Constructing compressed number lines of latent variables using a cognitive model of memory and deep neural networks

Sahaj Singh Maini[1], **James Mochizuki-Freeman**[1], **Chirag Shankar Indi**[1],
Brandon G. Jacques[2], **Per B. Sederberg**[2], **Marc W. Howard**[3], **Zoran Tiganj**[1]
[1]{sahmaini, jmochizu, chshan, ztiganj}@iu.edu
Department of Computer Science, Indiana University Bloomington
[2]{bgj5hk, pbs5u}@virginia.edu
Department of Psychology, University of Virginia
[3]marc777@bu.edu
Department of Psychological and Brain Sciences, Boston University

Abstract

Humans use log-compressed number lines to represent different quantities, including elapsed time, traveled distance, numerosity, sound frequency, etc. Inspired by recent cognitive science and computational neuroscience work, we developed a neural network that learns to construct log-compressed number lines from a cognitive model of working memory. The network computes a discrete approximation of a real-domain Laplace transform using an RNN with analytically derived weights giving rise to a log-compressed timeline of the past. The network learns to extract latent variables from the input and uses them for global modulation of the recurrent weights turning a timeline into a number line over relevant dimensions. The number line representation greatly simplifies learning on a set of problems that require learning associations in different spaces – problems that humans can typically solve easily. This approach illustrates how combining deep learning with cognitive models can result in systems that learn to represent latent variables in a brain-like manner and exhibit human-like behavior manifested through Weber-Fechner law.

1 Introduction

The human ability to map sensory inputs onto number lines is critical for rapid learning, reasoning, and generalizing. Recordings of activity from individual neurons in mammalian brains suggest a particular form of representation that could give rise to mental number lines over different variables. For instance, the presentation of a salient stimulus to an animal triggers sequential activation of neurons called *time cells* which are characterized by temporally tuned unimodal basis functions [14, 22, 7]. Each time cell reaches its peak activity at a particular time after the onset of the stimulus. Together, a population of time cells constitutes a temporal number line or a timeline of the stimulus history [12, 23]. Similarly, as animals navigate spatial environments neurons called *place cells* exhibit spatially tuned unimodal basis functions [16]. A population of place cells constitutes a spatial number line that can be used for navigation [3, 2]. The same computational strategy seems to be used to represent other variables as well, including numerosity [17], integrated evidence [15], pitch of tones [1], and conjunctions of these variables [18]. Critically, many of these "neural number lines" appear to be log-compressed [4, 17], providing a natural account of the Weber-Fechner law observed in psychophysics [5, 8]. Here we present a method by which deep neural networks can construct continuous, log-compressed number lines of latent task-relevant dimensions.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

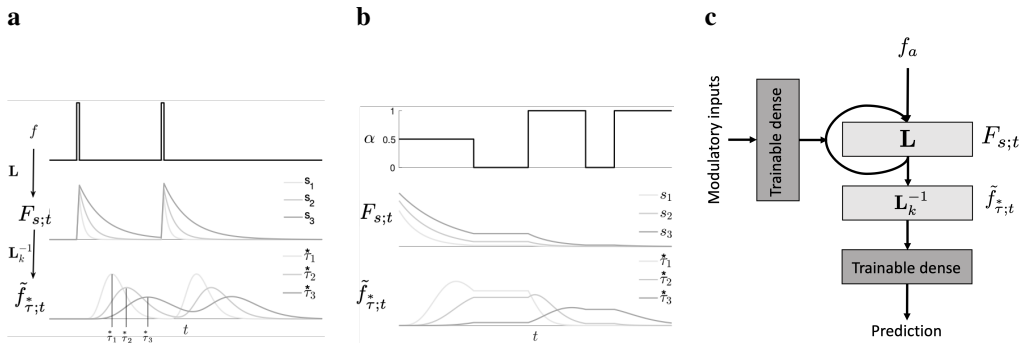

Figure 1: **a.** The impulse response of the Laplace transform approximation decays exponentially with a decay rate s. The impulse response of the inverse Laplace transform approximation has a unimodal form such that each curve peaks at $\overset{*}{\tau}$. Note that if time t was shown on the log-scale, the unimodal curves would be equally wide and equidistant. **b.** α modulates the decay rate of $F_{s;t}$. When $\alpha = 0$, $F_{s;t}$ is constant since its time derivative is 0. When $\alpha = 0.5$ the decay of $F_{s;t}$ is slowed down by a factor of 2 (see also Fig. A1). **c.** Schematic of the CNL network with input f_a and modulatory inputs. Modulatory inputs are passed through a trainable dense layer and modulate the recurrent weights. The network has a single prediction output which is trained to predict the activity of target f_b.

Inspired by experiments on animals, we conduct experiments where a neural network learns number lines for spatial distance and count of objects appearing over time. We designed an experimental setup where the network needs to predict when a target event will happen. In our experiments, time to the target event depends either on traveled spatial distance, or the count of how many times some object appears in the input. Critically, just like in experiments with animals, these variables are not directly observable from the inputs – they are hidden and have to be learned from the spatiotemporal dynamics of the input.

Building on models from computational and cognitive neuroscience [21, 11, 13], we propose a neural network architecture that gives rise to a number line supported by unimodal basis functions. The input is fed into a recurrent layer with the weights analytically computed to approximate the real-domain Laplace transform of the input. Critically, we use properties of the Laplace domain and apply global modulation to the recurrent weights to convert functions of time into functions of other variables, such as distance or count.

2 Methods

We first express a modified version of the Laplace transform $F(s;t)$ of $f(t)$ in a differential form: $\frac{dF(s;t)}{dt} = -sF(s;t) + f(t)$. This modified version differs from the standard Laplace transform only in variable s: instead of s being a complex number, we use real and positive s.

We define a time derivative of $x(t)$ as $\alpha(t) = \frac{dx}{dt}$ and modify Eq. (2) such that $\alpha(t)$ modulates the decay rate s via the chain rule giving rise to the Laplace transform of $f(x(t))$, $F(s;x)$:

$$\frac{dF(s;t)}{dt} = -\alpha(t)sF(s;t) + f(t) \longrightarrow \frac{dF(s;x)}{dx} = -sF(s;x) + f(x). \tag{1}$$

The inverse Laplace transform which we denote as $\tilde{f}(\overset{*}{\tau};t)$ can be computed using the Post inversion formula [19]: $\tilde{f}(\overset{*}{\tau};t) = \mathbf{L}_k^{-1}F(s;t) = \frac{(-1)^k}{k!}s^{k+1}\frac{d^k}{ds^k}F(s;t)$ where $\overset{*}{\tau} := k/s$ and $k \to \infty$. See Fig. 1 and Fig. A1 for illustration of how $\tilde{f}(\overset{*}{\tau};t)$ constitutes a Compressed Number Line (CNL). In the rest of the paper, we refer to our approach as CNL.

We constructed an approximation of the modified Laplace and inverse Laplace transform as a two-layer neural network with analytically computed weights and log-spaced $\overset{*}{\tau}$ (Fig. 1c). The first layer implements the modified Laplace transform through an RNN. The second layer implements the inverse Laplace transform as a dense layer with weights analytically computed to implement k-th order derivative with respect to s (see Appendix Sec. A.1 for details on discretization).

To convert a log-compressed timeline into a log-compressed number line, we implemented global modulation of recurrent weights as in Eq. (1). We assume that α is not known and that it needs to be learned from the input data. $\alpha(t)$ is computed through a feedforward projection which receives a subset of inputs designated as modulatory inputs (*e.g.*, inputs from which a latent variable, such as velocity or count, can be computed). We feed $\tilde{f}_{\overset{*}{\tau};t}$ into a trainable dense layer with a single output and a sigmoid activation function.

3 Results

We tested the proposed network on two event prediction experiments. Event a predicted event b such that the time between them depends on either traveled distance with velocity as a latent variable which can be learned as a combination of the inputs (Experiment 1) or the number of occurrences of a specific pattern (Experiment 2). Our goal was to demonstrate that the proposed network can identify latent variables by learning an appropriate mapping from inputs onto a global modulatory signal α.

CNL parameters were kept the same for all the experiments. We set $k = 8$ and we used 64 neurons in F and 50 neurons in $\tilde{f}$ with input f_a. The number of neurons in $\tilde{f}$ was smaller than in F in order to avoid edge effects when taking a derivative with respect to s (the number of units in $\tilde{f}$ is $2k$ less than the number of units in F; see Appendix Sec. A.2 for details on the computation of the discrete approximation of the derivative). $\overset{*}{\tau}$ was composed of 50 log-spaced values between 5 and 20000. These lower and upper limits were chosen to cover the entire range of temporal relationships in the input signal. The output of $\tilde{f}$ layer was fed through trainable weights into a single output neuron with a sigmoid activation function.

To evaluate how commonly used RNNs perform on this task we compared CNL with a simple RNN, Gated Recurrent Units (GRU) [6], Long Short-Term Memory (LSTM) [9, 10], Lagrange Memory Unit (LMU) [24], and Coupled Oscillatory RNN (coRNN) [20] each followed by a fully connected layer. For LMU we experimented with hyperparameters and got the best performance with: $dt = 1$, $\theta = 5000$ and $d = 128$. For coRNN we set the hyperparameters to the same values as in Rusch and Mishra [20]: $dt = 0.016$, $\gamma = 94.5$ and $\epsilon = 9.5$. The hidden size was 64 in all cases. To investigate the importance of the inverse Laplace transform, we also compared CNL (which is composed of the Laplace and inverse Laplace transform) with $F(s)$ (only the Laplace transform, without the inverse) combined with a trainable dense layer of 50 neurons. We label the later model as CNL-F.

3.1 Experiment 1: Predicting distance in the presence of modulatory inputs

In this experiment input f_a predicts input f_b, but the time between them was modulated by other inputs, f_c and f_d (see Fig. A2 for an illustration of the experiment). For the CNL network, this meant that it had to learn that $\alpha = w_1 f_c + w_2 f_d$, where w_1 and w_2 were randomly generated weights between 0 and 1 and f_c and f_d had a magnitude ranging from 0 to 5. The network only observed f_c and f_d and not the weights w_1 and w_2, so it had to learn the relationship between α and f_c and f_d. We can view this problem as one of learning spatial distance between event a and event b such that velocity (in this case $\alpha(t)$) is not directly observable but has to be learned from the modulatory inputs f_c and f_d. Therefore, a does not predict b after a particular amount of time has elapsed, but instead after traveling a particular distance at a velocity that has to be learned from the network inputs. In the absence of modulatory inputs $f_a(t) = \delta(x)$ and $f_b(t) = \delta(x + x')$ with $x' \in [50, 500, 5000]$. Modulatory inputs f_c and f_d were stepwise functions broken into 15 pieces, with the amplitude of each piece between 0 and 10. The two modulatory inputs were passed through a set of weights W_α (one weight for each modulatory input), adding two more parameters to CNL.

We constructed 6 training, 22 validation, and 22 test examples. The error was quantified as binary cross-entropy loss in predicting $f_b(t)$. Results from Experiment 2 are given in Table 1 (average distance) and Table A1 (cross-entropy loss). Despite having much fewer parameters, CNL performed better than the other approaches at all timescales. We attribute this to the ability of CNL to learn α and construct a number line and then simply learn which component on the number line corresponds to the target. This is a simple associative task. From observing plots in Fig. A4 we see that other approaches failed to provide a useful prediction.

	x'=50	x'=500	x'=5000	Params
CNL-F	29.7 ± 14.5	354.0 ± 0.0	8546.1 ± 9514.7	**53**
CNL	$\mathbf{2.4 \pm 0.7}$	$\mathbf{35.2 \pm 5.2}$	$\mathbf{433.4 \pm 118.1}$	**53**
RNN	57.7 ± 22.1	735.2 ± 139.3	6783.5 ± 3448.6	4481
LSTM	55.9 ± 8.7	607.1 ± 50.6	8200.2 ± 3628.6	17729
GRU	62.2 ± 13.7	592.8 ± 90.9	5293.3 ± 1597.3	13313
coRNN	49.9 ± 10.7	568.1 ± 57.4	5553.4 ± 913.9	8513
LMU	47.9 ± 31.4	778.3 ± 128.8	7884.5 ± 2141.8	12740

Table 1: Experiment 1: The average distance between the actual timestamp of the target event and the timestamp that received the highest probability estimate.

	count=10		count=200		Params
	Loss	Distance	Loss	Distance	
CNL-F	0.469 ± 0.249	24.2 ± 53.1	0.085 ± 0.001	1005.0 ± 0.0	**196**
CNL	$\mathbf{0.201 \pm 0.036}$	$\mathbf{1.3 \pm 0.4}$	$\mathbf{0.077 \pm 0.004}$	$\mathbf{75.8 \pm 21.5}$	**196**
RNN	0.323 ± 0.076	69.8 ± 24.5	0.108 ± 0.023	1365.1 ± 421.1	5377
LSTM	0.303 ± 0.140	62.0 ± 14.2	0.096 ± 0.010	1264.4 ± 941.1	21313
GRU	0.324 ± 0.119	28.8 ± 1.7	0.116 ± 0.072	629.0 ± 1006.0	16001
coRNN	0.636 ± 0.117	60.8 ± 25.3	0.093 ± 0.000	1619.3 ± 0.0	9409
LMU	0.402 ± 0.297	65.5 ± 87.8	0.146 ± 0.135	597.7 ± 781.4	13650

Table 2: Experiment 2: Binary cross-entropy loss across different scales and the average distance between the actual timestamp of the target event and the timestamp that received the highest probability estimate.

3.2 Experiment 2: Counting

In Experiment 2, the time between a and b was modulated by the number of specific patterns presented as modulatory inputs (Fig. A3). At each time step, a pattern composed of 16 elements with binary values was presented. One of the patterns, which we refer to as the target pattern, was repeated multiple times and the network had to learn that the time between a and b depends on the number of target patterns. The patterns were generated randomly.

To solve this task, the network had to learn to recognize the target pattern and count its repetitions. In this case, α had to be learned as a temporal derivative of the count. Whenever the target pattern appears, the count changes by one, therefore $\alpha = 1$. Whenever the pattern was not the target pattern, the count stays the same and $\alpha = 0$. Fig. A1d shows an illustration of this, assuming that the network has learned appropriate α.

We conducted the experiment at two different scales, one with 10 target counts and one with 200 target counts. In the first case, the input signal had 200 time steps, and in the second case, it had 2000 time steps. The results are shown in Table 2. At both scales, CNL performed better than the other approaches, demonstrating that it learned the temporal derivative and constructed a number line representing the count of the target pattern. Fig. A5 shows prediction results for each of the seven networks.

4 Discussion

Cognitive models provide immense utility in understanding neural computations in the brain, but they are usually limited to handcrafted features and associative learning. Here we have shown that incorporating cognitive models into neural networks can expand the utility and explanatory power of these models. CNL can take advantage of deep learning and learn latent features while at the same time utilizing the benefits of the cognitive model and structured representation of knowledge which allows easy associative learning. Activity patterns of neurons in the proposed network resemble the activity of neurons recorded in mammalian brains. Fig. A1 shows time cells, place cells (in 1D environment) and cells tuned to a particular number of objects. Each of these cell types has been recorded in mammalian hippocampus [3, 14, 15] illustrating how cognitive models embedded in deep networks can generate useful neural predictions.

Acknowledgment

We gratefully acknowledge support from the Defense Advanced Research Projects Agency (DARPA) under project Time-Aware Machine Intelligence (TAMI) and the National Institutes of Health's National Institute on Aging, grant 5R01AG076198-02. This content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA or the National Institutes of Health's National Institute on Aging. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

References

[1] D. Aronov, R. Nevers, and D. W. Tank. Mapping of a non-spatial dimension by the hippocampal-entorhinal circuit. *Nature*, 543(7647):719–722, 2017. doi: 10.1038/nature21692.

[2] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557 (7705):429–433, 2018.

[3] J. Bures, A. Fenton, Y. Kaminsky, and L. Zinyuk. Place cells and place navigation. *Proceedings of the National Academy of Sciences*, 94(1):343–350, 1997.

[4] R. Cao, J. H. Bladon, S. J. Charczynski, M. E. Hasselmo, and M. W. Howard. Internally generated time in the rodent hippocampus is logarithmically compressed. *bioRxiv*, 2021.

[5] N. Chater and G. D. A. Brown. From universal laws of cognition to specific cognitive models. *Cognitive Science*, 32(1):36–67, 2008. doi: 10.1080/03640210701801941.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] H. Eichenbaum. Time cells in the hippocampus: a new dimension for mapping memories. *Nature Reviews Neuroscience*, 15(11):732–44, 2014. doi: 10.1038/nrn3827.

[8] G. Fechner. *Elements of Psychophysics. Vol. I.* Houghton Mifflin, 1860/1912.

[9] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] M. W. Howard, C. J. MacDonald, Z. Tiganj, K. H. Shankar, Q. Du, M. E. Hasselmo, and H. Eichenbaum. A unified mathematical framework for coding time, space, and sequences in the hippocampal region. *Journal of Neuroscience*, 34(13):4692–707, 2014. doi: 10.1523/JNEUROSCI.5808-12.2014.

[12] M. W. Howard, K. H. Shankar, W. Aue, and A. H. Criss. A distributed representation of internal time. *Psychological Review*, 122(1):24–53, 2015.

[13] M. W. Howard, A. Luzardo, and Z. Tiganj. Evidence accumulation in a laplace domain decision space. *Computational brain & behavior*, 1(3):237–251, 2018.

[14] C. J. MacDonald, K. Q. Lepage, U. T. Eden, and H. Eichenbaum. Hippocampal "time cells" bridge the gap in memory for discontiguous events. *Neuron*, 71(4):737–749, 2011.

[15] A. S. Morcos and C. D. Harvey. History-dependent variability in population dynamics during evidence accumulation in cortex. *Nature Neuroscience*, 19(12):1672–1681, 2016.

[16] E. I. Moser, E. Kropff, and M.-B. Moser. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.*, 31:69–89, 2008.

[17] A. Nieder and E. K. Miller. Coding of cognitive magnitude: compressed scaling of numerical information in the primate prefrontal cortex. *Neuron*, 37(1):149–57, 2003.

[18] E. H. Nieh, M. Schottdorf, N. W. Freeman, R. J. Low, S. Lewallen, S. A. Koay, L. Pinto, J. L. Gauthier, C. D. Brody, and D. W. Tank. Geometry of abstract learned knowledge in the hippocampus. *Nature*, pages 1–5, 2021.

[19] E. Post. Generalized differentiation. *Transactions of the American Mathematical Society*, 32:723–781, 1930.

[20] T. K. Rusch and S. Mishra. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. *arXiv preprint arXiv:2010.00951*, 2020.

[21] K. H. Shankar and M. W. Howard. A scale-invariant internal representation of time. *Neural Computation*, 24(1):134–193, 2012.

[22] Z. Tiganj, J. Kim, M. W. Jung, and M. W. Howard. Sequential firing codes for time in rodent mPFC. *Cerebral Cortex*, 27:5663–5671, 2017.

[23] Z. Tiganj, J. A. Cromer, J. E. Roy, E. K. Miller, and M. W. Howard. Compressed timeline of recent experience in monkey lPFC. *Journal of Cognitive Neuroscience*, 30:935–950, 2018.

[24] A. Voelker, I. Kajić, and C. Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

A Appendix

A.1 Neural networks implementation of the Laplace and inverse Laplace transform with global weight modulation

While in the Laplace domain s is a continuous variable, here we redefine s as N elements long vector. We can now write a discrete-time approximation of Eq. (1) as an RNN with a diagonal connectivity matrix and a linear activation function:

$$F_{s;t} = WF_{s;t-1} + f_t, \tag{2}$$

where $W = \text{diag}(e^{-\alpha(t)s\Delta t})$ is an N by N diagonal matrix. At every time step t, $F_{s;t}$ is N elements long vector. For brevity of the notations, we assume that the duration of a discrete-time step $\Delta t = 1$.

Following $\tilde{f}(\overset{*}{\tau};t) = \mathbf{L}_k^{-1}F(s;t) = \frac{(-1)^k}{k!}s^{k+1}\frac{d^k}{ds^k}F(s;t)$, a discrete approximation of the inverse Laplace transform, $\tilde{f}_{\overset{*}{\tau};t}$, can be implemented as a dense layer on top of $F_{s;t}$. The connectivity matrix of the dense layer is $\mathbf{L}_k^{-1}$(see Appendix Sec. A.2 for the derivation of the exact matrix form of $\mathbf{L}_k^{-1}$).

To interpret $\tilde{f}_{\overset{*}{\tau};t}$ and to select values s in an informed way, we compute the impulse response of $\tilde{f}_{\overset{*}{\tau};t}$. For input $f(t) = \delta(0)$, the activity of $\tilde{f}_{\overset{*}{\tau};t}$ is:

$$\tilde{f}_{\overset{*}{\tau};t} = \frac{1}{u(t)}\frac{k^{k+1}}{k!}\left(\frac{u(t)}{\overset{*}{\tau}}\right)^{k+1}e^{-k\frac{u(t)}{\overset{*}{\tau}}}, \tag{3}$$

where $u(t = t_n) = \sum_{i=0}^{t_n}\alpha(t_i)$. If $\alpha(0 \le t < t_n) = 1$, then $u(t = t_n) = t_n$. To investigate properties of this approximation and to motivate our choice for $\overset{*}{\tau}$ and consequently s, we analyze the case where $\alpha(0 \le t \le t_n) = 1$.

The impulse responses of units in $\tilde{f}_{\overset{*}{\tau};t}$ is a set of unimodal basis functions (Fig. 1a and Fig. A1a). To better characterize its properties we first find the peak time by taking a partial derivative with respect to t, equate it with 0 and solve for t: $\partial\tilde{f}_{\overset{*}{\tau};t}/\partial t = 0 \rightarrow t = \overset{*}{\tau}$. Therefore each unit in $\tilde{f}_{\overset{*}{\tau};t}$ peaks at $\overset{*}{\tau}$. Note that if we computed the exact continuous-time Laplace and inverse Laplace transform (which would require infinitely many neurons, since s and $\overset{*}{\tau}$ would be continuous variables), the impulse response would be a $\delta(\overset{*}{\tau})$. This would provide a perfect reconstruction of the input function $f(0 < t' < t)$ rather than its approximation.

To further characterize our approximation, we express the width of the unimodal basis functions of the impulse response of $\tilde{f}_{\overset{*}{\tau};t}$ through the coefficient of variation c (see Appendix Sec. A.3 for the derivation of c): $c = 1/\sqrt{k+1}$. Importantly, c does not depend on t and $\overset{*}{\tau}$, implying that the width of the unimodal basis functions increases linearly with their peak time. Therefore when observed as a function of $\log(t)$, the width of the unimodal basis functions is constant. Note that this property is critical for log-compression.

We choose values of $\overset{*}{\tau}$ as log-spaced between some minimum and maximum (see Results section for the values used in the experiments). Because of the log-spacing and because c does not depend on t and $\overset{*}{\tau}$, when analyzed as a function of $log(t)$, the unimodal basis functions are equidistant and equally wide, providing uniform support over $log(t)$ axis. This result is analogous to the scale-invariance observed in human timing and perception, formalized as the Weber-Fechner law. Intuitively, this is beneficial since the more recent past carries more predictive power than the more distant past. Hence, our approximation of function $f(0 < t' < t)$ will be better for values closer to t than to 0. Note that fixing the values of $\overset{*}{\tau}$ and choosing k also fixes values of s since $s = k/\overset{*}{\tau}$ so s is not a trainable parameter.

A.2 Derivation of the connectivity matrix for the inverse Laplace transform

The inverse Laplace transform is performed using the Post inversion formula [19]:

$$\tilde{f}(\overset{*}{\tau};t) = \mathbf{L}_k^{-1}F(s;t) = \frac{(-1)^k}{k!}s^{k+1}\frac{d^k}{ds^k}F(s;t),$$

where $\overset{*}{\tau} := k/s$. To implement this equation in a neural network we construct a discrete approximation for $\frac{d^k}{ds^k}$. First we compute N by N linear operator D which approximates the first order derivative: $\frac{d}{ds}$ (Alg. 2) and then raise D to power k to implement k-th order derivative: $D^k \approx \frac{d^k}{ds^k}$.

Algorithm 1 Constructing N by N linear operator D which approximates the first order discrete derivative.

$D \leftarrow zeros(N, N)$
for $i \leftarrow 1$ to $N-1$ **do**
$D[i, i-1] \leftarrow -\frac{s[i+1]-s[i]}{(s[i]-s[i-1])(s[i+1]-s[i-1])}$
$D[i, i] \leftarrow \frac{\frac{s[i+1]-s[i]}{s[i]-s[i-1]} - \frac{s[i]-s[i-1]}{s[i+1]-s[i]}}{s[i+1]-s[i-1]}$
$D[i, i+1] \leftarrow \frac{s[i]-s[i-1]}{(s[i+1]-s[i])(s[i+1]-s[i-1])}$
end for

A.3 Computing coefficient of variation of $\tilde{f}$

$\tilde{f}$ has a unimodal impulse response with peak at $t = \overset{*}{\tau}$. The coefficient of variation, c, is a ratio of standard deviation and mean. The mean of $\tilde{f}$ is:

$$\begin{aligned}
\mu &= \int_0^\infty t\tilde{f}(s;t)dt \\
&= \int_0^\infty t\frac{1}{t}\frac{k^{k+1}}{k!}\left(\frac{t}{\overset{*}{\tau}}\right)^{k+1} e^{-k\frac{t}{\overset{*}{\tau}}}dt \\
&= \frac{k^{k+1}}{k!}\int_0^\infty \left(\frac{t}{\overset{*}{\tau}}\right)^{k+1} e^{-k\frac{t}{\overset{*}{\tau}}}dt \\
&= \overset{*}{\tau}\frac{k+1}{k}.
\end{aligned}$$

The standard deviation of $\tilde{f}$ is:

$$\begin{aligned}
\sigma &= \sqrt{\int_0^\infty (t-\mu)^2\tilde{f}(s;t)dt} \\
&= \sqrt{\int_0^\infty (t-\mu)^2\frac{1}{t}\frac{k^{k+1}}{k!}\left(\frac{t}{\overset{*}{\tau}}\right)^{k+1} e^{-k\frac{t}{\overset{*}{\tau}}}dt} \\
&= \overset{*}{\tau}\frac{\sqrt{k+1}}{k}.
\end{aligned}$$

Finally, the coefficient of variation is then:

$$c = \frac{\sigma}{\mu} = \frac{1}{\sqrt{k+1}}.$$

The coefficient of variation depends only on k. Therefore the variance grows with the peak time. In other words, variance is constant as a function of the logarithm of the peak time.

A.4 Visualizaion of neural dynamics in CNL

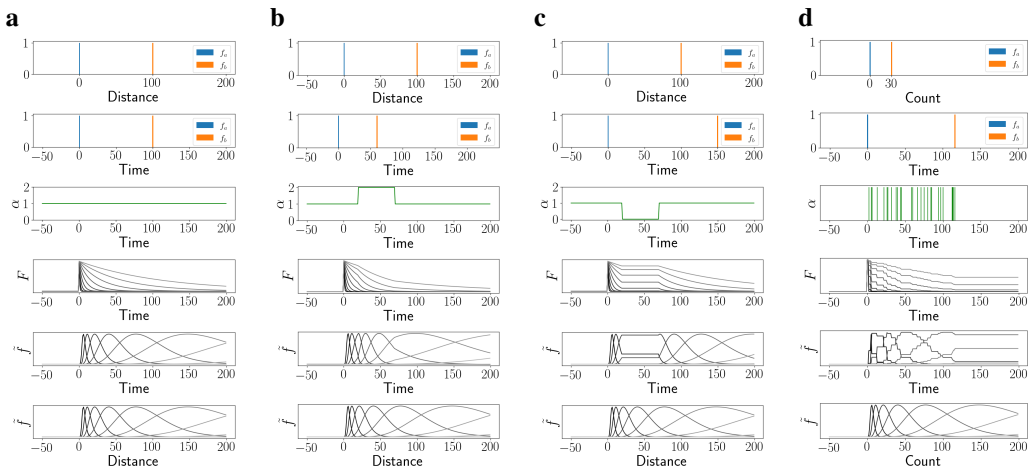

Figure A1: Neural dynamics with and without modulatory inputs. Delta pulse in f_a predicts a delta pulse in f_b. **a** No temporal modulation (*e.g.*, $\alpha = 1$). **b** and **c** When the modulatory signal corresponds to velocity (Experiment 1), the network represents traveled distance as manifested by units that activate sequentially as a function of distance (bottom row). **d** Temporal modulation with delta pulses (Experiment 2). The network represents the total count of presented pulses.

A.5 Illustration of Experiment 1

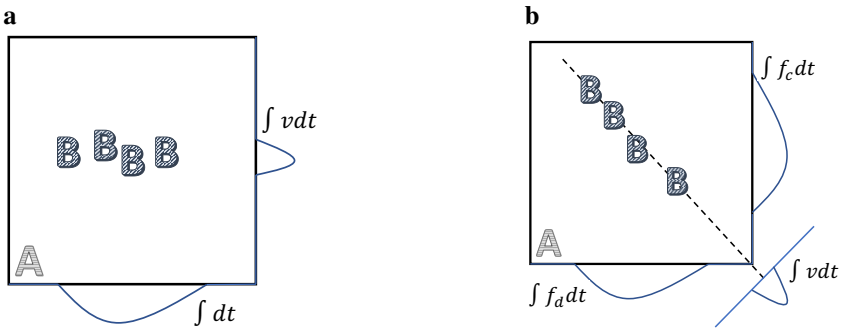

Figure A2: Illustration of predicting distance by learning latent modulatory inputs, as highlighted in Experiment 1. **a.** Let us assume that we want to estimate traveled distance from location A to location B. If we integrate time steps while traveling from A to B (x axis) then the estimate would not be accurate (broad distribution shown in blue) since our velocity changes during the trip. However, if we integrate velocity (y axis) then the estimate would be more accurate (narrow distribution shown in blue) and subject only to error in the velocity estimate itself. **b.** If velocity is not directly observable, it needs to be estimated as a combination of observable inputs, f_c and f_d as in Example 1 (for example, perhaps we are riding a bicycle and only observe motor outputs and sensory inputs - those would correspond to f_c and f_d here). If we use only f_c or f_d, our estimate of traveled distance will not be accurate (broad distributions shown in blue). However if we find a correct combination of f_c and f_d (in this simple illustration that is $f_c + f_d$) then the velocity estimate is more accurate (narrow distribution shown in blue) and subject again only to error in the velocity estimate itself.

```
Target Pattern
01001000001000000110

Input Data            S  T
00001111001001001011  0  0
11100100100100110001  0  0
01011001001001001111  0  0
01011110100000000010  0  0
01111101011111101001  1  0
11101000001001000111  0  0
11100110111001110111  0  0
10100000110000101100  0  0
11000000010010001010  0  0
01001011011100010111  0  0
01011101100000001110  0  0
01011011011011010110  0  1
10011111001111101001  0  0
01001101111110010111  0  0
```

Figure A3: Example output from the Counting dataset used in Experiment 2 (duration and size of the pattern are shortened for illustrative purposes). Input Data columns represent modulatory inputs, and rows represent time steps. In this example, the five rows highlighted in blue contain the target pattern, with the red portions being the target pattern itself. The target number of pattern occurrences is 3, corresponding to the number of patterns between the signal to start counting "S" (fed into the network as f_a) and the hidden target "T".

A.6 Binary cross-entropy loss in Experiment 1

	x'=50	x'=500	x'=5000	Params
CNL-F	0.537 ± 0.171	0.585 ± 0.006	0.633 ± 0.047	**53**
CNL	$\mathbf{0.275 \pm 0.007}$	$\mathbf{0.277 \pm 0.004}$	$\mathbf{0.336 \pm 0.018}$	**53**
RNN	0.578 ± 0.489	0.574 ± 0.316	0.575 ± 0.157	4481
LSTM	0.637 ± 0.349	0.677 ± 0.179	0.617 ± 0.179	17729
GRU	0.653 ± 0.098	0.659 ± 0.131	0.594 ± 0.096	13313
coRNN	0.611 ± 0.093	0.593 ± 0.071	0.509 ± 0.019	8513
LMU	0.645 ± 0.161	0.592 ± 0.258	0.521 ± 0.350	12740

Table A1: Experiment 1: Binary cross-entropy loss across different scales.

A.7 Visualization of representative examples in each experiment

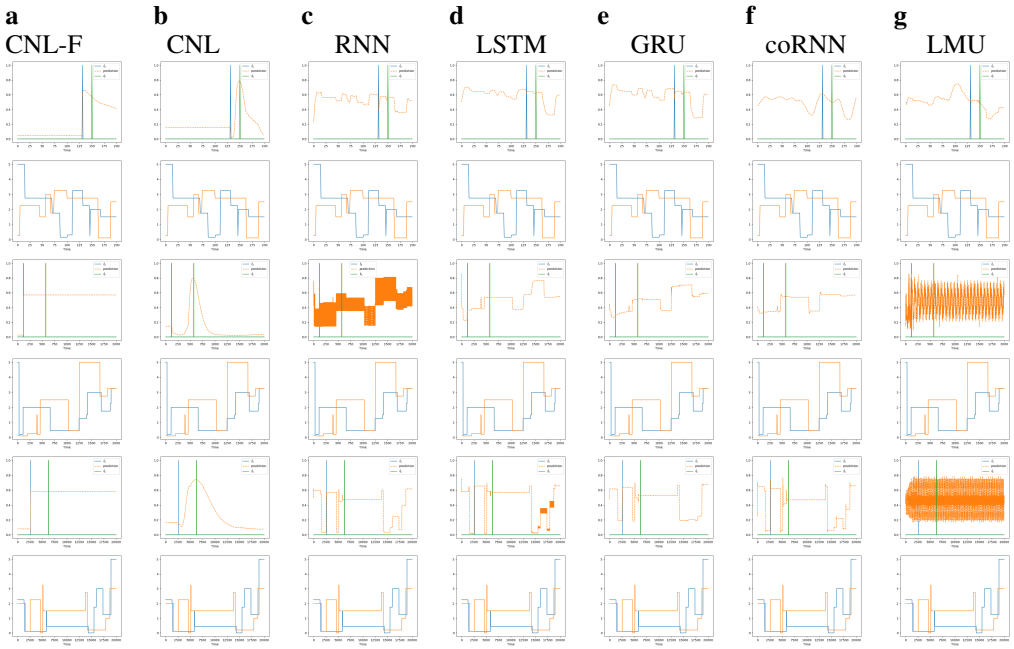

Figure A4: Experiment 1: Odd rows: representative examples of prediction (orange lines) for different models. Input f_a is shown in blue and target f_b is in green. Even rows: Modulatory inputs f_c and f_d. Plots are shown for three temporal scales. Top two rows: $x' = 50$, third and fourth row: $x' = 500$ and bottom two rows: $x' = 5000$.

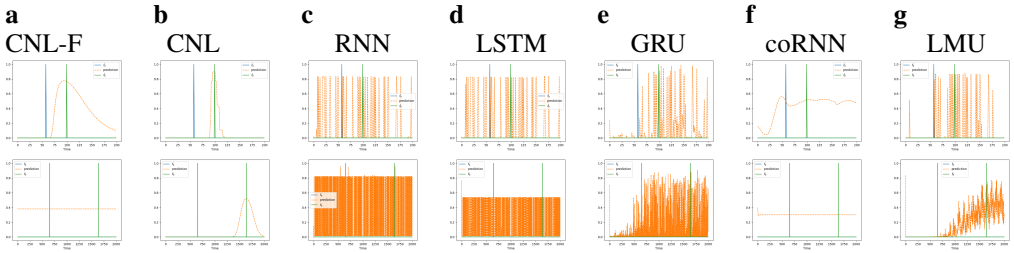

Figure A5: Experiment 2: Top row: representative examples of prediction (orange lines) for different models. Input f_a is shown in blue and target f_b is in green. Plots are shown for two temporal scales. Top row: $f_b = 1$ after 10 counts, bottom row: $f_b = 1$ after 200 counts.