TRAKR - A RESERVOIR-BASED TOOL FOR FAST AND ACCURATE CLASSIFICATION OF NEURAL TIME-SERIES PATTERNS

Muhammad Furqan Afzal; Christian David Márton; Erin L. Rich & Kanaka Rajan
Department of Neuroscience
Icahn School of Medicine at Mount Sinai
New York, NY 10029, USA
muhammadfurqan.afzal@icahn.mssm.edu
{christian.marton, erin.rich, kanaka.rajan}@mssm.edu

ABSTRACT

Neuroscience has seen a dramatic increase in the types of recording modalities and complexity of neural time-series data collected from them. The brain is a highly recurrent system producing rich, complex dynamics that result in different behaviors. Correctly distinguishing such nonlinear neural time series in real-time, especially those with non-obvious links to behavior, could be useful for a wide variety of applications. These include detecting anomalous clinical events such as seizures in epilepsy, and identifying optimal control spaces for brain machine interfaces. It remains challenging to correctly distinguish nonlinear time-series patterns because of the high intrinsic dimensionality of such data, making accurate inference of state changes (for intervention or control) difficult. Simple distance metrics, which can be computed quickly do not yield accurate classifications. On the other end of the spectrum of classification methods, ensembles of classifiers or deep supervised tools offer higher accuracy but are slow, data-intensive, and computationally expensive. We introduce a reservoir-based tool, state tracker (TRAKR), which offers the high accuracy of ensembles or deep supervised methods while preserving the computational benefits of simple distance metrics. After one-shot training, TRAKR can accurately, and in real time, detect deviations in test patterns. By forcing the weighted dynamics of the reservoir to fit a desired pattern directly, we avoid many rounds of expensive optimization. Then, keeping the output weights frozen, we use the error signal generated by the reservoir in response to a particular test pattern as a classification boundary. We show that, using this approach, TRAKR accurately detects changes in synthetic time series. We then compare our tool to several others, showing that it achieves highest classification performance on a benchmark dataset—sequential MNIST—even when corrupted by noise. Additionally, we apply TRAKR to electrocorticography (ECoG) data from the macaque orbitofrontal cortex (OFC), a higher-order brain region involved in encoding the value of expected outcomes. We show that TRAKR can classify different behaviorally relevant epochs in the neural time series more accurately and efficiently than conventional approaches. Therefore, TRAKR can be used as a fast and accurate tool to distinguish patterns in complex nonlinear time-series data, such as neural recordings.

1 INTRODUCTION

The size and complexity of neural data collected has increased greatly (Marblestone et al. (2013)). Neural data display rich dynamics in the firing patterns of neurons across time, resulting from the recurrently connected circuitry in the brain. As our insight into these dynamics increases through new recording modalities, so does the desire to understand how dynamical patterns change across time and, ultimately, give rise to different behaviors. *These authors contributed equally to this work.*
A lot of work in computational neuroscience over the past decade has focused on modeling the collective dynamics of a population of neurons in order to gain insight into how firing patterns are related to task variables (Marton et al. (2020); Richards et al. (2019); Yang et al. (2018); Remington et al. (2018); Kell et al. (2018); Zeng et al. (2018); Pandarinath et al. (2018); Durstewitz (2017); Chaisangmongkon et al. (2017); Rajan et al. (2016); Sussillo et al. (2015); Mante et al. (2013); Sussillo & Barak (2013); Barak et al. (2013); Sussillo & Abbott (2009)). These approaches, however, rely on fitting the whole dynamical system through many rounds of optimization, either indirectly by modeling the task inputs and outputs (Marton et al. (2020); Kell et al. (2018); Chaisangmongkon et al. (2017); Sussillo et al. (2015); Mante et al. (2013); Sussillo & Barak (2013), or directly by fitting the weights of a neural network to recorded firing patterns (Pandarinath et al. (2018); Durstewitz (2017)). Thus, these approaches can be too time- and computation-intensive for certain applications, e.g., in clinical settings where decisions need to be taken based on recordings in real-time. In these settings, neural time-series patterns need to be accurately distinguished in order to, say, detect the onset of seizures, or distinguish different mental states.

Previous approaches to classifying time series lie on a spectrum from simple distance metrics (e.g., Euclidean) to more computationally intensive approaches such as dynamic time warping (Xing et al. (2010)), ensembles of classifiers (Bagnall et al.) or deep supervised learning (Jeong (2020); Fawaz et al. (2019)). Computing simple distance metrics is fast and straightforward, but does not always yield high accuracy results because the patterns may not be perfectly aligned in time. On the other end of the spectrum, ensembles of classifiers and deep learning-based approaches (Bagnall et al., Jeong (2020); Fawaz et al. (2019)) have been developed that can offer high accuracy results, but at high computational cost. Dynamic time warping (DTW) has been consistently found to offer good results in practice relative to computational cost (Fawaz et al. (2019); Bagnall et al. (2016); Serra & Arcos (2014)) and is currently routinely used to measure the similarity of time-series patterns (Dau et al. (2019)).

Previous work in reservoir computing has shown that networks of neurons can be used as reservoirs of useful dynamics, so called echo-state networks (ESNs), without the need to train recurrent weights through successive rounds of expensive optimization (Vlachas et al. (2020); Pathak et al. (2018); Vincent-Lamarre et al. (2016); Buonomano & Maass (2009); Jaeger & Haas (2004); Maass et al. (2002)). This suggests reservoir networks could offer a computationally cheaper alternative to deep supervised approaches in the classification of neural time-series data. However, the training of reservoir networks has been found to be more unstable compared to methods that also adjust the recurrent connections (e.g., via backpropagation through time, BPTT) in the case of reduced-order data (Vlachas et al. (2020)). Even though ESNs have been shown to yield good results when fine-tuned (Tansario & Heidemann (2016); A swolinsky et al. (2016)), convergence represents a significant problem when training ESNs end-to-end to perform classification on complex time-series datasets, and is a hurdle to their wider adoption.

Here, we propose fitting the reservoir output weights to a single time series - thus avoiding many rounds of training that increase training time and could potentially cause instabilities. We use the error generated through the output unit in response to a particular test pattern as input to a classifier. We show that using this approach, we obtain high accuracy results on a benchmark dataset - sequential MNIST - outperforming other approaches such as simple distance metrics (e.g., based on Euclidean distance) and more compute-heavy approaches such as DTW and model-based approaches (e.g., naive Bayes). Importantly, while yielding high accuracy results, even when the data are corrupted by noise, our approach is computationally less time-intensive than DTW.

We also apply our tool, TRAKR, to neural data from the macaque orbitofrontal cortex (OFC), a higher-order brain region involved in encoding expectations, and inducing changes in behavior during unexpected outcomes (Rich & Wallis (2016); Rudebeck & Rich (2018); Jones et al. (2012); Wallis (2012); Schoenbaum (2009); Burke et al. (2009); Wallis & Miller (2003); Schoenbaum et al. (1998)). 128-channel electrocorticography (micro-ECOG) recordings were obtained from the macaque OFC, including anterior and posterior areas 11 and 13, during a reward expectation task. The task was designed to understand how expectations encoded in OFC are updated by unexpected outcomes. We show that TRAKR can be used to distinguish three different behaviorally relevant epochs based on the neural time-series with higher accuracy than conventional approaches.

Taken together, we show that TRAKR is a useful tool for the fast and accurate classification of time-series data. It can be applied to distinguish complex patterns in high-dimensional neural recordings.
2 Methods

2.1 Model Details

TRAKR (Figure 1A) is a reservoir-based recurrent neural network (RNN) with \( N \) recurrantly connected neurons. Recurrent weights, \( J \), are initialized randomly and remain aplastic over time (Buonomano & Maass (2009); Jaeger (b); Maass et al. (2002)). The readout unit, \( z_{\text{out}} \), is connected to the reservoir through a set of output weights, \( w_{\text{out}} \), which are plastic and are adjusted during training. The reservoir also receives an input signal, \( I(t) \), through an aplastic set of weights \( w_{\text{in}} \).

![Figure 1: A) TRAKR setup overview. TRAKR consist of a reservoir connected to input and readout units via dedicated weights. Recurrent weights \( J \) and input weights \( w_{\text{in}} \) are aplastic. Only the output weights \( w_{\text{out}} \) are subject to training. B) TRAKR equations for single unit activity, readout unit activity and error term.](image)

The network is governed by the following equations:

\[
\tau \frac{dx_i(t)}{dt} = -x_i(t) + g \sum_{j=1}^{N} J_{ij} \phi_j(t) + w_{i,\text{in}} I(t) \tag{1}
\]

\[
z_{\text{out}}(t) = \sum_i w_{\text{out},i}(t)x_i(t) \tag{2}
\]

Here, \( x_i(t) \) is the activity of a single neuron in the reservoir, \( \tau \) is the integration time constant, \( g \) is the gain setting the scale for the recurrent weights, and \( J \) is the recurrent weight matrix of the reservoir. The term \( g \sum_{j=1}^{N} J_{ij} \phi_j(t) \) denotes the strength of input to a particular neuron from other neurons in the reservoir and \( I(t) \) is the input signal (Equation 1). \( z_{\text{out}}(t) \) denotes the activity of the readout unit together with the output weights, \( w_{\text{out}} \) (Equation 2). In our notation, \( w_{ij} \) denote the weight from neuron \( j \) to \( i \), and so \( w_{\text{out},i} \) means the weight from \( i^{\text{th}} \) unit in the reservoir to the readout unit. \( \phi \) is the activation function given by:

\[
\phi_i(t) = \tanh(x_i(t)) \tag{3}
\]
We use recursive least squares (RLS) to adjust the output weights, \( w_{out} \) during training (Haykin, Simon S. (1996)). The algorithm and the update rules are given by:

\[
\Delta w_{out,i}(t) = -\eta (z_{out}(t) - f(t)) \sum_j P_{ij}(t) \phi_j(t) \\
w_{out,i}(t) = w_{out,i}(t-1) + \Delta w_{out,i}(t)
\]  

(4)  

Here, \( \eta \) is the learning rate, \( f(t) \) is the target function, and the term \( \sum_j P_{ij}(t) \phi_j(t) \) acts as a regularizer where \( P \) is the inverse cross-correlation matrix of the network firing rates. For details on setting hyperparameters, see Appendix A.

2.2 Adjusting Reservoir Dynamics

During training, the output weights, \( w_{out} \), are optimized using RLS based on the instantaneous difference between the output, \( z_{out}(t) \), and the target function, \( f(t) \). This optimization is performed in one shot (without the need for multiple optimization rounds). Here, we use the reservoir to autoencode the input signal, thus \( f(t) = I(t) \). The instantaneous difference gives rise to an error term, \( E(t) \), calculated as:

\[
E(t) = \sum_{i=1}^{N} \Delta w_{out,i}(t)
\]  

(6)

2.3 Obtaining the Error Signal

After training, the output weights, \( w_{out} \) are frozen. The test pattern is fed to the network via the input, \( I(t) \), and the network is iterated to obtain the error, \( E(t) \) over the duration of the test signal. The error, \( E(t) \) is computed as the difference between the test signal and the network output (Equation 6). The error may vary depending on the similarity of a given test signal to the learned time series. The error is used as input to a classifier.

2.4 Classification of the Error Signal

The error, \( E(t) \), is used as input to a support vector machine (SVM) classifier with a Gaussian radial basis function (rbf) kernel. The classifier is trained using leave-one-out cross-validation. The same classifier and training procedure was used in comparing the different approaches. Accuracy and area under the curve (AUC) are computed as a measure of classification performance.

2.5 Neural Recordings

2.5.1 Task Design

Neural recordings were obtained from the macaque OFC using a custom designed 128-channel micro-ECOG array (NeuroNexus), with coverage including anterior and posterior subregions (areas 11/13). During preliminary training, the monkey learned to associate unique stimuli (natural images) with rewards of different values. Rewards were small volumes of sucrose or quinine solutions, and values were manipulated by varying their respective concentrations.

The behavioral task design is shown in Figure 4A. During the task, the monkey initiated a trial by contacting a touch-sensitive bar and holding gaze on a central location. On each trial, either one or two images were presented, and the monkey selected one by shifting gaze to it and releasing the bar. At this point, a small amount of fluid was delivered, and then a neutral cue appeared (identical across all trials) indicating the start of a 5s response period where the macaque could touch the bar to briefly activate the fluid pump. By generating repeated responses, it could collect as much of the available reward as desired. There were two types of these trials. Match (mismatch) trials were those where the initial image accurately (did not accurately) signal the type of reward delivered on that trial. Behavioral performance and neural time series were recorded in 11 task sessions across 35 days. Each trial was approximately 6.5s long, including different behaviorally relevant epochs and cues.
The macaque performed approximately 550 trials within each task session (mean ± sd: 562 ± 72). Of note, 80% of the trials were match trials within each task session.

2.5.2 Data Pre-processing

ECoG data were acquired by a neural processing system (Ripple) at 30kHz and then resampled at 1kHz. The 128-channel data were first z-score normalized. Second-order butterworth bandstop IIR filters were used to remove 60Hz line noise and harmonics from the signal. We also used second-order Savitzky-Golay filters of window length 99 to smooth the data and remove very high frequency juice pump artifacts (⟩150Hz). For most of the analysis here, we used the average of the 128-channel time series as an input to TRAKR.

3 Results

3.1 Detecting Pattern Changes in Synthetic Time Series

First, we trained TRAKR on idealized, synthetic signals using sin-functions of two different frequencies (Figure 2). Reservoir output weights were fitted to the signal using recursive least squares (RLS; see subsection 2.1). In Figure 2A, the network was trained on the first half of the signal (blue) while the output weights, $w_{\text{out}}$, remained frozen during the second half. Then with $w_{\text{out}}$ frozen, a test signal (orange) was fed to the reservoir. The network output, $z_{\text{out}}(t)$, in red and the error signal, $E(t)$, in green are depicted in Figure 2. The network correctly detects the deviation of the test pattern (orange, 1st half of the signal) from the learned pattern (blue, 1st half of the signal), which results in an increase in the error signal (green, 1st half of the signal, Figure 2A). The second half of the test signal (orange) aligns with the trained signal (blue, 1st half) and thus yields no error (green, 2nd half). In Figure 2B, the order of the training procedure was reversed in that output weights remained frozen for the first half of the signal (blue) and were plastic during the second half. As expected, the increase in the error signal (green) now occurs during the second half of the test signal (orange). Thus, TRAKR correctly detects, via the error signal $E(t)$, when a new frequency pattern occurs in the test signal that deviates from the trained pattern.

![Figure 2](https://example.com/figure2.png)

Figure 2: A) (Blue) $w_{\text{out}}$ plastic for a 15 Hz sin-function, and frozen for a 5 Hz rhythm. (Orange) Test pattern with the same frequencies but the signal order reversed. (Red) TRAKR output. (Green) The error signal, $E(t)$, is showing an increase for the part of the test pattern which was not learned during training. B) Similar to A but $w_{\text{out}}$ were plastic during the second half of the training signal (5Hz rhythm).

3.2 Classifying Digits - Sequential MNIST

We applied TRAKR to the problem of classifying the ten digits from sequential MNIST, a benchmark dataset for time-series problems (Le et al. [2015]; Kerg et al. [2019]).
Figure 3: Classification performance on the sequential MNIST dataset. A) TRAKR outperforms all other methods (99\% AUC; \(*\ast\ast\ast\): \(p < 0.001\), Bonferroni-corrected). NB: naive Bayes; MI: mutual information; Euc: Euclidean distance; DTW: dynamic time warping. B) Classification performance with increasing amount of noise added to the digits. TRAKR performance declines smoothly with noise level, while still outperforming other approaches in classifying noisy digits. Chance level is at 10\%.

For training, we curated a dataset of 1000 sequential MNIST digits including 100 samples for each digit (0-9). We fed each sequential digit (28 x 28 pixel image flattened into a vector of length 784) as a one-shot training signal to TRAKR. Reservoir output weights were again fitted to the signal using recursive least squares (RLS; see Subsection 2.1). After fitting TRAKR to one time series corresponding to one of the samples of a particular digit, we froze the output weights and fed all the other digits as test samples to TRAKR. We obtained an error signal, \(E(t)\), from every test sample, with the magnitude of the error varying depending on the similarity with the learned digit. The error signal was then fed into a classifier which was trained to differentiate the digits based on the error terms (see Subsection 2.4 for more details). We repeated this procedure for all digits and samples in the dataset to obtain the averaged classification performance for TRAKR (Figure 3A).

We obtained a high performance for TRAKR, with an AUC = 99\% (Figure 3A, leftmost entry). We compared the performance of our approach against other common ways of classifying time series (Dau et al. (2019)), again using the same classifier as before (Subsection 2.4). We compared our results against other distance metrics (Euclidean distance, DTW, mutual information) and a generative model (naive Bayes). For DTW, an implementation of the FastDTW algorithm was used (Salvador & Chan (2007)). All other approaches performed significantly worse than TRAKR (\(p < 0.001\)), with naive Bayes derived metric performing the best among the compared approaches (AUC = 86\%).

We also tested the performance of TRAKR under different noise conditions (Figure 3B). For this purpose, we added random independent Gaussian noise to the training digits with \(\mu = 0\) and varying standard deviation (\(\sigma\)). The actual noise that was added (noise levels as depicted in Figure 3B) can be calculated as \(\sigma \ast 255\), with \(\sigma \in [0, 1]\). The number 255 represents the maximal pixel value in the sequential digits. We again compared against all the other approaches, as above. We found TRAKR to perform the best, also at high noise levels: performance decays gradually as the noise is increased and is at AUC = 70\% even at higher noise levels (\(\sigma = 1\)).

We also measured the time it takes to obtain the error signal using TRAKR (Table 1). While it does require upfront fitting, our approach has the advantage that it does not require multiple rounds of optimization because the signal is fit in one shot (see Subsection 2.2 for details). After fitting, TRAKR can detect deviations from the learned signal in real-time. While TRAKR is not as fast as obtaining a simple distance metric, we found that it performs relatively faster than DTW, a commonly used approach to differentiate time-series signals (Table 1). We actually found DTW to yield the lowest accuracy (Figure 3A) of all the approaches. Deep supervised approaches and ensemble methods are computationally even more intensive than DTW (Pawaz et al. (2019)). Altogether, this
Table 1: Computational cost compared

<table>
<thead>
<tr>
<th>Method</th>
<th>Time taken to classify test pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic time warping</td>
<td>188 ms ± 11.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)</td>
</tr>
<tr>
<td>TRAKR</td>
<td>32.8 ms ± 1.52 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>2.14 ms ± 119 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)</td>
</tr>
<tr>
<td>Mutual information</td>
<td>114 µs ± 711 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)</td>
</tr>
<tr>
<td>Euclidean distance</td>
<td>7.19 µs ± 403 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)</td>
</tr>
</tbody>
</table>

shows TRAKR yields good performance at relatively high computational speed, which is beneficial for real-time applications.

3.3 Performance on Neural Time Series Recorded from the Macaque OFC

Figure 4: A) Neural task design (see subsubsection 2.5.1 for detailed description). B) Example neural time series from a single trial recorded from a particular electrode, with three behaviorally relevant epochs (rest, choice and instrumental reward seeking period). Normalized voltage shown as amplitude (arbitrary units). C) TRAKR outperforms all other methods in classifying the 3 neural epochs (** *: p < 0.001, Bonferroni-corrected; chance-level at 33%) NB: naive Bayes; FFT: fast Fourier transform; Euc: Euclidean distance; DTW: Dynamic time warping. AUC in blue, Accuracy in red. D) TRAKR and other methods show a chance performance (50% AUC) in classifying the neural time-series patterns as belonging to match/mismatch trials. E) Classification performance (TRAKR) in distinguishing neural epochs decreases over 11 recording sessions (35 days).
The OFC is involved in encoding and updating affective expectations. We used a behavioral task designed to study the neural mechanisms of how such expectations are encoded in the OFC of primates and how they may guide behavior under different conditions. The goal here was to determine whether TRAKR could classify behaviorally relevant epochs from neural data, and whether it could further distinguish different task conditions (Figure 4A; see also subsubsection 2.5.1 for more details).

A sample of the different neural epochs is shown for a single trial from a particular recording electrode in Figure 4B. The three neural epochs are behaviorally meaningful in that they correspond to rest, choice and instrumental reward seeking. We used TRAKR to classify the neural time series recorded from different trials into these three epochs (see section 2 for more details).

We trained TRAKR on the neural time series corresponding to rest from a particular trial, and used the other complete trials as test signals to obtain the error, $E(t)$ as before. The error signal was used as input to a classifier. We repeated this procedure for all trials in the dataset to obtain the averaged classification performance. We also compared against other conventional approaches, as before. In addition, we also calculated the Fast Fourier transform (FFT) of the signals and obtained the magnitude (power) in the $\alpha$ ($0 - 12$ Hz), $\beta$ ($13 - 35$ Hz), and $\gamma$ ($36 - 80$ Hz) bands within the 3 epochs. We found that TRAKR outperformed all the other methods (Figure 4C), accurately classifying the neural time-series patterns as belonging to either rest, choice, or instrumental reward period ($AUC = 91\%$; $p < 0.001$).

Additionally, we determined whether TRAKR was able to distinguish the neural time-series patterns as belonging to either match or mismatch trials (as described in further detail in section 2). For this purpose, we trained TRAKR on the neural time series corresponding to choice period from a particular trial, and used the other complete trials as test signals to obtain the error, $E(t)$ as before. TRAKR, along with all the other methods, was not able to accurately classify the neural time-series patterns as belonging to either match or mismatch trials (Figure 4D). Further investigations of signal from individual electrodes or in specific frequency bands may be needed to detect such trial-wise differences.

We then used TRAKR to measure the classification performance over recording sessions (Figure 4E), for both classifying behaviorally relevant epochs in the neural signal (Figure 4C) and for classifying trials into either match or mismatch (Figure 4D). We found that classification performance for the behaviorally relevant epochs degrades over days (Figure 4E; blue & red solid lines), while that for match/mismatch trials consistently performs around chance-level (Figure 4E; blue & red dotted lines).

Lastly, we wanted to see if the activations of the units in the reservoir could be used to re-group the electrodes (128-channel recordings) into functionally meaningful groups. For this purpose, we
fitted the reservoir to the time series obtained from a particular electrode, froze the output weights, and used the signal from the other electrodes as test inputs to obtain the error terms. In order to visualize the recordings from the different electrodes in the reservoir space, we performed principal component analysis (PCA) on the tensor of reservoir activities obtained from all the test electrodes. We then projected the signal from every electrode onto the first three principal components of the reservoir space in order to examine if electrodes traced out similar trajectories in this space. Figure 5 shows the result of visualizing four different electrodes in this reservoir space. The four electrodes trace out different paths. Thus, in principle, TRAKR can be used to cluster the neural time series obtained from different electrodes into functionally meaningful groupings, which may represent coherent regions in the brain or interdigitated modules within single regions.

4 DISCUSSION

We have shown that TRAKR can accurately detect deviations from learned signals. TRAKR outperforms other approaches in classifying time-series data on a benchmark dataset, sequential MNIST, and on differentiating behaviorally meaningful epochs in neural data from macaque OFC.

While TRAKR could accurately classify neural epochs, it could not classify neural time-series patterns into either match or mismatch trials. It is possible that receiving a better or worse reward than expected affects the neural signal in distinct/opposite ways, such that the effect is cancelled out on average. It is also possible that the difference in neural time-series patterns is only discernible if the reward is maximally different (better or worse) than expected. In the current task design, there were 4 different levels of reward (flavors) that the macaque associated with different pictures. The number of trials in which the obtained reward was maximally different from the expected was low and possibly not sufficient for accurate classification. Another possibility, corroborated by several studies (Stalnaker et al. (2018); McDannald et al. (2014); Takahashi et al. (2013); Kennerley et al. (2011)), is that OFC neural activity may signal reward values but not reward prediction errors, which instead are mediated through the ventral tegmental area (VTA) in the midbrain.

We found that the classification performance decreased over recording sessions. This could mean that the difference between task epochs being classified decreased because of increased familiarity with the task. That is less likely, however, because the subject was well-trained prior to recordings. Instead, since the signal was recorded over a period of 35 days, the decrease in the classification performance could be a result of degrading signal quality, perhaps due to electrode impedance issues (Kozai et al. (2015a,b); Holson et al. (1998); Robinson & Camp (1991)).

TRAKR offers high classification accuracy at relatively low computational cost, outperforming a commonly used approach such as dynamic time warping (DTW). While ensemble methods and deep supervised approaches may yield high accuracy, they are more time-intensive than DTW (Fawaz et al. (2019)). In particular, deep learning-based approaches, with a high number of parameters to tune, come with high upfront computational cost during training. TRAKR avoids expensive rounds of successive optimization during training by allowing only output weights to change and by fitting a given time series directly using recursive least squares. Moreover, avoiding the need of training on many samples, the error signal can be used directly to distinguish patterns in real-time. This suggests TRAKR can be particularly useful for many real-time applications where available training time is restricted and fast classification performance is desired when deployed.

5 CONCLUSION

There is a need for and renewed interest in tools for the analysis of time-series data (Bhatnagar et al. (2021)). We show that TRAKR is a fast and accurate tool for the classification of time-series patterns. It is suitable for real-time applications where fast classification of time-series patterns is needed, such as in clinical settings. TRAKR is particularly suited for differentiating complex nonlinear signals, such as those obtained from neural or behavioral data in neuroscience, which can shed light on how complex neural dynamics are related to behavior.
6  ACKNOWLEDGEMENTS

This work was funded by NIH 1R01EB028166−01 (Dr. Rajan), NSF FOUNDATIONS Grant 1926800 (Dr. Rajan), Pew Biomedical Scholars Program supported by the Pew Charitable Trusts (Dr. Rich) and NARSAD Young Investigator Grant from the Brain & Behavior Research Foundation (Dr. Rich). We also thank Aster Perkins for neural data collection.
REFERENCES


A TRAKR HYPERPARAMETERS

The recurrent weights $J_{ij}$ are weights from unit $j$ to $i$. The recurrent weights are initially chosen independently and randomly from a Gaussian distribution with mean of 0 and variance given by $g^2/N$. The input weights $w_{in}$ are also chosen independently and randomly from the standard normal distribution.

An integration time constant $\tau = 1ms$ is used. We use gain $g = 1.2$ for all the networks.

The matrix $P$ is not explicitly calculated but updated as follows:

$$P(t) = P(t-1) - \frac{P(t-1)\phi(t)\phi'(t)P(t-1)}{1 + \phi'(t)P(t-1)\phi(t)}$$

The learning rate $\eta$ is given by $\frac{1}{1 + \phi'(t)P(t)\phi(t)}$.

The number of units used in the reservoir is generally $N = 30$. 