs: (a) the loss at the last output of RNNs (y_T) in order to focus on the prediction of the entire sequence; and (b) the loss at all time steps of RNNs sequence ($\sum_t y_t$), where we focus on not only the prediction at the end of the sequence, but also on the aggregate performance of the RNNs.

RNN-S1 We train the standard RNNs by using BPTT (see Section 3) with the binary cross entropy loss at the end of sequence. Here, only one set of weights need optimizing. The pseudo-code is shown in Algorithm 1.

Algorithm 1 RNN-S1

```
1: standard RNN weights $W^{S1}$={$W_x$, $W_h$, $W_y$, $b_h$, $b_y$}
2: for $iteration = 1, 2, \ldots$ do
3:     for $batch = 1, 2, \ldots, Max$ do
4:         input the time-series data to standard RNNs (see Equation 5 and Equation 6)
5:         compute binary cross entropy loss at the end of sequence $L = l_T$
6:         BPTT with $L$
7:         Adam optimization $W^{S1}_{old} \leftarrow W^{S1}$
8:     end for
9: end for
```

RNN-S2 We also train the standard RNNs by using BPTT with the sum of binary cross entropy loss at each time point. Note that only one set of weights need optimizing for this strategy as well. The pseudo-code is shown in Algorithm 2.

Algorithm 2 RNN-S2

```
1: standard RNN weights $W^{S2}$={$W_x$, $W_h$, $W_y$, $b_h$, $b_y$}
2: for $iteration = 1, 2, \ldots$ do
3:     for $batch = 1, 2, \ldots, Max$ do
4:         input the time-series data to standard RNNs (see Equation 5 and Equation 6)
5:         compute binary cross entropy loss at the end of sequence $L = \sum_{t=1}^{T} l_t$
6:         BPTT with $L$
7:         Adam optimization $W^{S2}_{old} \leftarrow W^{S2}$
8:     end for
9: end for
```

A.3.2 TV-RNN

In TV-RNNs, multiple sets of weights need to be trained. The optimization of all the TV RNNs weights is performed simultaneously (end-to-end) with the same BPTT, i.e., in each batch training. The pseudo-code of training TV RNNs is shown in Algorithm 3.

A.4 Visualization of RNNs outputs

In order to visualize the trained network activity succinctly, we plot the output of the networks ($y(t)$) in Figure A.1C and D. The RNN output trajectories (Figure A.1C) starts to diverge between the two

Algorithm 3 Time-varying RNN

```
1: window size $w$, time $T$
2: TV RNN weights $W^t$={$W_x^t$, $W_h^t$, $W_y^t$, $b_h^t$, $b_y^t$} $\in W^{1,2,\dots,\frac{T}{w}}$
3: initialize $W^t$=$W^{S1}$ $\forall t$
4: for $iteration = 1, 2, \dots$ do
5:     for $batch = 1, 2, \dots, Max$ do
6:         input the time-series data to TV-RNNs (see Equation 1 and Equation 2)
7:         $W^t = W^k,\ \forall t \in [(k-1)w, kw], k \in [1, \frac{T}{w}]$
8:         compute binary cross entropy loss $L$ by using the sum of loss at each time step $L = \sum_{t=1}^{T} l_t$
9:         BPTT with $L$
10:        Adam optimization $W_{old} \leftarrow W$
11:    end for
12: end for
```

Table 1: Earliest decoding time and AUAC comparison

10 windows TV-RNN	Earliest Decoding Time (s)	AUAC
w/o $k = 1$	-6.0	1.037
w/o $k = 2$	-6.0	0.983
w/o $k = 3$	-6.0	1.063
w/o $k = 4$	-5.9	0.892
w/o $k = 5$	-5.9	0.954
w/o $k = 6$	-4.9	1.005
w/o $k = 7$	-2.4	0.657
w/o $k = 8$	-2.8	0.817
w/o $k = 9$	-6.2	0.988
w/o $k = 10$	-6.2	1.063
All	**-6.2**	**1.089**

classes at an early time, and at around 2 seconds before the behavior, the two trajectories from the two classes start to diverge quickly. Thus, the evidence for decision making between the two classes does not exist in the output nodes until close to the final time step T, at which point the information moves from the memory to the output nodes and the classification is performed. On the contrary, in Figure A.1D, the TV-RNNs output trajectories are diverging at the beginning and towards the decision with accumulation of evidence [20], the 'behavior' and 'control' trajectories start to diverge at around 5 seconds before the behavior, then the 'behavior' trials are moving above the decision level (0.5) and towards 1, at the same time, 'control' trials are moving down and towards 0, and all of them are closer to their corresponding decision across time. We also plot the same output trajectories for the simulated data. The standard RNN has overlapping trajectories between 'behavior' and 'control' at early time, however, they cannot diverge well when the time is closer to the end. The TV-RNNs output trajectories for simulated data have similar tendency as the experimental dataset; they also diverge at the beginning and towards the decision with accumulation of evidence. Therefore, the TV-RNNs are considered as utilizing the temporal features in the data to accumulate evidence to make a decision. This is not the only reason that TV-RNNs outperform standard RNNs (in Figure A.1B) but also why TV-RNNs are able to achieve monotonically increasing classification accuracy.

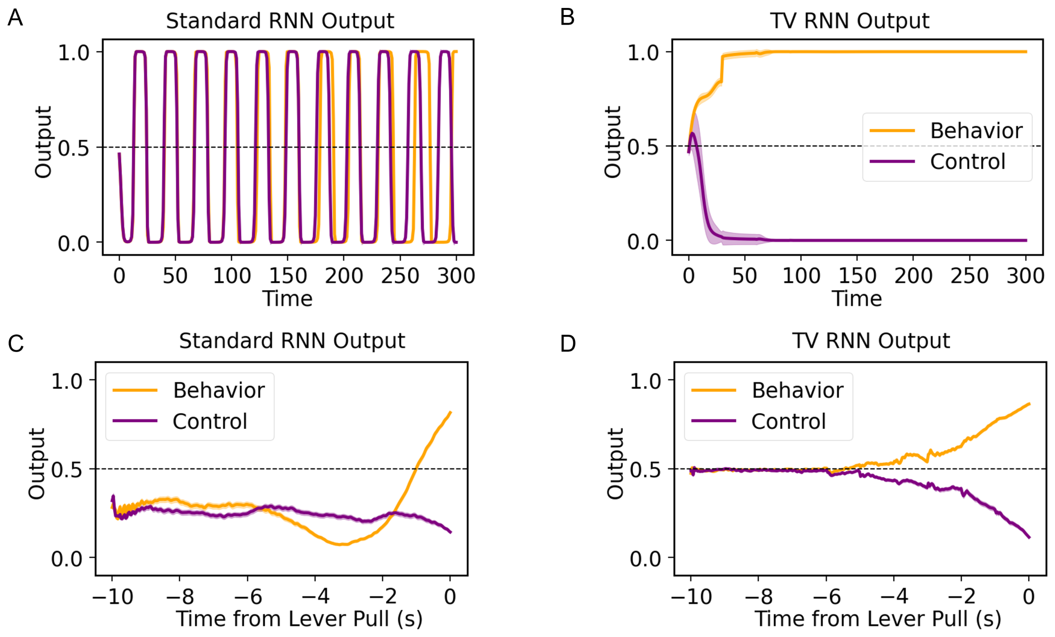

Figure A.1: RNNs outputs of simulated data: (A) Output traje ctories of standard RNNs, average across trials, the shadow means the standard deviation; (B) Output trajectories of TV-RNNs. RNNs outputs of widefield data: (C)standard RNNs; (D) TV-RNNs.

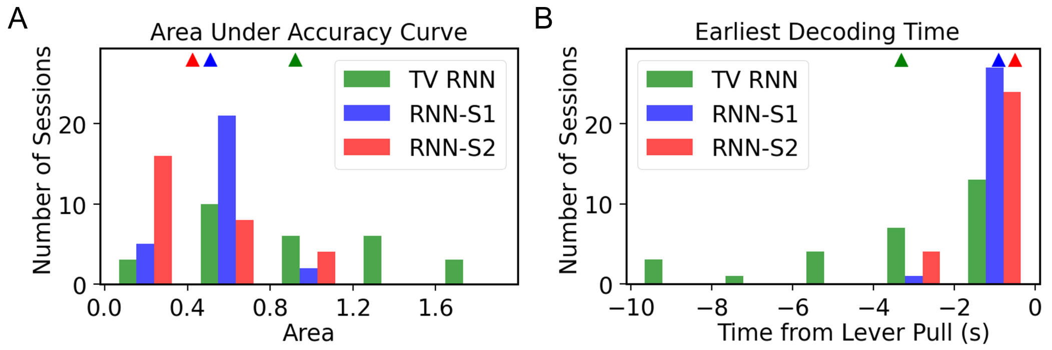

Figure A.2: (A) Histogram of the area under accuracy curve using standard RNNs and TV-RNNs for all sessions of mouse; (B) Histogram of the earliest decoding time using standard RNNs and TV-RNNs for all sessions of mouse.

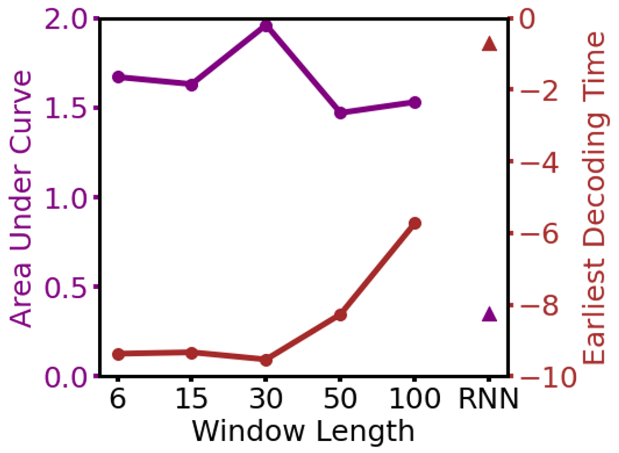

Figure A.3: Determining the window size w of TV-RNN: area under curve and earliest decoding time (see Section 3 of w from 6 to 30; triangles represent standard RNN.