

1 **Credit Assignment via Behavioral Timescale Synaptic Plasticity: Theoretical
2 Frameworks**

3 Ian Cone^{1,2*}, Claudia Clopath^{2#}, Rui Ponte Costa^{1#}

4 **1** Centre for Neural Circuits and Behaviour, Department of Physiology, Anatomy and
5 Genetics, University of Oxford, Oxford, United Kingdom

6 **2** Department of Bioengineering, Imperial College London, London, United Kingdom

7

8 *Email: ian(dot)cone(at)dpag(dot)ox(dot)ac(dot)uk

9 #co-senior

10 **Abstract**

11 Behavioral Timescale Synaptic Plasticity (BTSP) is a form of synaptic plasticity in which
12 dendritic Ca^{2+} plateau potentials in hippocampal pyramidal neurons drive rapid place field
13 formation. Unlike traditional learning rules, BTSP learns correlations on the timescales of
14 seconds and rapidly changes single-unit activity in only a few trials. To explore how BTSP-
15 like learning can be integrated into network models, we propose a generalized BTSP rule
16 (gBTSP), which we apply to unsupervised and supervised learning tasks, in both feedforward
17 and recurrent networks. Unsupervised gBTSP mirrors classical frameworks of competitive
18 learning, learning place field maps (in the feed-forward case), and attractive memory
19 networks (in the recurrent case). For supervised learning, we show that plateau events can
20 reduce task error, enabling gBTSP to solve tasks such as trajectory matching and delayed
21 non-match-to-sample. However, we find that credit assignment via gBTSP becomes harder
22 to achieve with increased network depth or CA3-like recurrence. This suggests that additional
23 features may be needed to support BTSP-mediated few-shot learning of complex tasks in the
24 hippocampus.

25

26 Introduction

27 Recent experimental observations have revealed the existence of a novel plasticity
28 phenomenon occurring in hippocampus, termed "Behavioral Timescale Synaptic Plasticity"
29 (BTSP). BTSP occurs in hippocampal pyramidal cells following strong, dendritic "plateau
30 potentials", and has been observed to be a "primary" driver of field formation (such as place
31 fields) in both hippocampal areas CA1¹⁻³ and CA3⁴. Unlike other established learning rules,
32 BTSP operates over a wide temporal range, potentiating and depressing inputs which were
33 active seconds before or after a postsynaptic "plateau" event. A similarly distinctive feature
34 of BTSP is its rapid learning speed - once triggered, it can form long-lasting hippocampal
35 fields in a one-shot or few-shot manner.

36 While the discovery of BTSP has advanced our understanding of single-cell learning in
37 hippocampus, we are still lacking a comprehensive theoretical framework to understand how
38 BTSP may contribute to network level plasticity. For example, experiments have shown that
39 inputs from entorhinal cortex layer 3 (EC3) are necessary to trigger plateau potentials in CA1,
40 and as such, have been hypothesized to act as a sort of "target signal"⁵ which guides plateau
41 generation. But what sort of "targets" should EC3 produce? That is, "when" and "where"
42 should BTSP-triggering plateau events occur for hippocampal learning to be successful? Or,
43 from a more general perspective, "when" and "where" should BTSP events occur to optimize
44 a network's function?

45 Furthermore, how can we reconcile the hallmarks of BTSP (wide temporal kernel and few-
46 shot learning) with traditional learning frameworks (particularly supervised ones)⁶⁻¹³, for
47 which small learning rates and temporally precise credit assignment are generally required
48 for convergence? Does the presence of BTSP put constraints on the possible tasks and the
49 neural architectures (i.e. circuit designs) which can learn them?

50 Previous theoretical work on BTSP has described in detail its single-cell properties^{1,2,14} and
51 consequences in memory networks^{4,15,16}. However, such work has thus far have been forced
52 to assume specific, hand-tuned plateau induction protocols. In contrast, this work aims to
53 formulate BTSP in such a way that we can describe where and when post-synaptic plateaus
54 should occur such that the network learns a given unsupervised or supervised objective.

55 Towards this goal, we formulate a generalized BTSP rule which can a) match existing
56 experimental data, and b) give us analytical, differentiable expressions for how the learning
57 performance (i.e. the loss) depends plateau events. We take this rule and demonstrate its
58 ability to learn in both feed-forward and recurrent networks, on both supervised and
59 unsupervised tasks. Since we derive an analytical expression for plateau "function", we can
60 predict the occurrence of plateau events, given that we know the weights, inputs and task.
61 Further, by applying constraints on our expression for plateau events, we can approximate
62 the sparse nature of these events in vivo. However, we show that the rapid, one-shot
63 formation of single fields associated with BTSP runs into critical stability issues when applied
64 in deep/recurrent networks, because of exploding and vanishing mathematical terms. We
65 finish by discussing how potential architectures and tasks may be able to avoid this issue,
66 and what the implications are for our understanding of hippocampal networks.

67 Altogether, this work provides a unified, analytical framework for understanding BTSP in
68 relation to network-level learning, establishing a theoretical foundation through which we can
69 explore how this unique form of plasticity can be integrated into hippocampus-mediated
70 learning processes.

71 Results

72 **Generalized BTSP Rule Recapitulates Experimentally Observed Plasticity Kernels**

73 We begin by building a generalized learning rule, based on experimental observations of
74 BTSP. For clarity, we will hereafter refer to our rule as “gBTSP” (generalized BTSP, **Figure**
75 **1a-c**) and refer to the experimental phenomena as simply “BTSP”. To start, consider the
76 simple case of a single postsynaptic neuron which triggers an instantaneous plateau event,
77 and a single presynaptic neuron which fires a spike (**Figure 1a**). We assume the postsynaptic
78 plateau updates weight W via some function, W_{kernel} , which depends on the timing of the
79 presynaptic spike relative to the plateau, i.e. $\Delta W \propto W_{kernel}(t_{pre} - t_{plateau})$.

80 Owing to the wide temporal window in which plateau potentials have been observed to
81 potentiate and depress inputs¹⁻³, we assume that W_{kernel} operates on a timescale much
82 larger than that of mere pre-post activity correlations. Specifically, we choose a W_{kernel} such
83 that the application of our learning rule matches observed plasticity following application of a
84 single plateau and bursting inputs *in vitro* (**Figure 1d**)². Next, we relax our previous
85 assumption that there is a single presynaptic neuron which fires a single spike, instead
86 considering continuous presynaptic activity (of each unit j), $x_j(t)$ (**Figure 1b**). Now, the
87 change in weights following a single plateau is a function of both the weight kernel and the
88 presynaptic activity, i.e. $\Delta W_j \propto f(W_{kernel}(t - t_{plateau}), x_j(t))$. See Methods for full derivation
89 and expression.

90 To match experimental data showing that the amplitude of the formed field depends on the
91 initial membrane voltage of the postsynaptic cell¹, we also add in a dependence on the
92 synaptic strength prior to the plateau event (see Methods). Following these additions to our
93 rule, we can now use the same W_{kernel} from **Figure 1d** and show that for place field-like
94 inputs $x_j(t)$, our rule recapitulates the observed plasticity kernels measure from single
95 plateaus *in vivo* (**Figure 1e**)¹. With this framework, the previously observed asymmetric offset
96 of observed plasticity (**Figure 1e**)¹ is a direct consequence of the shape of W_{kernel} **Figure 1d**
97 (see Methods).

98 Finally, we want to consider the case for which there are multiple postsynaptic neurons, each
99 of which may have a plateau (or potentially multiple). So, we introduce $P_i(t)$, a function
100 representing the post-synaptic plateau potential at time t for neuron i . Critically, this post-
101 synaptic plateau $P_i(t)$ is used only for learning and is distinct from the post-synaptic network
102 activity, $y_i(t)$. Now, the change in the weights will depend on the weight kernel, the
103 presynaptic activity, synaptic strength, and post-synaptic plateaus, i.e. $\Delta W_{ij} \propto f(W_{kernel}(t -$
104 $t_{plateau}), x_j(t), W_{ij}, P_i(t))$ (**Figure 1c**).

105 The full form of this dependence is given by the following equation, which we will hereafter
106 refer to as “generalized” BTSP, since it is derived from various steps of generalization from
107 our initial fundamental assumption ($\Delta W \propto W_{kernel}(t_{pre} - t_{plateau})$):

$$108 \quad \Delta W_{ij} = \int_0^T P_i(t) \left[\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij} \right] dt \quad (1)$$

109 See Methods for full details and derivation. From this gBTSP equation, we can now take any
110 set of inputs, choose any kernel, apply any arbitrary distribution of plateaus, and obtain a
111 resulting change in the network’s synaptic weights. Note that gBTSP (as with BTSP) does
112 not depend on postsynaptic activity directly, distinguishing it from standard Hebbian and
113 Hebbian-like (e.g. STDP) learning rules.

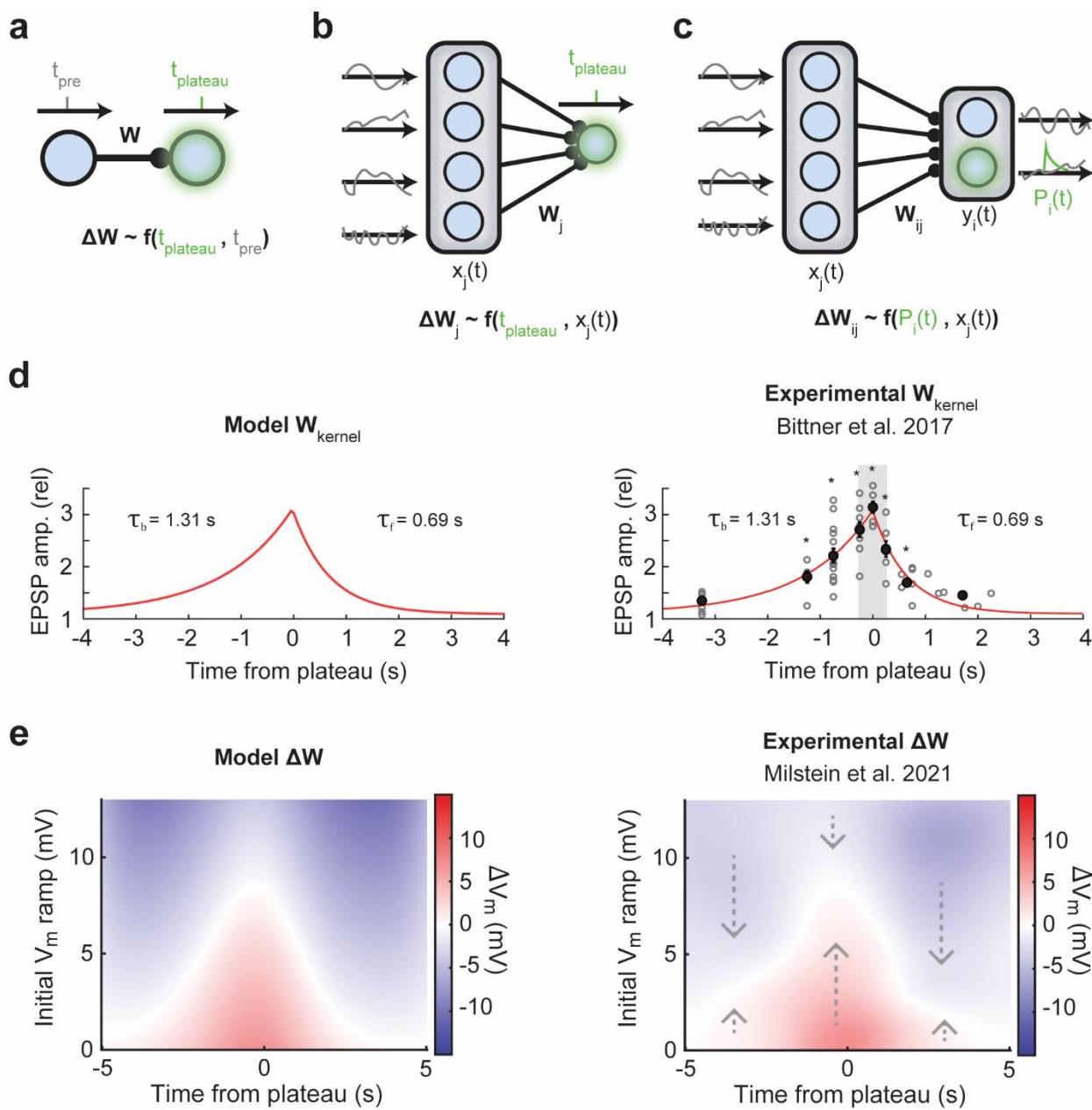


Figure 1 – Generalized BTSP recovers experimentally observed plasticity kernels

(a-c) Schematics of different BTSP induction setups. **a)** A single plateau occurs in a postsynaptic neuron, and a single spike occurs in presynaptic neuron. Weight changes ΔW depend on some function of the relative time between the plateau and the presynaptic spike. **b)** A single plateau occurs in a postsynaptic neuron at t_{plateau} , but now presynaptic neurons have some activity $x_j(t)$. Weight changes ΔW can be described as some function of the weight kernel and presynaptic activity. **c)** Potentially continuous plateau activity $P_i(t)$ occurs in the postsynaptic population. The resulting plasticity (“generalized BTSP” or “gBTSP”) depends on the weight kernel, presynaptic activity, and the postsynaptic plateau activity. **d)** Left, the kernel in our model uses two decaying exponentials, each with a different time constant. Right, the experimentally observed kernel, reprinted with permission from Bittner et al. 2017²; copyright AAAS. **e)** Left, the ΔW in our model when using the weight kernel from panel d) and a set of place fields (putatively from CA3) as presynaptic inputs. Right, observed ΔW in vivo, reprinted with permission from Milstein et al. 2021¹; CC BY 4.0.

115 Given this mathematical description of our learning rule, we now seek to gain a deeper
116 understanding of its function. How does it operate inside of a network? What types of learning
117 tasks is it well or poorly suited for? We will now investigate the properties of gBTSP in both
118 unsupervised and supervised contexts, for both feed-forward and recurrent networks.

119 ***Unsupervised gBTSP leads to competitive learning and one-shot field formation in***
120 ***feed-forward networks***

121 To understand how gBTSP operates in the simplest case, we first consider gBTSP as an
122 unsupervised rule. Returning to the simple case where we have a single plateau (and a short
123 temporal kernel), Equation 1 simplifies greatly (see Methods), giving us an expression for
124 plasticity of the form $\Delta W_j = x_j(t_{plateau}) - \lambda W_j$. So, in this approximation, upon each plateau
125 event, the weights would move towards a fixed point $W_j = \frac{1}{\lambda} x_j(t_{plateau})$. Such a formulation
126 is reminiscent of classical conceptions of “competitive learning”^{17–20}, whereby postsynaptic
127 neurons y_i “compete” with each other to encode a pattern x_j^p in its weights W_{ij} (where p is
128 the index of a particular pattern, and j indexes over the pattern’s components). The decay
129 term ($-\lambda W_j$) acts as heterosynaptic depression, promoting competition between units²¹.

130 Often, competitive learning is concerned with encoding *multiple* input patterns (or indeed, a
131 whole distribution of possible input patterns), using various forms of “competition” (via some
132 rule) to assign different postsynaptic neurons to represent distinct parts of the input space^{17–}
133 ²⁰. This algorithm has appealing aspects in the context of BTSP (and the hippocampus) –
134 when presented with an input distribution, competitive learning can quickly (few-shot for a
135 single unit) assign a unit to represent a part of that input space. Over the course of sampling
136 the input distribution, a3 population-level representation slowly emerges. We might consider
137 the hippocampus to be solving an analogous problem, e.g. forming a latent representation
138 which tiles a given input space, taking care to have both a) coverage over the whole space,
139 and b) well-separated or orthogonal latents which do not interfere with each other.

140 In order to adapt gBTSP for competitive learning, we must select a criterion for triggering
141 plasticity events. Commonly, competitive learning methods only apply the weight update to
142 the “best matching unit” (e.g. one which has a small Euclidean distance between x_j^p and
143 W_j)^{19,22}. If we followed that logic, we would only trigger plateau events for these “best
144 matching units”. However, this does not easily map onto learning in continuous time,
145 particularly when we consider our temporally extended weight kernel. Instead, we choose an
146 even simpler criterion, whereby a plateau event occurs in random neuron if the sum of total
147 postsynaptic network activity $\sum_i y_i(t)$ falls below some threshold θ (see Methods). In other
148 words, if $\sum_i y_i(t) < \theta$, we consider the currently arriving input $x(t)$ to be poorly represented
149 in the output layer $y(t)$. To amend this, the network fires a plateau, forming a new field (or
150 translocating an existing one) that is tuned to $x(t)$. To test this simple algorithm for plateau
151 assignment, we imagine a network of CA1 neurons to be receiving noisy but spatially tuned
152 input from CA3 neurons, as an agent traverses an environment (**Figure 2a**). We consider
153 both 1D (modelling an animal on a treadmill) and 2D (modelling an animal freely moving in a
154 box) environments.

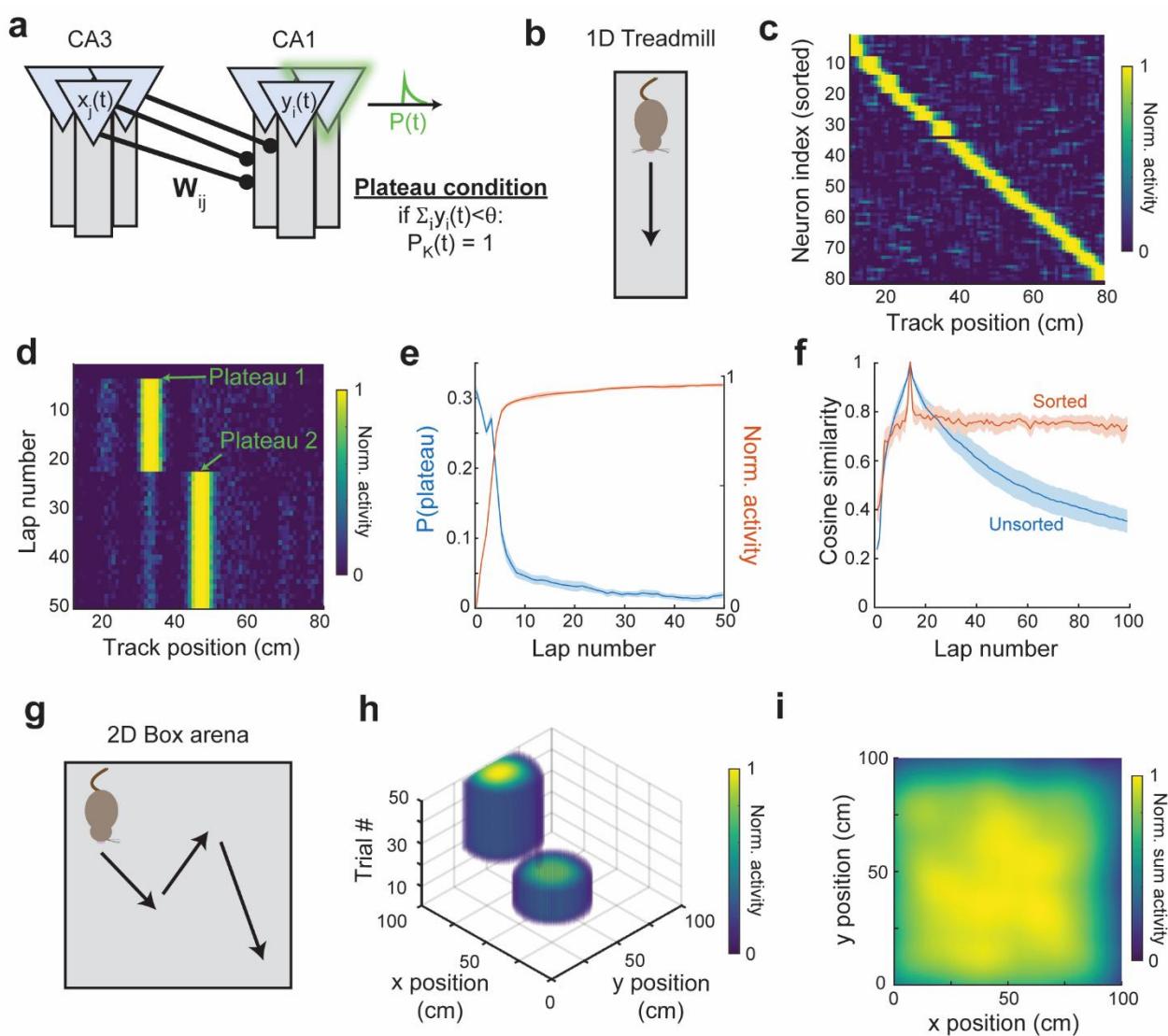


Figure 2 – Competitive learning via gBTSP allows for one-shot formation of fields which tile the input space

a) Modelled inputs from CA3 project to our model CA1 neurons through weights W_{ij} . A plateau fires at a random postsynaptic neuron when the sum of postsynaptic activity is below some threshold θ . **b)** For this task setup, inputs are drawn from a simulated agent running along a 1D treadmill. **c)** The learned fields across the population uniformly tile the 1D space. **d)** Using this plateau condition, single cells develop and translocate fields in a one-shot manner. For this particular unit, two plateau events occurred during training. **e)** The probability of a plateau event (blue line) peaks upon introduction to the novel environment, before decreasing to a baseline rate once a sufficient map has been learned. This time course is inverse to the total network activity (orange line). **f)** The baseline rate of plateau events (due to noise) causes representational drift. Blue, cosine similarity between unsorted network activity at the current lap and unsorted network activity at lap 10. Orange, cosine similarity between sorted network activity at the current lap and sorted activity at lap 10. Both measures peak at lap 10 because the cosine similarity of the activity at lap 10 with itself is 1. **g)** For this task setup, inputs are drawn from a simulated agent randomly exploring a 2D box. **h)** In a 2D environment, single fields still develop and translocate rapidly. **i)** Sum of neural activity for a trial where the agent explores the entire 2D environment. The learned latent representation covers the extent of the box.

156 For the case of a 1D environment, our agent moves along a treadmill at a uniform velocity
157 (**Figure 2b**), receiving spatially tuned inputs (see Methods), and applying a gBTSP plateau
158 every time the low activity condition ($\sum_i y_i(t) < \theta$) is met. After training, the population activity
159 has evolved to span the space of the inputs, forming place fields which tile the length of the
160 track (**Figure 2c**). Following the evolution of a single neuron in the network reveals that
161 plateaus can both form and translocate fields in a one-shot manner (**Figure 2d**). The full,
162 population level representation takes ~ 10 laps to mature, during which time there is a high
163 likelihood of plateau events (as the network fills in “blank” spaces in the representation). The
164 probability of plateau events scales inversely with the activity of the network, as we would
165 reasonably expect from our criterion for triggering plateaus (**Figure 2e**). For subsequent laps
166 after the network has evolved a mature representation (around lap 10), noise can still cause
167 our plateau condition to be triggered. This can cause the translocation of existing fields
168 (**Figure 2d**), also opening representational gaps that were previously filled. This effect leads
169 to representational drift in unsorted representations, whereby the cosine similarity between
170 the unsorted representation of the current lap and that of a reference lap (lap 10) increases
171 as a function of experience (**Figure 2f, blue**), as has been reported experimentally^{23,24}.
172 However, this does not mean the content of the representation is fading – if we instead
173 calculate the cosine similarity between sorted representation of the current lap and the sorted
174 representation of a reference lap (lap 10), this measure remains stable over experience
175 (**Figure 2f, orange**). This reveals that most of the representational drift occurring in the
176 network is index-related, i.e. neurons may shift their tuning (or “label”) and “shuffle” where
177 they occur in the sequence, but the internal, population-level sequential structure is
178 maintained (**Figure 2c**)²⁵.

179 We can extend further to a 2D environment (**Figure 2g**), where an agent takes a random
180 walk inside a box, again receiving spatially tuned but noisy inputs (see Methods). Unlike the
181 case of the 1D treadmill, where each lap the animal encountered the exact same input, here,
182 the animal’s random walk means that it will experience a unique sequence of inputs each
183 trial. Over the course of training, individual cells develop characteristic 2D place fields which
184 evenly tile the space. Place field emergence in single-cells is still one- to few-shot, even in
185 the 2D case (**Figure 2h**). We make the agent traverse the entire environment after training
186 and find that the sum of network activity provides a map which covers the extent of the box
187 environment (**Figure 2i**).

188 In summary, we find that for unsupervised learning in feed-forward networks, our
189 mathematical formulation of BTSP can be mapped onto the classical framework of
190 competitive learning. By applying gBTSP in simulated environments, we find that our network
191 acts as we would expect from a competitive learner, taking a high-dimensional input space
192 and summarizing it with a discrete set of lower-dimensional latent states. If BTSP indeed
193 follows a simple threshold principle for competition, our model would predict that plateau
194 probability across a network should be inversely proportional to that network’s activity (**Figure**
195 **2e**). Further, our model predicts that representational drift for a given learned neural trajectory
196 is mostly a consequence of a “musical chairs-like” resorting, whereby transient and stochastic
197 dips in total network activity in one location are likely to trigger a translocating plateau event,
198 leading to a dip in network activity in the translocated field’s previous location (**Figure 2d-f**).

199 ***Unsupervised gBTSP can facilitate attractor learning in recurrent networks***

200 Given that BTSP has been observed in CA3, driving plasticity in recurrent CA3→CA3
201 synapses⁴, we now consider how unsupervised gBTSP might be understood in the context
202 of a recurrent network. A common model of CA3 is that of an attractor network^{26–30}, where
203 an “attractor” can be framed in the context of discrete memory states (e.g. a Hopfield
204 model)^{31,32}, or a continuous manifold^{27,33–35}. Experimental results have demonstrated CA3’s

205 ability to both pattern complete and tune its activity via velocity-dependent inputs^{4,36,37},
206 hallmarks of a (recurrent) attractor network. As such, it is reasonable to suspect BTSP may
207 be involved with the formation or maintenance of these networks. Indeed, previous theoretical
208 work has shown that the BTSP rule's characteristic kernel is well suited for optimal memory
209 storage in discrete memory networks⁴, but it remains unclear if/how an unsupervised form of
210 BTSP can give rise to attractors.

211 In order to simplify our problem, we will utilize a two-part architecture in our network, inspired
212 by similar parametrizations of recurrent nets designed to learn or sustain attractors^{38–41}. In
213 short, we imagine there to be two distinct populations in CA3, with only one of the populations
214 eligible to receive plateaus via gBTSP. This assumption is based on experimental results
215 which have shown that the ability or propensity of pyramidal cells in CA3 to have complex
216 bursting events (i.e. a plateau) is variable from cell to cell, and may depend on features such
217 as topographic position and/or dendritic morphology^{42,43}. Both of our populations (“visible”
218 neurons $u_j(t)$ and “seed” neurons $s_i(t)$) connect recurrently to each other, via “encoding”
219 weights \mathbf{W}^e and “decoding” weights \mathbf{W}^d , with the visible neurons receiving external input
220 $o_k(t)$, and only the seed neurons are eligible for plateaus (**Figure 3a**). The relative strength
221 of recurrent/external input onto the visible neurons is governed by a gating function which
222 depends on the norm of the external input (see Methods): when external input is high,
223 recurrent input is low, and vice versa. Owing to this gating, high external input effectively
224 turns the network into a feed-forward one, and when input is removed, the network restores
225 its recurrency. This dual nature of the network allows us to take advantage of recurrent
226 computation in the low-input phase, while making use of unsupervised, feed-forward gBTSP
227 in the high-input phase. For full details, see Methods.

228 Since our aim is to learn attractor states (i.e. $\mathbf{u}(\mathbf{t}) = \mathbf{u}(\mathbf{t} - \mathbf{1})$ for no input), we would like our
229 effective recurrence, $\mathbf{W}^{rec} = \mathbf{W}^d \mathbf{W}^e$ to be approximately to the identity matrix. To avoid trivial
230 solutions, we make two choices. First, we set the seed population to be smaller (in number)
231 than the visible population, forcing the network to compress and then decompress its
232 representations (this is equivalent to making \mathbf{W}^{rec} low-rank). Second, we apply the same
233 competitive learning framework from the feed-forward case ($\sum_i s_i(t) < \theta$) to learn the
234 encoding weights, so that the seed neurons learn latent representations which tile the input
235 space. The decoder weights are set to be the transpose of the encoder weights, which is
236 sufficient since the learned encoding is well-separated (near orthogonal). Ideally, some
237 biophysically plausible learning rule can govern the evolution of these decoder weights⁴⁴, but
238 for the purposes of this study, we use the transpose relationship as a simple approximation.

239 We simulate an agent running along a 1D treadmill, again receiving spatially selective inputs
240 which are processed by our model CA3 network (**Figure 3a**). The seed neurons are allowed
241 to plateau, doing so under the same low-activity criteria as in the feed-forward case. During
242 training, the agent runs along the track and external inputs are strong. Plateau events occur
243 in response, guiding the evolution of the encoding weights (and thereby the decoder weights).
244 Seed neurons form receptive fields to their visible neuron counterparts (**Figure 3b**), similar
245 to the formation of place receptive fields in the purely feedforward case (**Figure 2c**). After just
246 the first lap of training, the recurrent weights have formed a ring topology, with fixed point
247 nodes (memories) at locations dictated by the plateau events (**Figure 3c**). This topology is
248 well explained by the first two principal components (**Supplemental Figure 1**). So long as
249 external input continues, the network remains largely feed-forward, but upon removal of
250 external inputs, the network is dominated by recurrency, and relaxes into one of its learned
251 fixed points. If partial and intermittent inputs are given, the network can switch between these
252 encoding (feed-forward) and recall (recurrent) modes repeatedly, recovering a new memory
253 each time it samples its inputs (**Figure 3d**).

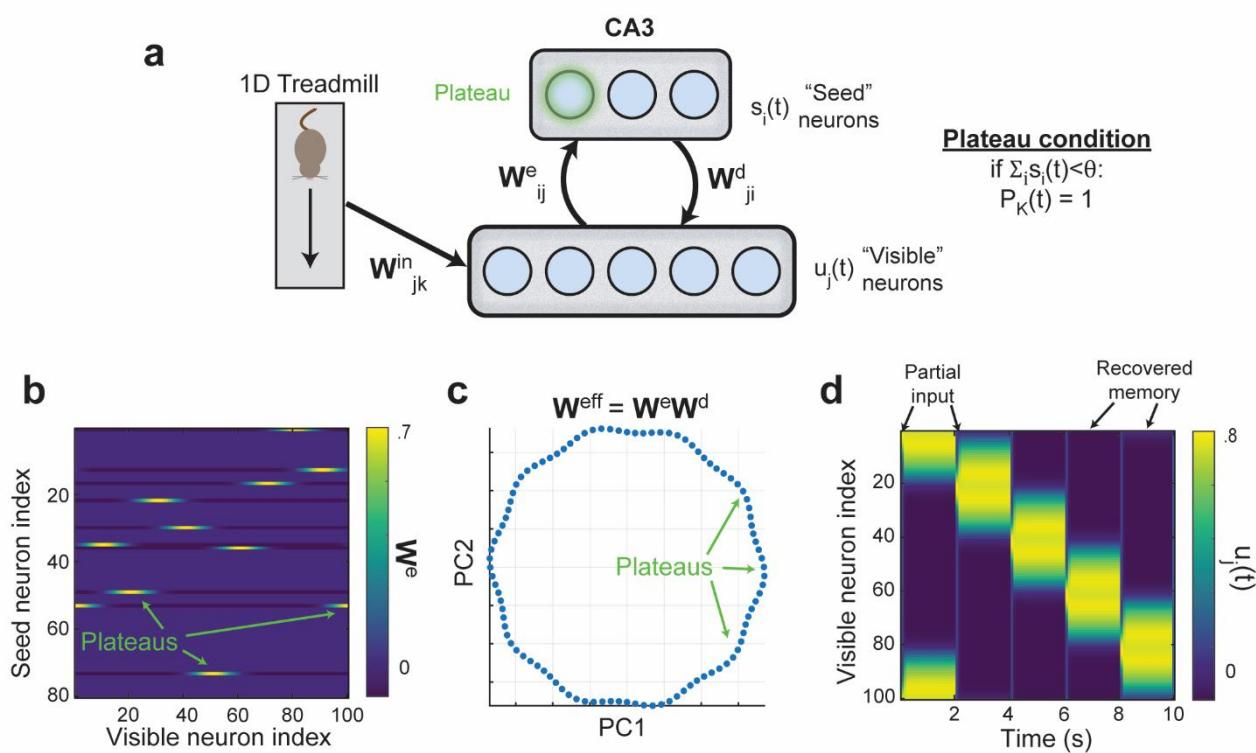


Figure 3 – Building an attractor network via gBTSP.

a) A set of spatially tuned noisy inputs are drawn from a simulated agent running along a 1D treadmill. These inputs project to “visible” neurons through a set of input weights. Visible neurons project to “seed” neurons via encoding weights, and seed neurons project back to visible neurons through decoding weights. There are no recurrent weights within each layer. This produces an effective recurrent weight matrix $W^{rec} = W^e W^d$. A plateau fires at a random seed neuron when the sum of postsynaptic activity is below some threshold θ . **b)** Plateau events create seed neurons which are sensitive to certain combinations of visible neurons. **c)** The effective recurrent weight matrix, W^{rec} , forms a ring attractor, as viewed in the first two principal components. Each of the nodes along the attractor is a fixed point created by a plateau event. **d)** Partial inputs given to the network for a single timestep at times 0,2,4,6, and 8 seconds recover a memory state of the network, which persists until the next partial input is presented.

254 Altogether, by using a low-rank formulation of our network recurrency, and gating plateau
 255 events to occur in a certain subpopulation of our network, we demonstrated that gBTSP can
 256 play a crucial role in the rapid formation of an attractor network. Such a role would be
 257 consistent with observations of BTSP in CA3¹⁰, a region often hypothesized to play the role
 258 of an attractor^{26–30}. Future experimental and theoretical work can further illuminate the
 259 functional structure of CA3 and the role BTSP plays in forming and maintaining attractor
 260 states.

261 **Supervised gBTSP can support rapid task learning in feed-forward networks**

262 While competitive learning provides an unsupervised framework by which we might
 263 understand the function of BTSP during novel, unguided exploration, it still leaves
 264 unanswered what role direct supervisory credit assignment might play. A popular hypothesis
 265 for BTSP posits that EC3 inputs to the distal dendrites act as supervisory “targets”, which in
 266 turn trigger plateau events so that the somatic activity can match this dendritic target⁵.
 267 However, it is not clear what these targets are, i.e. “when” and “where” should a plateau event
 268 occur? To rephrase the question in a more quantifiable way: if we define a given loss \mathcal{L} as
 269 the mean squared error between the network output $y_i(t)$ and some target $\hat{y}_i(t)$, when and

270 where should we trigger plateau events to minimize this loss? Equipped with our learning
271 rule, we have the tools to answer this question. To this end, we set our expression for ΔW_{ij}
272 from gBTSP (Equation 1) to be equal to expressions for ΔW_{ij} from traditional supervised
273 learning (in the simplest feed-forward case, the “delta rule”), and solve for $P_i(t)$ (see
274 Methods). In the case of a simple feed-forward network, that expression is the following:

$$275 \quad P_i(t) = \varepsilon_i(t) \sum_{j=1}^N \frac{x_j(t)}{\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij}} \quad (2)$$

276 Where is our $\varepsilon_i(t) = \hat{y}_i(t) - y_i(t)$ task error. This expression gives the plateau function $P_i(t)$
277 which will descend the loss gradient on a given trial. Equipped with this formula, we can now
278 test the ability of gBTSP to learn in supervised learning contexts.

279 As a sanity check, we first consider the trivial case where our inputs are already spatially
280 selective (such as those arriving from CA3)^{1,14} and our output represents a single CA1
281 pyramidal cell subject to gBTSP (**Figure 4a**). We choose a target function $\hat{y}(t)$ that is a
282 putative place field, modeled as a Gaussian bump centered at a specific location in the
283 environment (see Methods). We find that the network can match this target through plateau-
284 driven learning (**Figure 4b**), demonstrating the fundamental capability of gBTSP to adapt
285 network weights toward a desired output function (**Supplemental Figure 2**). The plateau
286 function which solves the task is, as expected, centered at the location of our target function,
287 and is most significant within the first 3-5 trials. The field itself also rapidly emerges on this
288 same timescale (**Figure 4c**), in agreement with experimental results where place fields were
289 formed via the artificial induction of plateaus a) at the location of the desired place field, and
290 b) over only a few (<10) trials¹⁻³.

291 Next, we test our ability to train the network on a more complicated task, in a network with a
292 single hidden layer which is subject to gBTSP. In this task, we model an agent learning to
293 match its location in a 2D arena to some target trajectory in that arena (indicated, say, by
294 targeted illumination). One can consider this task as a navigation-based analogue to smooth
295 pursuit or continuous reaching tasks. Rather than merely generating static spatial patterns,
296 the network now must take a dynamic input $x(t)$ and learn to generate a dynamic 2D position
297 output $y(t)$ (see Methods, **Figure 4d**). After the first 10 trials, the agent has learned to track
298 the target trajectory (**Figure 4e**). Unlike the previous example, where the plateau location
299 was obvious by design, here it is unclear a priori when and where plateaus should occur in
300 the hidden layer to solve the task. We find that a more complex pattern emerges for the
301 plateau function in a sample neuron, and there is no longer a simple correlation between the
302 network target and the shape of its plateaus (**Figure 4f**). This is because our expression for
303 the plateau function (Equation 2) will depend on the backpropagated error (see Methods) in
304 networks with more than one layer. Finally, our model predicts that, in a supervised
305 framework, the probability of plateau events in the full population should be inversely
306 correlated with task performance (or positively correlated with task error), decreasing over
307 the course of task learning (**Figure 4g**).

308 Together, these results demonstrate that we can use gBTSP to descend the gradient of a
309 supervised loss. In other words, the algorithm “distributes” plateau events to certain neurons
310 at certain times in order to optimize overall network performance. If EC3 does indeed dictate
311 plateau induction via supervised “targets”, as has been suggested⁵, then our framework
312 provides a computational tool to understand and potentially infer the content of these signals.
313 One testable prediction our supervised framework makes is that plateau probability should
314 rise and fall with the inverse of task performance (**Figure 4g**).

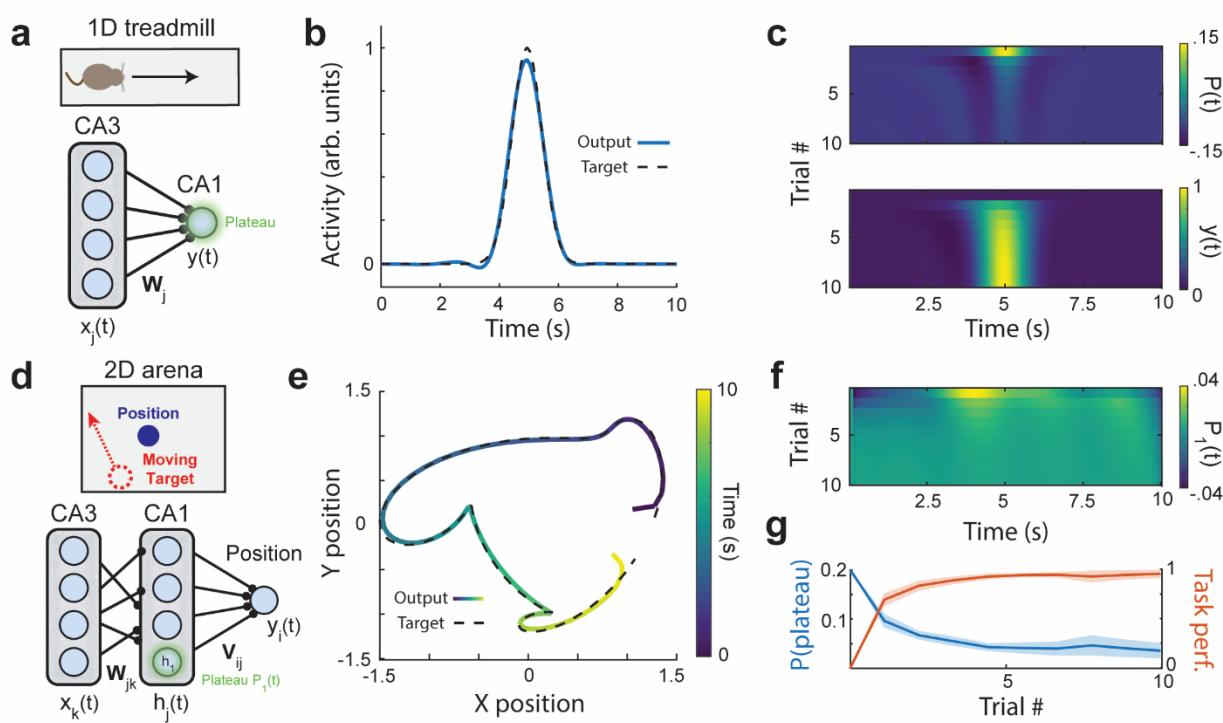


Figure 4 – Feed-forward network rapidly acquires task via supervised gBTSP.

a) Feed-forward network with inputs $x_j(t)$ project to output $y(t)$ via weights W_j . The inputs $x_j(t)$ are spatially selective and represent an animal running along a 1D track. **b)** A unimodal gaussian target function (dotted black line) and the trained output (blue line) after the first 10 trials of gBTSP training. **c)** Top, the plateau function $P(t)$ over the first 10 trials of training. Middle, the output $y(t)$ over the first 10 trials of training. Bottom, the output activity $y(t)$ over the first 10 trials of training. **d)** A two-layer network with inputs $x_k(t)$, hidden units $h_j(t)$, and 2D output $y_i(t)$ which represents location. The target trajectory moves in a set 2D path each trial, and the network must learn to track the target. **e)** The target path (dotted black line), and the learned path (colored line). The color here represents the time at which the agent is in a given location. **f)** Plateau function for the first unit in the hidden layer, $P_1(t)$, over the first 10 trials of training. **g)** Task performance (proportional to $1 - \varepsilon_i(t)$) and plateau probability over the first 10 trials of training.

315

316 **Supervised gBTSP in recurrent networks fails to support rapid, one-shot learning**

317 Finally, we wish to examine the feasibility of our rule in a fully recurrent network (to mimic
 318 CA3), learning a supervised task which requires maintenance of an internal memory (via
 319 recurrence). Our network consists of hidden units with activation $h_j(t)$, recurrently connected
 320 via weights W_{ji}^{rec} , and projected to output $y(t)$ via weights W_j^{out} (**Figure 5a**). Internal weights
 321 W_{ji}^{rec} are trained indirectly through plateau induction, which is dictated by a recurrent update
 322 rule, that we derive by comparing our gBTSP weight update to that of backpropagation
 323 through time (BPTT) (see Methods).

324 We illustrate the behavior of the model using a standard delayed-non-match-to-sample
 325 (DNMS) task, where our simulated agent must distinguish between sequential pairs of odors,
 326 and only “lick” in responses to non-matching pairs (AB, BA), refraining from licking following
 327 matching pairs (AA, BB) (**Figure 5a**). We choose this task because it requires the network to
 328 maintain a memory of the first odor’s identity (by leveraging recurrent learning). Moreover,

329 previous experimental work has shown that animals trained on the same task developed two
330 distinct hippocampal sequences of activity which encoded the identity of the first odor⁴⁵.

331 To avoid instabilities during training, we combine our update with an adaptive optimizer
332 (ADAM) before updating the weights of the network (see Methods)⁴⁶. We find that gBTSP
333 can learn the target function, choosing to “lick” when the two samples are non-matching, and
334 forgoing licking when the two samples match (**Figure 5b**). However, unlike the simpler tasks
335 we have thus far described, training a recurrent network on the DNMS task takes many
336 thousands of trials (**Figure 5c**). In order to solve the task, the network develops distinct
337 internal representations for the cases when Odor 1 = A, and when Odor 1 = B (**Figure 5d**).
338 These representations are well explained by their first three principal components, with Odor
339 1 = A trials and Odor 2 = B trials making distinct trajectories in this subspace (**Figure 5e**,
340 **Supplemental Figure 3**). These distinct representations act as a memory trace of the first
341 odor’s identity, thereby allowing the network to judge “match” vs. “no-match” upon
342 presentation of the second odor. These representations resemble neural sequences
343 observed in experimental studies⁴⁵, but as in previous theoretical work⁴⁷, we found that
344 adding a ramping component to the task target best recovered this sequential activity (see
345 Methods). The plateaus in the network which facilitate learning were constrained to be
346 stochastic and sparse, with only 10% of neurons allowed to plateau on a given trial, and only
347 events which crossed an absolute magnitude threshold contributing to learning (**Figure 5f**).
348 Though these constraints may result in single trial samples of $P(t)$ which share the sparse
349 nature of plateaus observed in vivo^{5,48}, when we examine the evolution of single cell fields,
350 we see that they develop very slowly, taking thousands of trials (**Figure 5g,h**). Another
351 alternative would be to increase learning rates, but doing so results in unstable learning
352 (**Supplemental Figure 4**). In short, spatiotemporal credit assignment in recurrent networks
353 is notoriously difficult⁴⁹, and unsurprisingly, solving for $P_i(t)$ via gBTSP (as opposed to
354 solving for W_{ij} directly via BPTT) does not bypass these limitations. In other words, the
355 standard tool of gradient descent does not stably resolve the question of “where” and “when”
356 BTSP events should occur in a recurrent network in order to solve a supervised task. We will
357 now elaborate on a more complete answer and discuss how we can reconcile these apparent
358 hard limits on the speed of learning with the existence of BTSP in CA3, a highly recurrent
359 network.

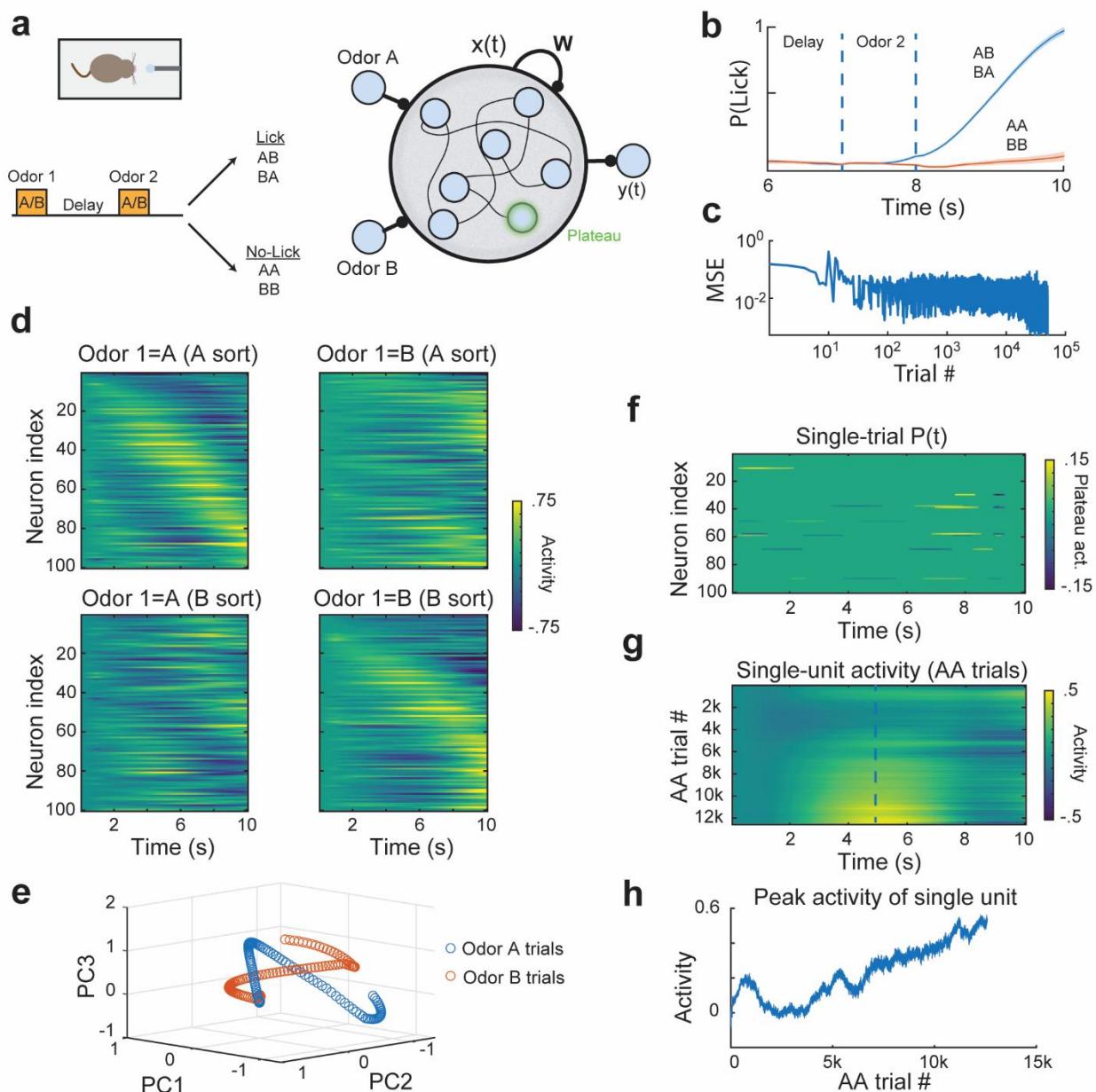


Figure 5 – Learning a complex recurrent task with gBTSP requires slow and precise credit assignment

a) Simulated agents are trained on a delayed-non-match-to-sample (DNMS) task where they must distinguish between sequential “odor” pairs. The agent must learn to “lick” for non-matching sequences (AB, BA), and refrain from licking for matching sequences (AA, BB). Right, the model consists of a recurrent network with two odor inputs, hidden activities $x(t)$ and recurrent weights W . The hidden units project to an output $y(t)$ via weights V . Recurrent weights W are trained via plateaus occurring in the hidden units according to our gBTSP algorithm (see Methods). Output weights are trained via the delta rule. **b)** After training, the model output has learned to lick following odor 2 in the AB/BA trials while refraining from licking in the AA/BB trials. **c)** Mean squared error (MSE) decreases over training, demonstrating successful learning, albeit across tens of thousands of trials. **d)** Neural activity patterns across all 100 neurons, averaged over trials which began with odor A (left column), or odor B (right column). The neurons are sorted by time of maximum activity in the odor A trials (top row), or time of maximum activity in odor B trials (bottom row). The activity maps reveal the network has learned distinct sequences of activity for the initial odors, thereby forming a working memory of the first odor’s identity. **e)** Activity of each trial type (odor A or odor B), projected onto the first 3 principal components of the total activity space after training. For variance explained, see (Supplemental Figure 3). **f)** Single-trial plateau events across neurons

and time, showing sparse activation. **g)** Evolution of activity for a representative single unit during AA trials over training. The final neural tuning slowly develops over many thousands of trials. **h)** Activity of the same representative unit at $t = 5$ seconds, across AA trials, again highlighting gradual field development.

360

361 **Rapid activity changes due to gBTSP are fundamentally limited in deep and/or**
362 **recurrent networks**

363 In shallow feed-forward networks, gBTSP could recover few-shot learning as observed
364 experimentally with BTSP. However, as we have demonstrated particularly for supervised
365 learning in the recurrent network, single-cell learning via gBTSP was very slow. Moreover,
366 attempting to speed-up learning results in instabilities (**Supplemental Figure 4**). Why is this?

367 For the following, we will step aside from the specifics of gBTSP to make a more general
368 formulation of the problem. Let us assume only that a) plateau events exist, and b), they
369 cause single-cell activity to change by a fixed amount Δx , remaining agnostic about the
370 type(s) of learning involved in bringing about this change Δx . We can consider learning via
371 these plateau events from the perspective of optimizing within a loss landscape, taking a step
372 Δx along the direction of the descending gradient (first derivative). In landscapes with “fine-
373 grained” or “sharp” features, a step of size Δx can overshoot the global minimum (**Figure 6a**).
374 Conventional learning approaches address this issue by reducing step sizes (i.e. taking a
375 step of size $\delta x < \Delta x$) (**Figure 6ai**), thereby allowing learning to converge to the minimum.

376 An alternative approach involves modifying the loss landscape itself. By “stretching” the
377 landscape, the same step size Δx becomes proportionally smaller relative to the landscape
378 features, “smoothing” out sharp features preventing overshooting (**Figure 6aii**).
379 Mathematically, this “stretching” operation locally shrinks both the first and second derivatives
380 of the loss with respect to activity. Since we are assuming each optimization step takes a
381 fixed step size Δx regardless of the gradient (first derivative) amplitude, we can focus on
382 conditions on the second derivative (which we will hereafter refer to, for simplicity, as the local
383 “curvature” C), and show how this curvature depends on features of the network. Intuitively,
384 learning in this “stretched” landscape might be considered akin to the evolution of microwave
385 popcorn, in the sense that while population-level representations (the popcorn bag) may
386 evolve gradually, individual units (the kernels) undergo rapid, stochastic transitions to their
387 final states on timescales significantly shorter than the overall system evolution. To support
388 this “popcorn”-like approach (which we posit to be more BTSP-like), local curvature must be
389 small (i.e. the loss must be locally “shallow”) to prevent overshooting, but non-zero to enable
390 learning in the first place.

391 In the case of a single-layer feedforward network, inputs $x_j(t)$ project to output $y_i(t) =$
392 $\sum_j W_{ij}x_j(t)$. (**Figure 6b**). “Plateaus” of size Δx occur at the inputs, and the loss is the mean
393 squared error between output y and target \hat{y} . Solving for the curvature, we find that $C \propto W^T W$
394 (see Methods). It is trivial enough to construct a network where $W^T W$ is small but non-zero.
395 For example, if $y_i(t)$ receives many inputs, each with a small weight W_{ij} , any change in a
396 single input $x_j(t)$ will lead to small change in $y_i(t)$. So, in the case of a single layer feed-
397 forward network, a small but non-zero curvature is achievable, meaning rapid changes in
398 activity arising from BTSP can lead to stable learning.

399 If we consider a deep feed-forward network with layers l and layer specific weights W_l
400 (**Figure 6c**), the expression for the curvature becomes more complicated, depending on
401 products of all the layer-specific weights together in sequence ($C \propto (\prod_{l=1}^L W_l)^T \prod_{l=1}^L W_l$).
402 Unfortunately, it is not trivial to make these products small but non-zero - in fact, they are the

403 same troublesome mathematical objects which lead to the problem of exploding and
 404 vanishing gradients in gradient descent^{49–51}.

405 Recurrent networks (**Figure 6d**) can be conceptualized similarly to deep feed-forward
 406 networks (**Figure 6c**), but with each “layer” representing a different timestep in the network,
 407 with the weight matrix W is applied at each timestep. In turn, curvature of the loss in a
 408 recurrent network depends on similar products ($C \propto (\prod_{t=1}^{t_0} W)^T \prod_{t=1}^{t_0} W$) which also lead to
 409 exploding and vanishing contributions from an update Δx .

410 These conditions set fundamental limits on both the architectures and the tasks for which
 411 deep or recurrent artificial networks can support rapid changes in single-unit activity. Despite
 412 these theoretical limitations, few-shot BTSP events have been observed in CA3⁴, which is
 413 highly recurrent. The analysis above is idealized, and biological neural networks may have
 414 yet unknown mechanisms which allow them to bypass these restrictions. However, if we
 415 hypothesize that recurrent connectivity in CA3 is indeed subject to these constraints,
 416 specialized architectural features would be required to stabilize learning dynamics and
 417 prevent vanishing or exploding effects arising from rapid plasticity events driven by BTSP.

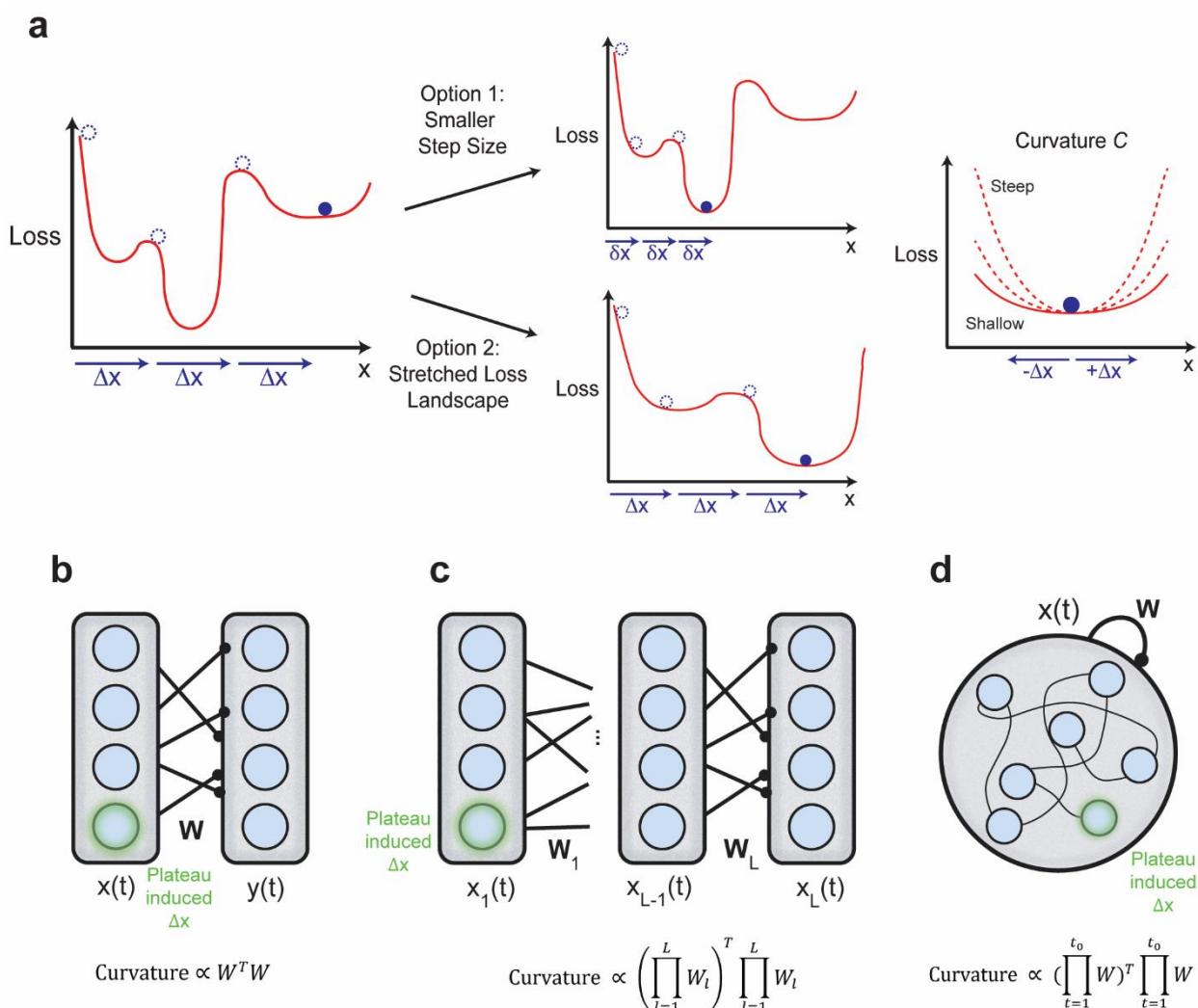


Figure 6 – Shallow losses are required for few-shot learning

a) Red, an arbitrary loss function $\mathcal{L}(x)$ which depends on the network state, x . In order to learn, the network is restricted to make discrete jumps of size Δx . Blue dotted circles, previous values of $\mathcal{L}(x)$. Filled blue circle, current value of $\mathcal{L}(x)$. Top middle, smaller discrete jumps of size δx are

sufficient to reach our learning objective, but results in a slow evolution of network activity over learning (“dough” learning, right). Bottom middle, stretching the loss function also allows us to reach our learning objective, while maintaining a fixed step size Δx . Single unit activities rapidly change over the course of learning, even if the population representation evolves slowly (“popcorn” learning). This stretching picture requires a shallow but non-zero curvature (right). **b)** Shallow feed-forward network, and its associated curvature. **c)** Deep feed-forward network, with L layers, and its associated curvature. **d)** Recurrent network which runs for T timesteps, and its associated curvature. Note that this picture can be related to that of the deep feed-forward network, if we imagine each “layer” to be the activity of the recurrent net at a time t , and the weights between these layers to be the shared weight matrix W .

418

419 **Discussion**

420 The discovery of Behavioral Timescale Synaptic Plasticity (BTSP) unearthed an apparent
421 paradox in our understanding of learning in the hippocampus. One the one hand, successful
422 models of complex population-level hippocampal function (e.g. the formation of cognitive
423 maps^{52–54}) depend critically on recurrent computation, and in turn, seem to depend on the
424 slow, precise training of recurrent weights. On the other hand, experiments in hippocampus
425 observe a learning rule (BTSP) which is very fast and has a very distinct lack of temporal
426 specificity. BTSP's hallmark features—its wide temporal kernel spanning seconds and its
427 rapid, one-shot field formation—run counter to conventional wisdom that precise, gradual
428 weight changes are necessary for stable learning. To examine the computational implications
429 of BTSP, we proposed a generalized mathematical framework, gBTSP, for which plasticity is
430 governed by wide temporal kernels and a postsynaptic “plateau function” $P(t)$. We test its
431 properties across different network architectures (feed-forward and recurrent) and learning
432 paradigms (supervised and unsupervised).

433 We demonstrated that unsupervised gBTSP in feed-forward networks maps well onto the
434 framework of competitive learning, wherein neurons “compete” to represent distinct regions
435 of the input space. This framework accounts for experimentally observed phenomena,
436 including the rapid formation of place fields and their distribution across the environment. If
437 we assume BTSP is operating according to unsupervised principles, our model predicts that
438 plateau probability should inversely correlate with network activity, offering a testable
439 hypothesis for future experiments. Moreover, we found that individual neurons can undergo
440 rapid remapping while the population-level representation maintains coverage of the
441 environment—exhibiting representational drift primarily through index “shuffling” rather than
442 degradation of the underlying representation. However, we only consider a few hundred trials
443 of unsupervised learning, so it remains unclear to what extent this model can explain recent
444 experimental results regarding place field stability over days⁵⁵ or over the course of goal-
445 oriented spatial learning⁵⁶. One could also imagine extending our framework by considering
446 other forms of competitive learning, such as self-organizing maps^{19,57}, which assume an
447 underlying functional structure which is maintained during learning. Such an extension might
448 explain the observations of functional clustering of field formation events around “seed”
449 plateau neurons⁵⁸, and other topographically related phenomenon observed in plateau
450 generation⁵⁹.

451 In recurrent networks like CA3, we demonstrated that unsupervised gBTSP can facilitate
452 attractor learning when implemented with appropriate architectural constraints. This aligns
453 with previous theoretical work showing BTSP's suitability for memory storage in discrete
454 attractor networks⁴. To preserve stability during learning, we used a low-rank
455 parameterization of the network's recurrent weights, only applying gBTSP to the “encoding”
456 portion of this parameterization. However, there are also other promising avenues for

457 considering BTSP in the context of unsupervised, recurrent learning. For example, under
458 certain conditions STDP in a recurrent network can approximate Hidden Markov Model
459 Learning, a very powerful tool for discovering underlying latent structure⁶⁰. Recent
460 experimental work which recorded hippocampal activity over the course of learning observed
461 an orthogonalization of the latent map⁶¹ – a feature they found was best described by
462 networks which learned HMMs^{60,62}, including a version of the STDP-recurrent network.
463 Although BTSP learns with a much larger temporal kernel than STDP, they have a similar
464 fundamental structure. One can imagine that a mapping of BTSP onto HMM learning may be
465 possible, though the rapid learning and large temporal kernel of BTSP present non-trivial
466 challenges to stability and convergence.

467 For learning based on explicit error functions (i.e. supervised learning), we derived analytical
468 expressions that determine when and where plateau events should occur to optimize task
469 performance. This formulation allows us to understand BTSP in the context of gradient-based
470 learning, with the plateau function effectively distributing "credit" for errors across the
471 network. Further, because we have an explicit analytical expression for our plateau function,
472 we can constrain it to be stochastic and sparse, akin to BTSP events observed *in vivo*^{5,48}. We
473 show that gBTSP can successfully learn feed-forward tasks, while retaining key features of
474 observed BTSP, such as few-shot learning.

475 While our supervised gBTSP successfully learned complex tasks in feed-forward networks,
476 maintaining the rapid learning characteristic of BTSP, deep recurrent networks proved more
477 challenging. We showed that these challenges arise from fundamental stability limitations of
478 large, rapid activity changes in deep and recurrent networks, limitations which are similar
479 mathematically to exploding or vanishing gradients in backpropagation^{49–51}. This presents an
480 apparent paradox, as BTSP has been experimentally observed in the highly recurrent CA3
481 region. Perhaps structures such as so-called orthogonal or unitary networks, which preserve
482 spectral norms of the recurrent weights (and thereby maintain stable gradient flow), can offer
483 a solution, but training in these networks is difficult to reconcile with gBTSP^{63–65}. Alternatively,
484 it might be sensible to model CA3 as dynamically regulating its recurrence, gating certain
485 pathways such that they behave as feed-forward networks during learning episodes (note
486 that we used this method earlier to train our unsupervised recurrent network; **Figure 3**).

487 Our analysis has focused on a "strong" hypothesis, by which hippocampal learning is
488 governed mainly via BTSP, and the plateau events follow some unifying principle (such as
489 minimizing a particular error or loss function). We call this the "strong" assumption, because
490 a) it remains unknown what fraction of overall learning is due to discrete, rapid BTSP events,
491 b) it is highly likely that multiple forms of plasticity, including BTSP, are active simultaneously,
492 and c) it is unclear if plateau-driven learning is guided by any sort of governing computational
493 principle. A "softer" hypothesis might posit that BTSP is but a small fraction of hippocampal
494 learning, and/or it is relegated to a trivial function such as taking mere random "snapshots"
495 of complex representations occurring in other cortical areas. While this "soft" hypothesis
496 remains worthy of consideration and further study, rates of BTSP appear to be relatively high,
497 particularly in novel environments^{5,48}, and previous theoretical work suggests that purely
498 random plateau occurrence is inconsistent with the task-specific formation of complex
499 hippocampal representations (i.e. splitter cells)⁶⁶.

500 The limitations we identify suggest that while BTSP may indeed play a crucial role in
501 hippocampal learning, at the very least, its implementation likely requires specialized circuit
502 designs and/or other forms of plasticity to maintain stability. Still, the stark contrast between
503 traditional gradient-based learning (where both population and single-unit representations
504 evolve gradually), and BTSP-like learning, (where individual units can change rapidly while

505 population representations evolve more gradually), highlights a fundamental difference
506 between learning in artificial and biological systems.

507 Ultimately, this work introduces a generalized mathematical and analytical framework for
508 BTSP (gBTSP) and uses this framework to investigate how plateau events may be distributed
509 to solve learning tasks. Our findings suggest that while placing the entire burden of credit
510 assignment on plateau events alone may be insufficient to explain complex aspects of
511 hippocampal learning, BTSP is capable of rapid memory formation and latent encoding,
512 particularly in feed-forward, and constrained recurrent networks. Future work should identify
513 the biological circuits and plasticity mechanisms that stabilize hippocampal networks
514 undergoing BTSP, particularly within CA3, to better understand how BTSP contributes to the
515 development of hippocampal cognitive functions.

516

518 **Methods**

519 All parameters for the following methods are included in **Table 1**.

520

521 **Generalized Learning Rule for Behavioral Timescale Plasticity**

522 To begin our derivation, we consider the simple case of a single postsynaptic neuron which
523 triggers an instantaneous plateau event, and a single presynaptic neuron which fires a spike
524 (**Figure 1a**). We assume the postsynaptic plateau updates weight W via some function,
525 W_{kernel} , which depends on the timing of the presynaptic spike relative to the plateau:

526
$$\Delta W = W_{kernel}(t_{pre} - t_{plateau}) \quad (3)$$

527 Specifically, we choose a W_{kernel} such that the application of our learning rule matches
528 observed plasticity following application of a single plateau and bursting inputs *in vitro* (**Figure**
529 **1d**)². The specific form of the weight kernel is:

530
$$W_{kernel}(t - t^p) = \begin{cases} e^{(t-t^p)/\tau_b}, & t < t^p \\ e^{-(t-t^p)/\tau_f}, & t \geq t^p \end{cases} \quad (4)$$

531 Where t^p is the time of the plateau, and τ_b and τ_f are “backward” and “forward” time
532 constants.

533 Next, we relax our previous assumption that there is a single presynaptic neuron which fires
534 a single spike, instead describing the continuous activity of presynaptic neuron j at time t as
535 $x_j(t)$ (**Figure 1b**). Now, the change in weights following a single plateau depends on the
536 integrated presynaptic activity across a temporal window Δt relative to the time of the
537 plateau):

538
$$\Delta W_j = \int_{t_{plateau}-\Delta t}^{t_{plateau}+\Delta t} W_{kernel}(t - t_{plateau}) x_j(t) dt \quad (5)$$

539 Note that if $x_j(t)$ is taken to be a delta function $\delta(t - t_{pre})$, Equation 5 reduces to Equation
540 3. Further, the asymmetric offset of observed plasticity in **Figure 1e** is a direct consequence
541 of the shape of W_{kernel} (Equation 4). For some intuition on why this is the case, notice that
542 Equation 5 is equivalent to a cross-correlation, so we can imagine “sliding” or “smearing”
543 W_{kernel} across the input $x_j(t)$ to get a given horizontal slice of **Figure 1e**.

544 To match experimental data showing that the amplitude of the formed field depends on the
545 initial membrane voltage of the postsynaptic cell¹, we add in a dependence on the synaptic
546 strength prior to the plateau event, leading to the equation:

547
$$\Delta W_j = \int_{t_{plateau}-\Delta t}^{t_{plateau}+\Delta t} W_{kernel}(t - t_{plateau}) x_j(t) dt - \lambda W_j \quad (6)$$

548 where the weight dependence is parametrized by λ . Note since ΔW is only applied when a
549 plateau occurs, $-\lambda W$ is not a continuous weight decay.

550 Finally, we want to consider the case for which there are multiple postsynaptic neurons, each
551 of which may have multiple plateaus. So, we introduce $P_i(t)$, a function representing the post-
552 synaptic plateau potential at time t for neuron i . Now, the change in weights depends on an
553 integral over the presynaptic activity, as well as an integral over any post-synaptic plateaus
554 (**Figure 1c**):

555
$$\Delta W_{ij} = \int_0^T P_i(t) \left[\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij} \right] dt \quad (1)$$

556 in which weight W_{ij} is updated after each trial according to the presence of $P_i(t)$. Note that if
 557 $P_i(t)$ is taken to be a delta function $\delta(t - t_{plateau})$, Equation 1 reduces to Equation 6. Since
 558 Equation 1 is derived from various degrees of generalization (Equations 3, 5, and 6), we call
 559 this equation “generalized BTSP”.

560 **Unsupervised Feed-Forward Task**

561 Notice in Equation 6, that if we take the temporal kernel $W_{kernel}(t - t_{plateau})$ to be the delta
 562 function $\delta(t - t_{plateau})$, the integral over time goes away and we get the following expression:

563
$$\Delta W_j = x_j(t_{plateau}) - \lambda W_j \quad (7)$$

564 So, in this approximation, upon each plateau event, the weights would move towards a fixed
 565 point $W_j = \frac{1}{\lambda} x_j(t_{plateau})$, akin to classical conceptions of “competitive learning”^{17–20}.

566 For the 1-D unsupervised feed-forward task in **Figure 2b–e**, we assume an animal is running
 567 along a 1-D treadmill at constant velocity β , i.e. the animal’s position $u(t) = \beta t$. The external
 568 sensory input is modeled in the form of stereotypical 1-D tuning curves with added noise:

569
$$x_j(t) = e^{-\left(\frac{(u(t)-u_j)}{2\sigma^2}\right)^2} + N(0, \sigma_N^2) \quad (8)$$

570 where j indexes over N total inputs, and u_j are the locations (or equivalently, times) of the
 571 tuning curve centers, which have standard deviation σ . Zero-mean Gaussian noise is added,
 572 with standard deviation σ_N . These inputs are connected to an output $y_i(t)$ by feed-forward
 573 weights W_{ij} :

574
$$y_i(t) = \sum_j W_{ij} x_j(t) \quad (9)$$

575 which are learned via gBTSP. We used the following rule to trigger a plateau:

576
$$\text{if } \sum_i y_i(t) < \theta: P_K(t) = 1 \quad (10)$$

577 Where θ is a firing rate threshold, and K is a random index from 1 to N . A single plateau can
 578 drive the network above the threshold θ (at a given time). Weights were initialized at zero and
 579 the network was trained on 100 laps.

580 For the 2-D unsupervised feed-forward task in **Figure 2f–h**, we assume an animal begins at
 581 a random location inside a 2-D box, (u_0, v_0) and takes T steps of a random walk along a
 582 trajectory $(u(t), v(t))$. The external sensory input is modeled in the form of stereotypical 2-D
 583 tuning curves with added noise:

584
$$x_j(t) = e^{-\left(\frac{(u(t)-u_j)}{2\sigma^2}\right)^2} e^{-\left(\frac{(v(t)-v_j)}{2\sigma^2}\right)^2} + N(0, \sigma_N^2) \quad (11)$$

585 where j indexes over N^2 total inputs, and (u_j, v_j) are the 2-D locations of the tuning curve
 586 centers, which have standard deviation σ . Zero-mean Gaussian noise is added, with standard
 587 deviation σ_N . These inputs are connected to an output $y_i(t)$ by feed-forward weights W_{ij}
 588 (Equation 9), which are learned via gBTSP, just as in the 1-D case.

589

590 **Unsupervised Recurrent Task**

591 For the unsupervised recurrent task in **Figure 3**, we assume an animal is running along a 1-
 592 D treadmill at constant velocity β , i.e. the animal's position $u(t) = \beta t$. The external sensory
 593 input is modeled the same as for the unsupervised feed-forward task (Equation 8). There are
 594 two populations of neurons, “visible” neurons $x_j(t)$ and “seed” neurons $s_i(t)$. These
 595 populations connect recurrently to each other, but not amongst themselves. Visible neurons
 596 receive external input $o_k(t)$, and only the seed neurons eligible for plateaus (**Figure 3a**). The
 597 activity of these two populations is governed by the following equations:

598
$$s_i(t) = \sum_j W_{ij}^e x_j(t-1) \quad (12)$$

599
$$x_j(t) = (1 - \alpha) \sum_i W_{ji}^d s_i(t) + \alpha \sum_k W_{jk}^{in} o_k(t) \quad (13)$$

600 where W_{ij}^e are “encoding” weights, W_{ji}^d are “decoding” weights, and W_{jk}^{in} are input weights. α
 601 is a gating variable governed by the norm of the external input: when external input is high,
 602 recurrent input is low, and vice versa:

603
$$\alpha = \frac{C - \|\mathbf{o}(t)\|}{C} \quad (14)$$

604 Where C is a constant and $\|\mathbf{o}(t)\|$ is the norm of the external input. In the limit of no external
 605 input ($\alpha = 0$), our equation for the visible neurons reduces to:

606
$$x_j(t) = \sum_{j'} \sum_i W_{ji}^d W_{ij'}^e x_{j'}(t-1) = \sum_{j'} W_{jj'}^{rec} x_{j'}(t-1) \quad (15)$$

607 where $W_{jj'}^{rec}$ is the low-rank recurrence, $\sum_i W_{ji}^d W_{ij'}^e$, of the visible neurons. However, in the limit
 608 of large external input, our network becomes effectively feed-forward:

609
$$x_j(t) = \sum_k W_{jk}^{in} o_k(t) \quad (16)$$

610 This dual nature of the network allows us to take advantage of recurrent computation in the
 611 low-input phase, while making use of unsupervised, feed-forward gBTSP in the high-input
 612 phase. Encoding weights are learned via the same competitive learning algorithm as the
 613 feed-forward case (Equation 10). The decoder weights are set to be the transpose of the
 614 encoder weights. Following a single lap of training, a test phase was conducted, whereby
 615 inputs were only shown for one timestep before being removed. The inputs were shown at
 616 times 0, 2, 4, 6, and 8 seconds.

617

618 **Supervised Feed-Forward Task**

619 In order to apply plateaus in the supervised context, we derive an expression for $P(t)$ which
 620 minimizes a given error/loss. Assuming some target output $\hat{y}(t)$, and a loss function \mathcal{L} (here
 621 we choose a mean squared error loss).

622
$$\mathcal{L} = \frac{1}{2} [\hat{y}(t) - y(t)]^2. \quad (17)$$

623 We can compare our gBTSP weight update

624
$$\Delta W_{ij} = \int_0^T P_i(t) \left[\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij} \right] dt \quad (1)$$

625

626 to that of simple backpropagation (for a single layer, this is just the delta rule):

627
$$\Delta W_{ij} = \int_0^T \varepsilon_i(t) x_j(t) dt \quad (18)$$

628

629 Where the error term $\varepsilon_i(t) = \hat{y}_i(t) - y_i(t)$. By setting these two equations to be equal, we
630 can find an expression for the function $P_i(t)$:

631
$$P_i(t) = \varepsilon_i(t) \sum_{j=1}^N \frac{x_j(t)}{\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij}} \quad (2)$$

632

633 For the first task, we consider a shallow feed-forward network (Equation 9). We choose a
634 target function $\hat{y}(t)$ that is a putative place field, modeled as a Gaussian bump centered at a
635 specific location in the environment:

636
$$\hat{y}(t) = e^{-\left(\frac{u(t)-u_0}{2\sigma^2}\right)^2} \quad (19)$$

637 Where $u(t)$ is the animal's position, and u_0 is the location of the tuning curve center, which
638 has standard deviation σ .

639 For the navigation task, the network has three layers (input, hidden, output), for which only
640 the hidden neurons can receive plateaus, i.e. only the input to hidden weights W_{jk} are
641 trainable:

642
$$y_i(t) = \sum_j V_{ij} h_j(t)$$

$$h_j(t) = \sum_k W_{jk} x_k(t) \quad (20)$$

643 The 2D target trajectory for the task is:

644
$$\hat{y}_1(t) = \cos\left(\frac{2\pi t}{T}\right) + \frac{1}{2} \sin\left(\frac{4\pi t}{T}\right) + \frac{1}{4} \cos\left(\frac{8\pi t}{T}\right)$$

$$\hat{y}_2(t) = \sin\left(\frac{2\pi t}{T}\right) + \frac{1}{2} \sin\left(\frac{4\pi t}{T}\right) + \frac{1}{4} \sin\left(\frac{8\pi t}{T}\right) \quad (21)$$

645 We train the network (Equations 1 and 2) for 100 trials on both tasks.

646

647 Supervised Recurrent Task

648 For our recurrent task, a network of hidden units with activation $h_j(t)$ is recurrently connected
649 via weights W_{ij}^{rec} . The dynamics of the hidden units are governed by the following:

650
$$\tau \frac{dh_i(t)}{dt} = -h_i(t-1) + \varphi \left(\sum_j W_{ij}^{rec} h_j(t-1) + \sum_k W_{ik}^{in} x_k(t) \right) \quad (22)$$

651 here φ is a non-linear function (i.e. tanh) of the recurrent inputs, and these activations are
652 initialized at h_0 . Input $x_k(t)$ is projected to the network via input weights W_{ik}^{in} . We initialize the
653 internal weights as a random Gaussian matrix with a gain factor g . These hidden units project
654 to output $y(t)$ via weights W_i^{out} :

655
$$y(t) = \sum_j W_j^{out} h_j(t) \quad (23)$$

656 To find $P_i(t)$ which minimizes the error in a recurrent network, we again compare our gBTSP
657 rule

658
$$\Delta W_{ij} = \int_0^T P_i(t) \left[\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij} \right] dt \quad (1)$$

659 to the full backpropagation through time update: (for a full derivation, see Murray, 2019¹³):

660
$$\Delta W_{ij} = \int_0^T z_i(t) \varphi' \left(\sum_j W_{ij}^{rec} h_j(t-1) + \sum_k W_{ik}^{in} x_k(t) \right) dt \quad (24)$$

661 where φ' is the derivative of our activation function, and the Lagrange multiplier $z_i(t)$ is equal
662 to:

663
$$\begin{aligned} z_i(t) &= \left(1 - \frac{1}{\tau}\right) z_i(t+1) \\ &+ \frac{1}{\tau} \sum_{j=1}^N z_i(t+1) \varphi'(u_i(t+1)) W_{ji}^{rec} \\ &+ \sum_{l=1}^{N_l} W_{il} \varepsilon_l(t) \end{aligned} \quad (25)$$

664 where $u_i(t)$ is our total input current to the unit, i.e.,

665
$$u_i(t) = \sum_j W_{ij}^{rec} h_j(t-1) + \sum_k W_{ik}^{in} x_k(t) \quad (26)$$

666 The Lagrange multiplier $z_i(t)$ is calculated in the “backwards” phase, by starting with the
667 terminal value, $z_i(T)$, and working back to $z_i(0)$. $z_i(T)$ takes the form:

668
$$z_i(T) = \sum_{l=1}^{N_l} W_{il} \varepsilon_l(T) \quad (27)$$

669 By setting Equation 1 and Equation 24 to be equal, we solve for the plateau function and
670 get the following expression:

671
$$P_i(t) = \frac{z_i(t)}{\tau} \varphi'(u_i(t)) \sum_j \frac{h_j(t-1)}{\int_{t-\Delta t}^{t+\Delta t} W_{kernel}(t' - t) x_j(t') dt' - \lambda W_{ij}} \quad (28)$$

672 The delayed non-match to sample task consists of two inputs, representing odor A and odor
673 B, and two outputs: one representing licking probability and the other which represents a

674 ramping temporal component. This component encourages the network to develop
 675 sequential internal representations⁴⁷. For each trial, a random odor combination
 676 (AA,AB,BA,BB) was selected. The first odor input was presented for the first 1s of the trial,
 677 and the second odor input was present between 7-8s. No odor inputs were given in the
 678 delay period. For non-matching pairs, the target for licking probability was 1 for all timesteps
 679 after 8 seconds, and 0 otherwise. For matching pairs, the target licking probability was
 680 always 0. Output weights were trained using the delta rule, while recurrent weights were
 681 trained using our gBTSP update (Equation 1), after selecting plateaus (Equation 28). The
 682 network was trained for 50,000 trials, with the gBTSP update (Equation 1) passed through a
 683 momentum-based optimizer (ADAM⁴⁶) to avoid critical instabilities (see Table 1 for
 684 parameters).

685

686 Constraints on Few-Shot Learning

687 In the case of the shallow network, inputs $x_j(t)$ project to output $y_i(t) = \sum_j W_{ij}x_j(t)$. (**Figure**
 688 **6b**). “Plateaus” of size Δx occur at the inputs, and the loss is the mean squared error between
 689 output y and target \hat{y} . Solving for the Hessian (local curvature), we find:

$$690 \begin{aligned} \mathcal{L} &= \frac{1}{2}(\hat{y} - Wx)^T(\hat{y} - Wx) \\ \frac{\partial \mathcal{L}}{\partial x} &= -W^T(\hat{y} - Wx) \\ \frac{\partial^2 \mathcal{L}}{\partial x \partial x^T} &= \frac{\partial}{\partial x^T} \left(\frac{\partial \mathcal{L}}{\partial x} \right) = W^T W \end{aligned} \quad (29)$$

691

692 If we consider a deep feed-forward network with layers l and layer specific weights W_l
 693 (**Figure 6c**), the expression for the Hessian becomes more complicated:

$$694 \begin{aligned} \mathcal{L} &= \frac{1}{2}(\hat{y} - \prod_{l=1}^L W_l x)^T(\hat{y} - \prod_{l=1}^L W_l x) \\ \frac{\partial \mathcal{L}}{\partial x} &= -\left(\prod_{l=1}^L W_l\right)^T(\hat{y} - \prod_{l=1}^L W_l x) \\ \frac{\partial^2 \mathcal{L}}{\partial x \partial x^T} &= \frac{\partial}{\partial x^T} \left(\frac{\partial \mathcal{L}}{\partial x} \right) = \left(\prod_{l=1}^L W_l\right)^T \prod_{l=1}^L W_l \end{aligned} \quad (30)$$

695

696 For recurrent networks (**Figure 6d**), the Hessian in a recurrent network depends on the T th
 697 product of W , which also leads to exploding and vanishing contributions from an update Δx :

698

$$\begin{aligned}\mathcal{L} &= \frac{1}{2} (\hat{y} - \prod_{t=1}^{t_0} Wx)^T (\hat{y} - \prod_{t=1}^{t_0} Wx) \\ \frac{\partial \mathcal{L}}{\partial x} &= -(\prod_{t=1}^{t_0} W)^T (\hat{y} - \prod_{t=1}^{t_0} Wx) \\ \frac{\partial^2 \mathcal{L}}{\partial x \partial x^T} &= \frac{\partial}{\partial x^T} \left(\frac{\partial \mathcal{L}}{\partial x} \right) = (\prod_{t=1}^{t_0} W)^T \prod_{t=1}^{t_0} W\end{aligned}\quad (31)$$

699 These conditions set fundamental limits on both the architectures and the tasks for which
700 deep or recurrent artificial networks can support rapid changes in single-unit activity.

701

702 **References**

- 703 1. Milstein, A. D. *et al.* Bidirectional synaptic plasticity rapidly modifies hippocampal representations. *eLife*
704 **10**, e73046 (2021).
- 705 2. Bittner, K. C., Milstein, A. D., Grienberger, C., Romani, S. & Magee, J. C. Behavioral time scale synaptic
706 plasticity underlies CA1 place fields. *Science* **357**, 1033–1036 (2017).
- 707 3. Bittner, K. C. *et al.* Conjunctive input processing drives feature selectivity in hippocampal CA1 neurons.
708 *Nat. Neurosci.* **18**, 1133–1142 (2015).
- 709 4. Li, Y., Briguglio, J. J., Romani, S. & Magee, J. C. Mechanisms of memory-supporting neuronal dynamics in
710 hippocampal area CA3. *Cell* **0**, (2024).
- 711 5. Grienberger, C. & Magee, J. C. Entorhinal cortex directs learning-related changes in CA1 representations.
712 *Nature* **611**, 554–562 (2022).
- 713 6. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
- 714 7. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560
715 (1990).
- 716 8. Laje, R. & Buonomano, D. V. Robust timing and motor patterns by taming chaos in recurrent neural
717 networks. *Nat. Neurosci.* **16**, 925–933 (2013).
- 718 9. DePasquale, B., Cueva, C. J., Rajan, K., Escola, G. S. & Abbott, L. F. full-FORCE: A target-based method for
719 training recurrent networks. *PLOS ONE* **13**, e0191527 (2018).
- 720 10. Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support
721 error backpropagation for deep learning. *Nat. Commun.* **7**, 13276 (2016).
- 722 11. Lillicrap, T. P. & Santoro, A. Backpropagation through time and the brain. *Curr. Opin. Neurobiol.* **55**, 82–
723 89 (2019).
- 724 12. Williams, R. J. & Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural
725 Networks. *Neural Comput.* **1**, 270–280 (1989).
- 726 13. Murray, J. M. Local online learning in recurrent networks with random feedback. *eLife* **8**, e43299 (2019).
- 727 14. Cone, I. & Shouval, H. Z. Behavioral Time Scale Plasticity of Place Fields: Mathematical Analysis. *Front.*
728 *Comput. Neurosci.* **15**, (2021).

729 15. Li, P. Y. & Roxin, A. Rapid memory encoding in a recurrent network model with behavioral time scale
730 synaptic plasticity. *PLOS Comput. Biol.* **19**, e1011139 (2023).

731 16. Wu, Y. & Maass, W. A simple model for Behavioral Time Scale Synaptic Plasticity (BTSP) provides content
732 addressable memory with binary synapses and one-shot learning. *Nat. Commun.* **16**, 342 (2025).

733 17. Fukushima, K. Cognitron: A self-organizing multilayered neural network. *Biol. Cybern.* **20**, 121–136
734 (1975).

735 18. Grossberg, S. Adaptive pattern classification and universal recoding: I. Parallel development and coding
736 of neural feature detectors. *Biol. Cybern.* **23**, 121–134 (1976).

737 19. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43**, 59–69
738 (1982).

739 20. Rumelhart, D. E. & Zipser, D. Feature Discovery by Competitive Learning. *Cogn. Sci.* **9**, 75–112 (1985).

740 21. Wallis, G. & Rolls, E. T. INVARIANT FACE AND OBJECT RECOGNITION IN THE VISUAL SYSTEM. *Prog.*
741 *Neurobiol.* **51**, 167–194 (1997).

742 22. Fritzke, B. Growing cell structures—A self-organizing network for unsupervised and supervised learning.
743 *Neural Netw.* **7**, 1441–1460 (1994).

744 23. Ziv, Y. *et al.* Long-term dynamics of CA1 hippocampal place codes. *Nat. Neurosci.* **16**, 264–266 (2013).

745 24. Geva, N., Deitch, D., Rubin, A. & Ziv, Y. Time and experience differentially affect distinct aspects of
746 hippocampal representational drift. *Neuron* **111**, 2357-2366.e5 (2023).

747 25. Rubin, A. *et al.* Revealing neural correlates of behavior without behavioral measurements. *Nat.*
748 *Commun.* **10**, 4745 (2019).

749 26. Khona, M. & Fiete, I. R. Attractor and integrator networks in the brain. *Nat. Rev. Neurosci.* **23**, 744–766
750 (2022).

751 27. Samsonovich, A. & McNaughton, B. L. Path Integration and Cognitive Mapping in a Continuous Attractor
752 Neural Network Model. *J. Neurosci.* **17**, 5900–5920 (1997).

753 28. Rolls, E. T. An attractor network in the hippocampus: Theory and neurophysiology. *Learn. Mem.* **14**,
754 714–731 (2007).

755 29. Wills, T. J., Lever, C., Cacucci, F., Burgess, N. & O'Keefe, J. Attractor Dynamics in the Hippocampal
756 Representation of the Local Environment. *Science* (2005) doi:10.1126/science.1108905.

757 30. Tsodyks, M. & Sejnowski, T. J. Associative memory and hippocampal place cells. *Int. J. Neural Syst.* **6**, 81–
758 86 (1995).

759 31. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities.
760 *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982).

761 32. Abu-Mostafa, Y. & St. Jacques, J. Information capacity of the Hopfield model. *IEEE Trans. Inf. Theory* **31**,
762 461–464 (1985).

763 33. Agmon, H. & Burak, Y. A theory of joint attractor dynamics in the hippocampus and the entorhinal cortex
764 accounts for artificial remapping and grid cell field-to-field variability. *eLife* **9**, e56894 (2020).

765 34. Burak, Y. & Fiete, I. R. Accurate Path Integration in Continuous Attractor Network Models of Grid Cells.
766 *PLOS Comput. Biol.* **5**, e1000291 (2009).

767 35. Ságodi, Á., Martín-Sánchez, G., Sokół, P. & Park, I. M. Back to the Continuous Attractor. Preprint at
768 <https://doi.org/10.48550/arXiv.2408.00109> (2024).

769 36. Neunuebel, J. P. & Knierim, J. J. CA3 Retrieves Coherent Representations from Degraded Input: Direct
770 Evidence for CA3 Pattern Completion and Dentate Gyrus Pattern Separation. *Neuron* **81**, 416–427
771 (2014).

772 37. Guzman, S. J., Schlögl, A., Frotscher, M. & Jonas, P. Synaptic mechanisms of pattern completion in the
773 hippocampal CA3 network. *Science* **353**, 1117–1123 (2016).

774 38. Seung, H. S. Learning Continuous Attractors in Recurrent Networks. in *Advances in Neural Information
775 Processing Systems* vol. 10 (MIT Press, 1997).

776 39. Darshan, R. & Rivkind, A. Learning to represent continuous variables in heterogeneous neural networks.
777 *Cell Rep.* **39**, 110612 (2022).

778 40. Sussillo, D. & Abbott, L. F. Generating Coherent Patterns of Activity from Chaotic Neural Networks.
779 *Neuron* **63**, 544–557 (2009).

780 41. Chandra, S., Sharma, S., Chaudhuri, R. & Fiete, I. Episodic and associative memory from spatial scaffolds
781 in the hippocampus. *Nature* **638**, 739–751 (2025).

782 42. Magó, Á., Kis, N., Lükő, B. & Makara, J. K. Distinct dendritic Ca²⁺ spike forms produce opposing input-
783 output transformations in rat CA3 pyramidal cells. *eLife* **10**, e74493 (2021).

784 43. Raus Balind, S. *et al.* Diverse synaptic and dendritic mechanisms of complex spike burst generation in
785 hippocampal CA3 pyramidal cells. *Nat. Commun.* **10**, 1859 (2019).

786 44. Tapson, J. & van Schaik, A. Learning the pseudoinverse solution to network weights. *Neural Netw.* **45**,
787 94–100 (2013).

788 45. Taxidis, J. *et al.* Differential Emergence and Stability of Sensory and Temporal Representations in
789 Context-Specific Hippocampal Sequences. *Neuron* **108**, 984-998.e9 (2020).

790 46. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. Preprint at
791 <https://doi.org/10.48550/arXiv.1412.6980> (2017).

792 47. Zhou, S., Seay, M., Taxidis, J., Golshani, P. & Buonomano, D. V. Multiplexing working memory and time in
793 the trajectories of neural networks. *Nat. Hum. Behav.* **7**, 1170–1184 (2023).

794 48. Priestley, J. B., Bowler, J. C., Rolotti, S. V., Fusi, S. & Losonczy, A. Signatures of rapid plasticity in
795 hippocampal CA1 representations during novel experiences. *Neuron* **110**, 1978-1992.e6 (2022).

796 49. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. in
797 *Proceedings of the 30th International Conference on Machine Learning* 1310–1318 (PMLR, 2013).

798 50. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult.
799 *IEEE Trans. Neural Netw.* **5**, 157–166 (1994).

800 51. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).

801 52. Whittington, J. C. R., McCaffary, D., Bakermans, J. J. W. & Behrens, T. E. J. How to build a cognitive map:
802 insights from models of the hippocampal formation. Preprint at <http://arxiv.org/abs/2202.01682> (2022).

803 53. Stachenfeld, K. L., Botvinick, M. M. & Gershman, S. J. The hippocampus as a predictive map. *Nat.*
804 *Neurosci.* **20**, 1643–1653 (2017).

805 54. Behrens, T. E. J. *et al.* What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron* **100**,
806 490–509 (2018).

807 55. Vaidya, S. P., Li, G., Chitwood, R. A., Li, Y. & Magee, J. C. Formation of an expanding memory
808 representation in the hippocampus. *Nat. Neurosci.* 1–9 (2025) doi:10.1038/s41593-025-01986-3.

809 56. Qian, F. K., Li, Y. & Magee, J. C. Mechanisms of experience-dependent place-cell referencing in
810 hippocampal area CA1. *Nat. Neurosci.* 1–11 (2025) doi:10.1038/s41593-025-01930-5.

811 57. Kohonen, T. Essentials of the self-organizing map. *Neural Netw.* **37**, 52–65 (2013).

812 58. Geiller, T. *et al.* Local circuit amplification of spatial selectivity in the hippocampus. *Nature* 1–5 (2021)
813 doi:10.1038/s41586-021-04169-9.

814 59. McKenzie, S. *et al.* Preexisting hippocampal network dynamics constrain optogenetically induced place
815 fields. *Neuron* **109**, 1040–1054.e7 (2021).

816 60. Kappel, D., Nessler, B. & Maass, W. STDP Installs in Winner-Take-All Circuits an Online Approximation to
817 Hidden Markov Model Learning. *PLOS Comput. Biol.* **10**, e1003511 (2014).

818 61. Sun, W. *et al.* Learning produces an orthogonalized state machine in the hippocampus. *Nature* 1–11
819 (2025) doi:10.1038/s41586-024-08548-w.

820 62. George, D. *et al.* Clone-structured graph representations enable flexible learning and vicarious
821 evaluation of cognitive maps. *Nat. Commun.* **12**, 2392 (2021).

822 63. Arjovsky, M., Shah, A. & Bengio, Y. Unitary Evolution Recurrent Neural Networks. Preprint at
823 <https://doi.org/10.48550/arXiv.1511.06464> (2016).

824 64. Le, Q. V., Jaitly, N. & Hinton, G. E. A Simple Way to Initialize Recurrent Networks of Rectified Linear
825 Units. Preprint at <https://doi.org/10.48550/arXiv.1504.00941> (2015).

826 65. Henaff, M., Szlam, A. & LeCun, Y. Recurrent Orthogonal Networks and Long-Memory Tasks. in
827 *Proceedings of The 33rd International Conference on Machine Learning* 2034–2042 (PMLR, 2016).

828 66. Cone, I. & Clopath, C. Latent representations in hippocampal network model co-evolve with behavioral
829 exploration of task structure. *Nat. Commun.* **15**, 687 (2024).

830

831

832 Acknowledgements

833 This work was supported by BBSRC (BB/N013956/1), Wellcome Trust (200790/Z/16/Z), the
834 Simons Foundation (564408), EPSRC (EP/R035806/1 and EP/X029336/1) and ERC-UKRI
835 (EP/Y027841/1).

836 Author Contributions

837 I.C., C.C. and R.P.C. conceived and designed the model. I.C. developed and performed the

838 simulations. I.C., C.C. and R.P.C. wrote the manuscript.

839 **Competing Interests**

840 The authors declare no competing interests.

841

Tables

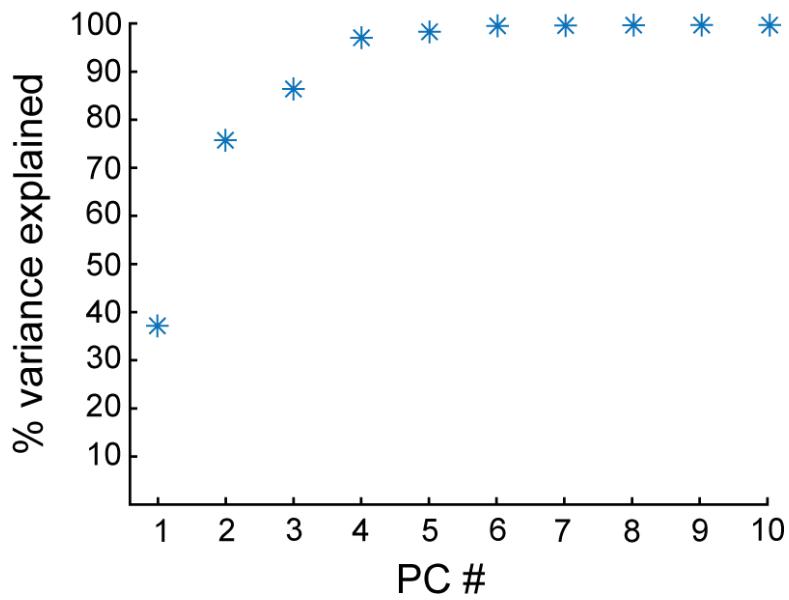
Parameter	Value	Units	Description
T	200	dt	Total timesteps
dt	50	ms	Timestep
τ_b	1.31	s	“Backwards” kernel time constant
τ_f	0.69	s	“Forwards” kernel time constant
λ	1	-	Weight constant
Δt	5	s	Time window of integration around plateau
γ	10	-	Constant input, to go from ΔW to ΔV (Figure 1)
σ	0.075	s	Standard deviation of input Gaussians
θ	8	-	Threshold for plateau event (Figure 2)
N_{input}	200	-	Number of input neurons
N_{output}	81	-	Number of output neurons
η	0.95	-	Learning rate (Figure 2)
σ_n	$\mathcal{N}(0, 1/35)$	-	Noise, output neuron activity
L	100	cm	Box side length
σ_w	$\mathcal{N}(0, 1)$	cm	Random walk updates (2D)
ϑ	0.15	-	Threshold for plotting neural activity (Figure 2i)
N_{seed}	81	-	Number of seed neurons
$N_{visible}$	100	-	Number of visible neurons
θ	.6	-	Threshold for plateau event (Figure 3)
η	0.3	-	Learning rate (Figure 3)
W^{in}	\mathbb{I}	-	Input weights (Figure 3)
C	2.71	-	Constant for gating term
N_{input}	100	-	Number of input neurons (Figure 4)
N_{inter}	10	-	Number of hidden neurons (Figure 4, fixation task only)
N_{output}	1, 2	-	Number of output units (Figure 4, place cell task, fixation task)
T	100	dt	Total timesteps (Figure 5)
τ_{net}	10	dt	Network time constant (Figure 5)
N_{input}	2	-	Number of input neurons (Figure 5)
N_{rec}	100	-	Number of recurrent neurons (Figure 5)
N_{output}	2	-	Number of output neurons (Figure 5)
g	.85	-	Gain, recurrent weight initialization (Figure 5)
η	0.01	-	Learning rate, ADAM (Figure 5)
β_1	0.9	-	Decay rate for first moment estimate in output weights, ADAM (Figure 5)
β_2	0.99	-	Decay rate for second moment estimate in output weights, ADAM (Figure 5)
$\beta_{1,rec}$	0.999	-	Decay rate for first moment estimate in recurrent weights, ADAM (Figure 5)
$\beta_{2,rec}$	0.9999	-	Decay rate for second moment estimate in recurrent weights, ADAM (Figure 5)
ϵ	10^{-8}	-	Constant to avoid division by zero, ADAM (Figure 5)

Table 1: Model Parameters

842

843

844 **Supplemental Figures**

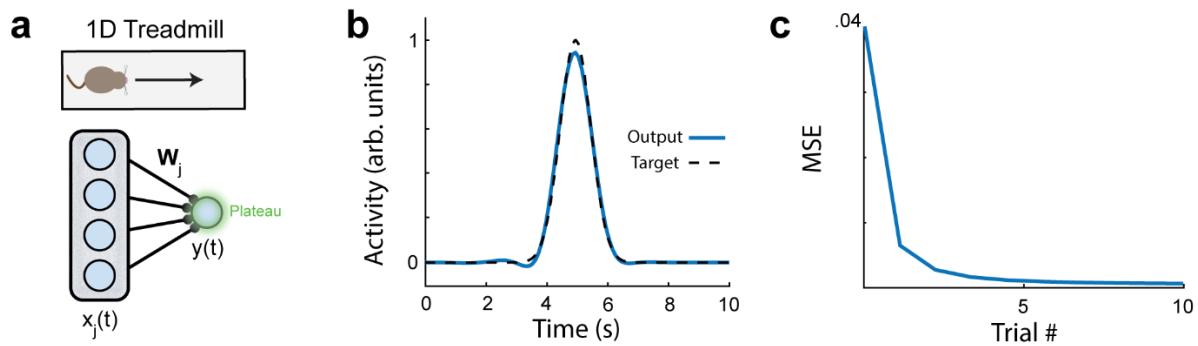


Supplemental Figure 1 – Variance explained by principal components in recurrent attractor network

Cumulative sum of variance explained by the first ten principal components of the effective recurrent weight matrix, $\mathbf{W}^{\text{rec}} = \mathbf{W}^{\text{e}}\mathbf{W}^{\text{d}}$, after competitive learning via gBTSP. The first two components are shown in Figure 3c.

845

846



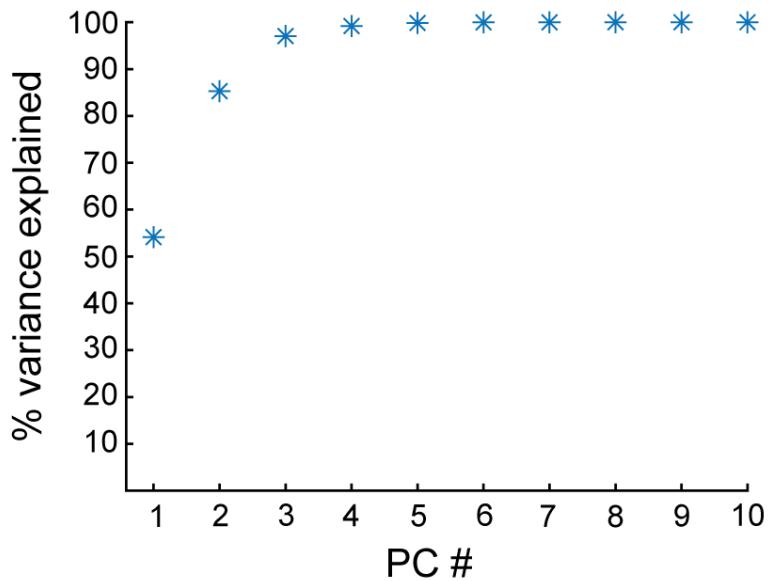
Supplemental Figure 2 – Mean squared error for supervised feed-forward tasks

a) Feed-forward network with inputs $x_j(t)$ project to output $y(t)$ via weights W_j . The inputs $x_j(t)$ are spatially selective and represent an animal running along a 1D track. **b)** A unimodal gaussian target function (dotted black line) and the trained output (blue line) after the first 10 trials of gBTSP training. **c)** The mean squared error over the first 10 trials of training.

847

848

849

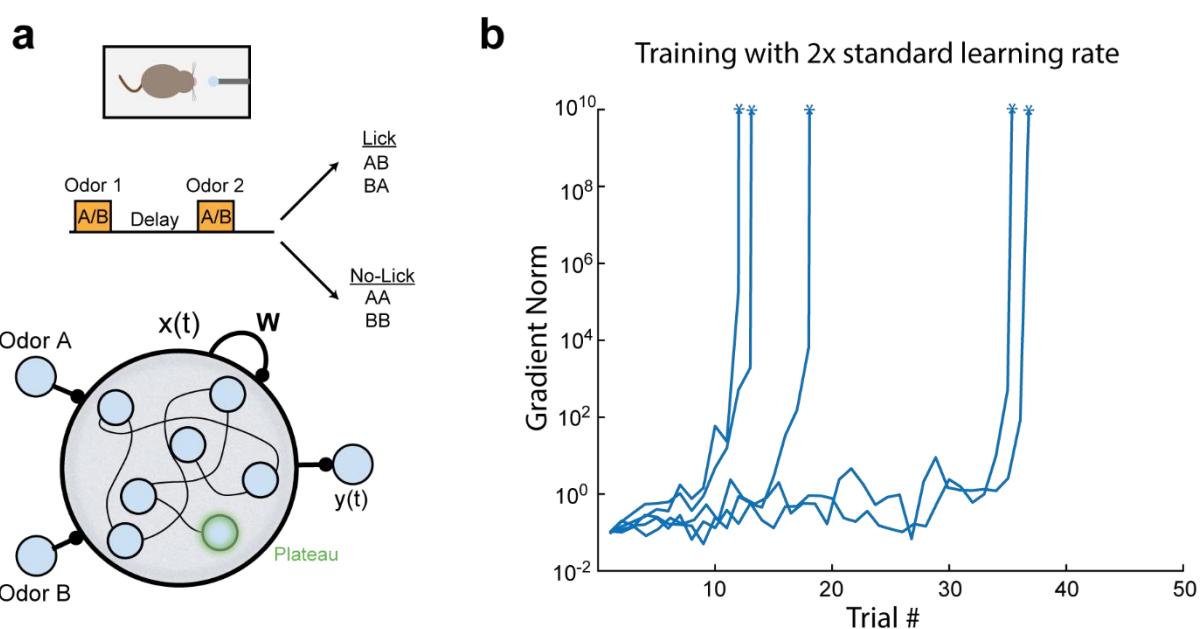


Supplemental Figure 3 – Variance explained by principal components in DNMS task

Cumulative sum of variance explained by the first ten principal components of network activity in the DNMS task. The first three components are shown in Figure 5e.

850

851



Supplemental Figure 4 – Large learning rates lead to instability in gBTSP in recurrent networks

a) Simulated agents are trained on a delayed-non-match-to-sample (DNMS) task where they must distinguish between sequential “odor” pairs. The agent must learn to lick for non-matching sequences (AB, BA), and refrain from licking for matching sequences (AA, BB). Bottom, the model consists of a recurrent network with two odor inputs, hidden activities $x(t)$ and recurrent weights W . The hidden units project to an output $y(t)$ via weights V . Recurrent weights W are trained via plateaus occurring in the hidden units according to our gBTSP algorithm (see Methods). Output weights are trained via the delta rule. **b)** Training using a learning rate double that used in Figure 6 (see also Table 1) results in gradient explosion. First five random seeded runs shown. Y-axis is the norm of the gradient of the recurrent weights and is truncated at a value of 10^{10} for visual clarity.