Flexible Timing with Delay Networks – The Scalar Property and Neural Scaling

Anonymous ICCM Submission

Abstract

We propose a spiking recurrent neural network model of flexible human timing behavior based on the delay network (DN). The well-known 'scalar property' of timing behavior arises from the model in a natural way, and critically depends on how many dimensions are used to represent input history. The model also produces heterogeneous firing patterns in individual neurons that scale with the timed interval, consistent with available neural data. The model suggests that the scalar property and neural scaling are tightly linked. Further extensions of the model are discussed that may capture additional behavior, such as continuative timing, temporal cognition, and learning how to time.

Keywords: Interval Timing; Scalar Property; Spiking Recurrent Neural Networks; Neural Engineering Framework; Delay Network

Introduction

Time is a fundamental dimension against which our mental lives play out: we remember the past, experience the present, and anticipate the future. Humans are sensitive to a wide range of temporal scales, from microseconds in sound localization to tens of hours in circadian rhythms. It is somewhere in between—on the order of hundreds of milliseconds to several minutes—that we consciously perceive time and coordinate actions in our environment. How does our brain represent time as accurately as possible, and how does it flexibly deal with different temporal intervals?

Scalar Property

Given the centrality of time to our experience, it is no wonder that timing and time perception have been the subject of extensive empirical study over the past 150 years. Many perceptual, cognitive, and neural mechanisms related to time perception have been studied, and perhaps the most well-known finding from the literature is the *scalar property* (Gibbon, 1977). The scalar property captures two fundamental features of timing and time perception:

1. *Mean Accuracy*. The mean of time estimates is accurate, i.e., coincides with the target time.
2. *Scalar Property of Variance*. The standard deviation of time estimates are linearly proportional to the mean of the estimated time.

The scalar property has been confirmed by a wide variety of experimental data (Wearden & Lejeune, 2008). However, some research suggests that the scalar property does not always hold. For example, Grondin (2014) notes that the scalar property of variance critically depends on the range of intervals under consideration, and cites many examples of increases in slope after intervals of about 1.3 seconds.

Most models of timing take the scalar property as a starting point, or consider conformity to the scalar property as a crucial test. This seriously undermines their ability to explain violations of the scalar property. Here, we take the approach of not assuming the scalar property *a priori*, but instead construct a biologically plausible model that is trained to optimally represent time. We then systematically explore ranges of model parameters that lead the scalar property to be satisfied or violated, and provide a theoretical framework for unifying the variety of empirical observations.

Neural Scaling

Variance is not the only property of timing that scales with the estimated time interval. The firing patterns of individual neurons also stretch or compress proportional to the timed interval. In a recent study, Wang, Narain, Hosseini, and Jazayeri (2018) show that neurons in striatum and medial prefrontal cortex (mPFC) scale in this manner. During the timed interval, individual neurons have ramping, decaying, oscillating, or more complex firing patterns. In general, the specific shapes of temporal firing patterns for a given neuron remain the same, but those patterns become stretched for longer intervals and compressed for shorter intervals. Additionally, neurons in the thalamus display a different kind of scaling: their mean level of activity correlates with the timed interval. Both findings have been explained using a recurrent neural network (RNN) model (corresponding to neurons in striatum or mPFC) that receives a tonic input (originating from the thalamus) to scale the temporal dynamics of the network (Wang et al., 2018). The units in the neural network exhibit neural firing patterns and scaling similar to those observed experimentally. These findings suggest that, in order to perform timed actions as accurately as possible, the brain is able to flexibly scale its temporal dynamics. This implies a tight

connection between the scalar property of variance and the temporal scaling of individual neurons. The model of timing we propose reproduces the same findings as the RNN model described in Wang et al. (2018).

Neural Models of Timing

Many neurally inspired models of timing and time perception have been proposed. Some models are based on ramping neural activity (Simen, Balci, deSouza, Cohen, & Holmes, 2011), some decaying neural activity (Shankar & Howard, 2010) and some on oscillating neural activity (Matell & Meck, 2004). Interestingly, all these neural firing patterns (and more complex ones) have been observed by Wang et al. (2018) in striatum and mPFC during a motor timing task. Therefore, appealing to only one of these neural firing patterns may be insufficient to fully explain timing performance. In line with this observation, the recurrent neural network model by Wang et al. (2018) exhibits a wide variety of firing patterns. However, their model does not show why this heterogeneity of firing patterns is important for timing performance or what the role is of ramping, decaying, or oscillating neurons in timing performance. Randomly-connected recurrent neural networks—referred to as *reservoir computers*—already produce a wide variety of dynamics that can subsequently be extracted by a read-out population (Buonomano & Maass, 2009). A more structured approach to building a recurrent neural network may highlight the functional relevance of different neural firing patterns on timing performance.

One candidate for such a structured approach is the *delay network* (Voelker & Eliasmith, 2018). The delay network is a spiking recurrent neural network that approximates a rolling window of its input history by compressing its history into a q-dimensional state. It has been observed that individual neurons in the delay network show responses similar to time-cells (MacDonald, Lepage, Eden, & Eichenbaum, 2011). Here, we use the delay network to explain both the scalar property of timing and the scaling of individual neural responses by comparing delay network data to empirical data from Wang et al. (2018).

Methods

We first discuss the mathematics behind the delay network. Then, we show how to implement the delay network as a spiking recurrent neural network using the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). Lastly, we discuss the details of our simulations that follow the experimental setup of Wang et al. (2018).

The Delay Network

The delay network is a dynamical system that maintains a temporal memory of its input across a rolling window of θ seconds (Voelker & Eliasmith, 2018;Voelker, 2019). It does so by optimally compressing its input history into a q-dimensional state-vector. This vector continuously evolves through time in a way that captures the sliding window of history, while being amenable to representation by a population of spiking neurons using the NEF (as explained in the following subsection).

We consider the problem of computing the function $y(t) = u(t-\theta)$, where $u(t)$ is the input to the network, $y(t)$ is the output of the network, and $\theta > 0$ is the length of the window in time to be stored in memory. In order to compute such a function, the network must necessarily maintain a history of input across all intermediate moments in time, $u(t-\theta')$, for θ' ranging from the start of the window ($\theta' = 0$), going back in time to the end of the window ($\theta' = \theta$). This window must then slide forwards in time once $t > \theta$, thus always preserving the input over an interval of θ in the past. Computing this function in continuous time is challenging, as one cannot merely sample a finite number of time-points and shift them along; the time-step of the system could be arbitrarily small, or there may not even be a time-step as is the case for a mixed-analog implementation on neuromorphic hardware (Neckar et al., 2019).

The approach taken by Voelker and Eliasmith (2018) is to convert this problem into a set of differential equations, $d\mathbf{x}/dt = \theta^{-1}(A\mathbf{x} + Bu)$, where $\mathbf{x}$ is a q-dimensional state-vector, and (A,B) are matrices governing the dynamics of $\mathbf{x}$. We use the (A,B) matrices from Voelker, 2019 (section 6.1.3). This results in the approximate reconstruction: $u(t-\theta') \approx \mathcal{P}(\theta'/\theta) \cdot \mathbf{x}(t)$, where $\mathcal{P}$ are the shifted Legendre polynomials. Importantly, the dimensionality q determines the quality of the approximation. This free parameter controls the number of polynomials used to represent the window – analogous to a Taylor series expansion of the input using polynomials up to order q. Thus, the choice of q determines how much of the input's frequency spectrum, with respect to the ratio θ'/θ, should be maintained in memory. Another important property is that θ^{-1} corresponds to a gain factor on the integration of $\mathbf{x}(t)$ that can be controlled in order to dynamically adjust the length of the window on-the-fly.

The Neural Engineering Framework (NEF)

Given this mathematical formulation of the computations that the neurons must perform in order to represent their past input, we turn to the question of how to recurrently connect neurons such that they perform this computation. For this, we use the NEF (Eliasmith & Anderson, 2003).

In the NEF, the activity of a group of neurons forms a distributed representation of some underlying vector space $\mathbf{x}$. In particular, each neuron i has some *encoder* (or preferred direction vector) $\mathbf{e}_i$ such that this neuron will fire most strongly when $\mathbf{x}$ is similar to $\mathbf{e}_i$. To produce heterogeneity in the neural population, each neuron has a randomly chosen gain α_i and bias β_i. Overall, the current entering each neuron would ideally be $\alpha_i \mathbf{e}_i \cdot \mathbf{x} + \beta_i$. This input current determines the spiking activity of the neuron, based on the neuron model. In this work, we use the standard leaky integrate-and-fire (LIF) model. This results in a pattern of neural activity over time $a_i(t)$ that encodes some continuous vector over time $\mathbf{x}(t)$.

If we have two groups of neurons, one representing $\mathbf{x}$ and

one representing $\mathbf{y}$, and we want $\mathbf{y}$ to be some function of $\mathbf{x}$, then we can form connections from the first population to the second. In particular, we want to connect neuron i to neuron j with weights ω_{ij} such that the total sum from all the input connections will give the same result as the ideal equation assumed above. In other words, we want $\sum_i a_i(t)\omega_{ij} = \alpha_i \mathbf{e}_i \cdot \mathbf{y}(t)$. The ideal ω_{ij} values to achieve this are found using least-squares optimization.

Furthermore, this method for finding connection weights can be extended to support *recurrent* connections (i.e., connections from the neurons in one population back to itself). These connections are solved for in the same manner, and, as has been shown Eliasmith and Anderson (2003), the resulting network approximates a dynamical system of the form $d\mathbf{x}/dt = f(\mathbf{x}) + g(\mathbf{u})$, where $\mathbf{x}$ is the vector represented by the group of neurons, $\mathbf{u}$ is the vector represented by the group of neurons providing input to this group, and the functions f and g depend on both the functions used for solving for the connection weights (as per the previous paragraph) and the temporal properties of the synapses involved (most importantly, the post-synaptic time constant).

The result is that the NEF provides a method for generating a population of neurons (to represent the q-dimensional state) and finding the ideal recurrent connections between those neurons such that they compute the differential equations required by the delay network.

It should be noted that the resulting network is structured exactly like a standard reservoir computer: a large number of neurons are recurrently connected, an input is supplied to that network, and we can decode information from the dynamics of the network by computing weighted sums of the overall neural activity. However, rather than randomly generating the recurrent weights, we are using the NEF to find the optimal weights for storing information over a rolling window in time. This method has been shown to be far more computationally efficient and accurate than various forms of reservoir computer for computing delays (Voelker, 2019).

An example of the resulting system is shown in Figure 1. Here the network is optimized to represent the past $\theta = 1$ s of its own input using $q = 6$ dimensions. Part A shows the (one-dimensional) input to the network over time. In this case, the input is a Gaussian bump centred at $t = 0.5$ seconds. The resulting neural activity (for 50 randomly-chosen neurons) is shown in Part B. Note that the neural activity at the beginning (before the input bump) and at the end (after $t > 1.5$ s) is fairly constant. This is the stable background activity of the network in the absence of any input. Since the network only stores the last second, in the absence of any input it will settle back to this state in ~ 1 second.

Part C shows one example of decoding information out of this network. In particular, we are decoding the function $y(t) = u(t-0.5)$ – that is, the output should be the same as the input $\theta' = 0.5$ seconds ago. This output is found by computing the weighted sum of the spikes that best approximates this value, again, using least-squares optimization to find these

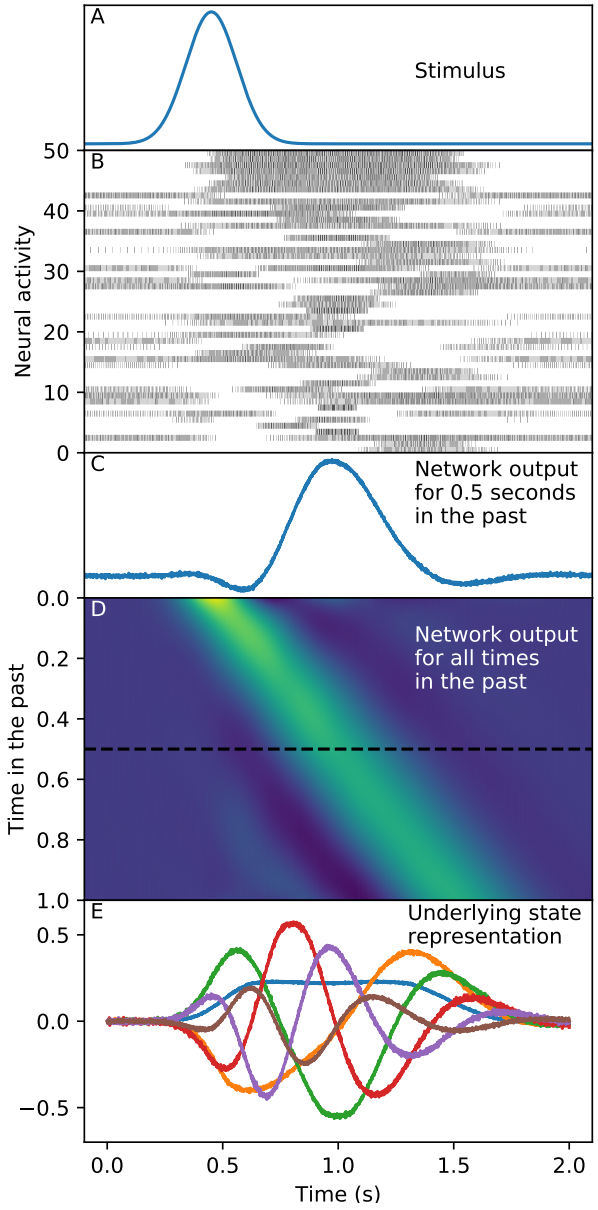

Figure 1: *The Delay Network – Optimized to represent the past* 1 *second of input history using* 6 *dimensions.* **(A):** The input to the network. **(B):** Neural activity of 50 randomly-chosen neurons within the network. **(C):** Decoding information from the network by taking the weighted sum of neural activity that best approximates the input from 0.5 seconds ago. **(D):** Decoding all information from the past 1 second. Each row is a different amount of time (from 0 to 1 second) in the past, and uses a different weighted sum of the same neural activity. The graph in part (C) is a slice through this image, indicated by a dotted line. **(E):** The underlying low-dimensional state information that represents the window.

weights. That is, $y(t) = \sum_i a_i(t) d_i$, where d_i is the decoding weight for the i^{th} neuron. We see that the network accurately represents the overall shape, although the Gaussian bump has become a bit wider, and the output dips to be slightly negative before and after the bump. These are side-effects of the neurons approximating the ideal math for the delay network, and its compression of the input into 6 dimensions.

In Part D, we show the same process as in Part C, but for all times in the past from right now ($\theta' = 0$ s) to the furthest point back in time ($\theta' = 1$ s). This is to show that we can decode all different points in time in the past, and the particular case shown in Part C is just one example (indicated with a dotted line). Each of these different outputs uses the same underlying neural activity, but different decoders d_i out of the recurrent population.

Finally, Part E shows that we can also decode out the q-dimensional state representation $\mathbf{x}(t)$ that the delay network uses for its representation. These are the values that are used to define the delay network, and they form a nonlinear basis for all the possible functions that could be decoded out from the neural activity. Indeed, each row in Part D can also be interpreted as a different linear transformation of the data shown in Part E. Voelker and Eliasmith (2018) derive the closed-form mathematical expression that provides such a transformation, thus relating all time-points within the rolling window to this underlying state-vector.

These different views on the delay network can be seen as a very clear example of David Marr's Tri-Level Hypothesis (Marr, 1982), where we can understand this system at varying levels of abstraction. For instance, we can consider only the *implementational level*, which consists of leaky integrate-and-fire neurons with recurrent connection weights between them, a set of input weights from the stimulus, and multiple sets of output weights. Or we can consider the *algorithmic level*, where the system is representing a q-dimensional state-vector $\mathbf{x}$ and changing that vector over time according to the differential equations given in the previous section. Or we can consider the *computational level*, where the network is storing a (compressed) history of its own input, and different slices of that input can be extracted from that memory. All of these are correct characterizations of the same system.

Simulation Experiment

In the original experiment by Wang et al. (2018), monkeys were presented with a "cue" signal that indicated the interval to be reproduced: red for a short interval (800 ms) and blue for a long interval (1500 ms). Then, they were presented with a "set" signal that marked the start of the interval. The monkeys had to issue a response after the cued interval had passed. We have attempted to match the relevant details of their experimental setup as follows. The delay network (with $q = 4$) continually receives input from a control population that scales θ in order to produce intervals around 800 ms or 1500 ms. In effect, this gain population controls the length of the window on-the-fly. The effective value of θ is 1 divided by the value that the gain population represents. When the value represented by the gain population is greater than 1, it makes the length of the window shorter; when it is smaller than 1, it makes the window longer. This enables us to choose values for the gain population that will let the delay network time intervals around 800ms or 1500ms. The delay network receives input that is continually represented, along with the history of this input. The input signal is a rectangular impulse of 500 ms. Also, a read-out population decodes the history of the input signal at the scaled value of θ.

Results

Scalar Property in the Delay Network

In order to quantify the scalar property in the delay network, we calculated the mean and standard deviation of the decoded output at θ seconds. We performed this analysis for delay networks with a range of values for θ and q. We considered only positive values around the peak of the decoded output. If the scalar property holds, we should observe a linear relationship between theta and the standard deviation of the impulse response. Our data suggests that the scalar property critically depends on q (Figure 2). Generally, the standard deviation is lower for higher values of q, because the representation of the input history becomes more accurate when we use more dimensions. For lower values of q, the standard deviation scales approximately linearly with theta, whereas for $q > 3$, we observe some nonlinearities. For example, for $q = 4$, we observe some break-points around 750 ms and 1750 ms.

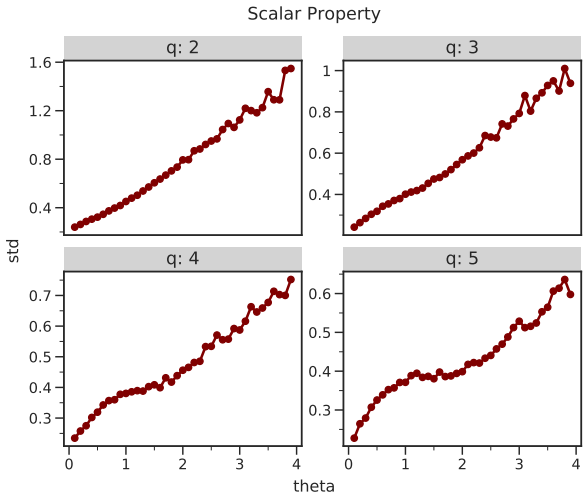

Figure 2: *Scalar Property*. The standard deviation of the impulse response plotted against θ for different values of q.

Neural Scaling in the Delay Network

Our simulations of the Wang et al. (2018) experiment produced results with a qualitative fit to the empirical data (Figure 3). First, the standard deviation of the decoded output

increased with θ (also see previous section). Second, the neural responses were highly heterogeneous, with ramping, decaying and oscillating neurons. These firing profiles were observed because they are linear combinations of the underlying state vector $\mathbf{x}(t)$ (see Figure 1E). Third, the responses of individual neurons stretched or compressed with the length of the timed response response, similarly to the empirical data in Wang et al. (2018).

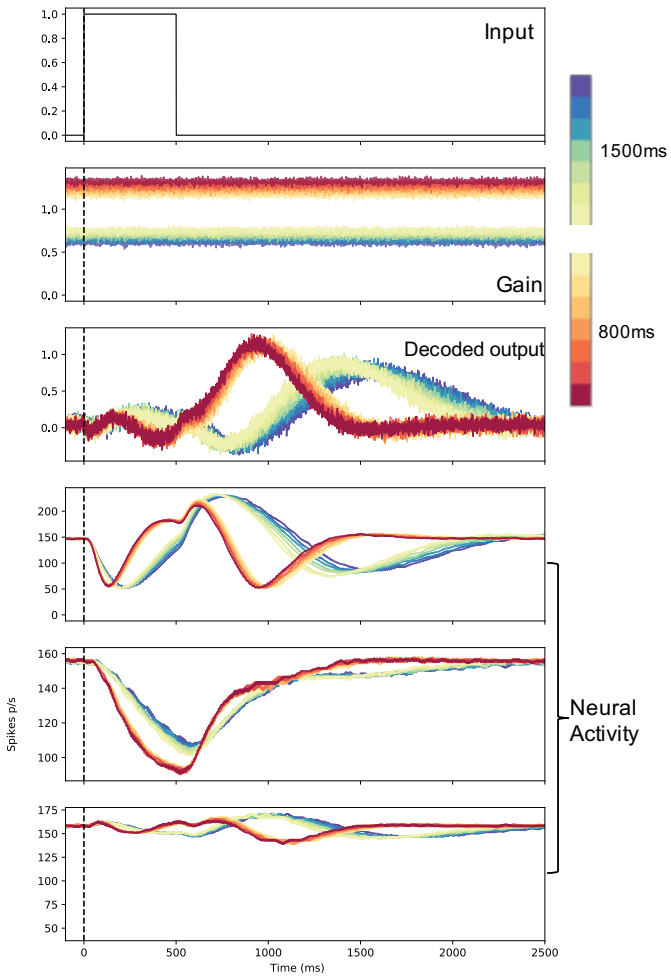

Figure 3: *Neural Scaling*. A square input was provided to the delay network, while varying the value of the gain input. The peak and standard deviation of the decoded output scale with the gain input. The heterogeneous firing patterns of individual neurons also scale with the gain input. Here, neural firing patterns of three example neurons are shown that qualitatively fit the data by Wang et al. (2018). We focused on the first period of the neural response to the "Set" stimulus. The top neuron shows ramping activity, the middle neuron shows decaying activity and the lower neuron shows ocillatory activity.

Discussion

The aim of the present study is to use the delay network to explain two findings in the timing literature: the scalar property of variance and neural scaling. We did not assume the scalar property *a priori*, but systematically explored the parameters of the delay network that lead the scalar property being satisfied or violated. Our results suggest that the scalar property critically depends on q. Notably, the time-cell data that was found in earlier work depended on $q = 6$ (Voelker & Eliasmith, 2018). Conformity to the scalar property may be explained by the delay network having a relatively low number of dimensions, whereas violations may be due to the delay network having more dimensions. We also found that scaling the dynamics of the delay network produces scaling of neural firing patterns, matching empirical data (Wang et al., 2018). Our model suggests that when the delay network represents its input history with more dimensions, neural firing patterns become more complex, since more linear combinations of Legendre polynomials are made. Further, these findings suggest that the scalar property and adaptive control over neural dynamics are tightly linked.

Previous models

The delay network shares some features with previous neural models of timing, but there are also critical differences. First, similar to previous RNN models, the delay network is an RNN that uses population-level dynamics to time intervals. However, previous RNN models use a random connectivity approach to generate the necessary dynamics for accurate timing, whereas the delay network explicitly defines the required dynamics and optimizes neural connectivity to implement those dynamics. Also, past RNN models of timing do not specify how the input history is represented. Similarly to memory models of timing (Shankar & Howard, 2010), the delay network makes this connection explicit. Even though memory models and the delay network both specify how input history is represented, the memory models do not specify how to optimally scale the dynamics of the network or compute arbitrary functions over the represented history. In contrast, the delay network is optimized to recurrently represent time, and comes with a general framework that links the input history, network representation, and spiking neural activity. In sum, we believe that the delay network is an improvement over previous models of timing by both explicitly specifying how time is represented and implementing that representation in a flexible neural framework.

Extending the Delay Network

In this work, we have used the delay network to explain the scalar property and neural scaling in a simple motor timing task. However, the delay network may be used to explain a wide variety of timing phenomena, including: continuative timing, temporal cognition, and learning how to time.

Continuative Timing First, the delay network can be extended to account for time perception in a wide variety of re-

alistic situations. A classic dichotomy in the timing literature is between prospective and retrospective timing. Prospective timing is explicitly estimating an interval with knowledge beforehand that attention should be focused on time. On the other hand, retrospective timing is estimating, in hindsight, how long ago an event happened. However, this distinction may be arbitrary, since in realistic situations, one often notices the duration of an ongoing interval. For instance, you may notice that a web page is taking too long to load but wait an additional amount of time before checking your signal reception. When this happens, one neither has earlier knowledge that time should be attended to (prospective) nor the instruction to estimate how much time has passed since an event (retrospective). Therefore, a more appropriate term for timing in realistic situations would be continuative timing (Van Rijn, 2018). The delay network, at any point in time, provides a rich source of information about the temporal structure of ongoing events, including how long ago an event started and stopped. This information can be used to infer how much time has elapsed since a salient event and compared to the typical temporal structure of an event in memory. Such comparisons could then facilitate decision-making, such as in deciding whether to wait for an additional amount of time.

Temporal Cognition Second, time is a crucial factor in a wide variety of cognitive processes. Timing models have been successfully integrated in ACT-R (Taatgen, Van Rijn, & Anderson, 2007) and models of decision-making (Balci & Simen, 2016). The delay network, built with the NEF, is compatible with other cognitive models that have been developed in the same framework, or indeed any cognitive models that can incorporate neural networks. Therefore, a future avenue of research will be to incorporate the delay network into existing models of cognitive processes, such as action-selection (Stewart, Bekolay, & Eliasmith, 2012) and working memory (Singh & Eliasmith, 2006).

Learning to Time Third, the delay network may be used to explain how timing is learned. In the experiment by Wang et al. (2018), the monkeys trained extensively before they could accurately perform the motor timing task. The monkeys received rewards according to the accuracy of their performance. Another open question is how an optimal mapping between cues and the gain population can be learned. Therefore, future work will focus on modeling how timing is mastered during reinforcement learning.

References

Balci, F., & Simen, P. (2016). A decision model of timing. *Current Opinion in Behavioral Sciences*, *8*, 94–101.

Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, *10*(2), 113–125.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: computation, representation, and dynamics in neurobiological systems*. Cambridge, Mass: MIT Press.

Gibbon, J. (1977). Scalar Expectancy Theory and Weber's Law in Animal Timing. *Psychological Review*, *84*(3), 279–325.

Grondin, S. (2014). About the (Non)scalar Property for Time Perception. In H. Merchant & V. de Lafuente (Eds.), *Neurobiology of Interval Timing* (Vol. 829, pp. 17–32). New York, NY: Springer New York.

MacDonald, C., Lepage, K., Eden, U., & Eichenbaum, H. (2011). Hippocampal Time Cells Bridge the Gap in Memory for Discontiguous Events. *Neuron*, *71*(4), 737–749.

Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman.

Matell, M. S., & Meck, W. H. (2004). Cortico-striatal circuits and interval timing: coincidence detection of oscillatory processes. *Cognitive Brain Research*, *21*(2), 139–170.

Neckar, A., Fok, S., Benjamin, B. V., Stewart, T. C., Oza, N. N., Voelker, A. R., et al. (2019). Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE*, *107*(1), 144–164.

Shankar, K. H., & Howard, M. W. (2010). Timing using temporal context. *Brain Research*, *1365*, 3–17.

Simen, P., Balci, F., deSouza, L., Cohen, J. D., & Holmes, P. (2011). A Model of Interval Timing by Neural Integration. *Journal of Neuroscience*, *31*(25), 9238–9253.

Singh, R., & Eliasmith, C. (2006). Higher-Dimensional Neurons Explain the Tuning and Dynamics of Working Memory Cells. *Journal of Neuroscience*, *26*(14), 3667–3678.

Stewart, T. C., Bekolay, T., & Eliasmith, C. (2012). Learning to Select Actions with Spiking Neurons in the Basal Ganglia. *Frontiers in Neuroscience*, *6*.

Taatgen, N. A., Van Rijn, H., & Anderson, J. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review*, *114*(3), 577–598.

Van Rijn, H. (2018). Towards Ecologically Valid Interval Timing. *Trends in Cognitive Sciences*, *22*(10), 850–852.

Voelker, A. R. (2019). *Dynamical systems in spiking neuromorphic hardware*. Phd thesis, University of Waterloo (on display).

Voelker, A. R., & Eliasmith, C. (2018). Improving Spiking Dynamical Networks: Accurate Delays, Higher-Order Synapses, and Time Cells. *Neural Computation*, *30*(3), 569–609.

Wang, J., Narain, D., Hosseini, E. A., & Jazayeri, M. (2018). Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, *21*(1), 102–110.

Wearden, J. H., & Lejeune, H. (2008). Scalar Properties in Human Timing: Conformity and Violations. *Quarterly Journal of Experimental Psychology*, *61*(4), 569–587.