Replay as a basis for backpropagation through time in the brain

Huzi Cheng, Ehren Newman, Joshua W. Brown

December 17, 2021

Abstract

How recurrent neural networks learn sequences has been a puzzle for the neuroscience community for a long time, as Back-propagation through time (BPTT) has been considered biologically implausible despite its widespread use in models. Most of the existing solutions focus on online approximations of BPTT. Here we show, inspired by the reverse replay phenomenon in the hippocampus, that an offline approximation of BPTT is possible, which computes the full error gradient through time, without violating locality constraints, and outperforms existing online alternatives. This new Reversible Recurrent Neural Network (R2N2) model consists of two separate RNNs (**consolidator** and **cache**) and forms a system that learns in both one-shot and statistical manners. Based on its success in various tasks and match to existing neurophysiological data, we also propose this system as a model of the entorhinal-hippocampal system in sequence learning.

1 Introduction

Whether making a decision to turn left or right in a T-maze task or playing piano, computations that rely on information from the past are pervasive in the brain. In learning those tasks, one of the main challenges for the recurrent neural networks (RNNs) in the brain is to properly compute necessary weight changes for all synapses. In artificial neural networks, this problem is typically solved by propagating the error signal in a

backward manner on the computation graph, both spatially and temporally (Figure 1 **A**). This is biologically implausible for two reasons: 1) a spatial back-propagation requires a symmetric relationship between the bottom-up and top-down connection (meaning the blue and red projections between the output layer and the recurrent layer must be the transpose of each other in Figure 1 **A**), which is thought to be unrealistic, and 2) in temporal back-propagation, the storage of the precise activation history of all neurons is crucial, without which the computation of the full gradient is thought to be impossible (Lillicrap et al., 2020) (In the propagation with blue projections in Figure 1 **A**, the neural activity of each unit in the recurrent layer needs to be stored and retrieved from $t = n$ to $t = 1$). In the machine learning world this is implemented with high RAM as the neural activity could be stored and recalled in RAM arbitrarily, but it's unclear how some brain regions could store and replay the firing rate history of all neurons in other regions.

There are solutions for both questions. For the first problem, it has been shown that even fixed random top-down connections can facilitate learning with performance similar to back-propagation(Lillicrap et al., 2016) (for the red projection in in Figure 1 **A**, we no longer set it to the transpose of blue projections but instead replace it with a random fixed matrix), and a proper learning mechanism within top-down connections can gradually adjust them to be symmetric with the bottom-up connections (Akrout et al., 2019). For the second problem, various online algorithms have been proposed (approximations of error signals generated by blue projections in Figure 1 **A** using only forward computations). Though the underlying mechanisms are diverse, many of them try to approximate the full gradient in the BPTT in a local and online way (Depasquale et al., 2018; Murray, 2019; Tallec and Ollivier, 2017). Some even dropped learning entirely in recurrent layers and were still successful in various tasks, such as the echo state network (Jaeger, 2002). One common feature of these algorithms is that they avoid temporal back-propagation of errors. However, we here question the validity of the second problem and propose an alternative approach to sequential computations in the brain: the reversible recurrent neural network (R2N2), which consists of a consolidator and cache system (Figure 1 **B**). This system functions in a similar way of BPTT: the system first interacts with the environment and gets a feedback (error) signal sequence from it. Then the cache network stores sequences in

a quick (one-shot) way and uses them to train the consolidator statistically in a slower manner, similar to the error propagation in Figure 1 **A** with red and blue projections.

During learning, as with BPTT, the whole system operates by running backward in time. Unlike BPTT in machine learning though, it does not require any external storage of unit activations at each time step. *The first key insight here is that instead of storing the activity of each unit at all times, the networks simply play the sequence back in reverse order, thus reconstructing the activity as needed instead of storing it.* Both the consolidator and cache network are reversible by design and thus can retrieve activity at previous time points without violating biological locality constraints. With this, the full BPTT delta rule for learning can be constructed arbitrarily far back in time (Eq. 4). The consolidator consists of two reciprocally connected populations of neurons and learns to reconstruct its history of activations by adjusting the balance between different branches of projections, which can switch the network between forward and reverse play mode.

The cache network is a variant of a Hopfield network but also reversible, and it learns a new sequence first in one shot so that it can play itself backwards to train the consolidator network. This leads to the *second key insight, which is that the error signal of the delta rule is reconstructed backward in time, in synchrony with the backward reconstruction of network activity (Fig. 2A). The error signals of the previous time steps are computed by repeatedly matrix-multiplying the error signal by two fixed, random (feedback alignment) matrices, then adding in the error derived from the difference between the desired output (as specified by the cache network) and the actual output (as specified by the consolidator network).*

In the following simulations, we show that together this system can be trained to perform various tasks and outperform some online alternatives. The reverse replay in both the consolidator and cache network is crucial for this type of learning. This matches previous observations that the reverse replay in the brain is key for sequence learning (Diba and Buzsáki, 2007; Hemberger et al., 2019; Schuck and Niv, 2019; Vaz et al., 2020; Eichenlaub et al., 2020; Fernández-Ruiz et al., 2019; Michon et al., 2019).

It implies that the hippocampal cortical system may be a neural instantiation of BPTT, and our proposed model might account for the underlying mechanism of reverse replay as well as its computational role in learning. Our simulation results also reveal

that during learning the system shows similar characteristics and internal representations to place cells and replay rate effects observed in previous studies (Ziv et al., 2013; Driscoll et al., 2017; Shin et al., 2019).

2 Results

In the following, we first describe the architecture and learning rules for the **consolidator** and **cache** networks in our system. We trained each network independently on several tasks to show their roles and capabilities in sequence learning. Lastly, the system as a whole is trained to perform a commonly used navigation task, and its properties are compared with previous animal experiment results.

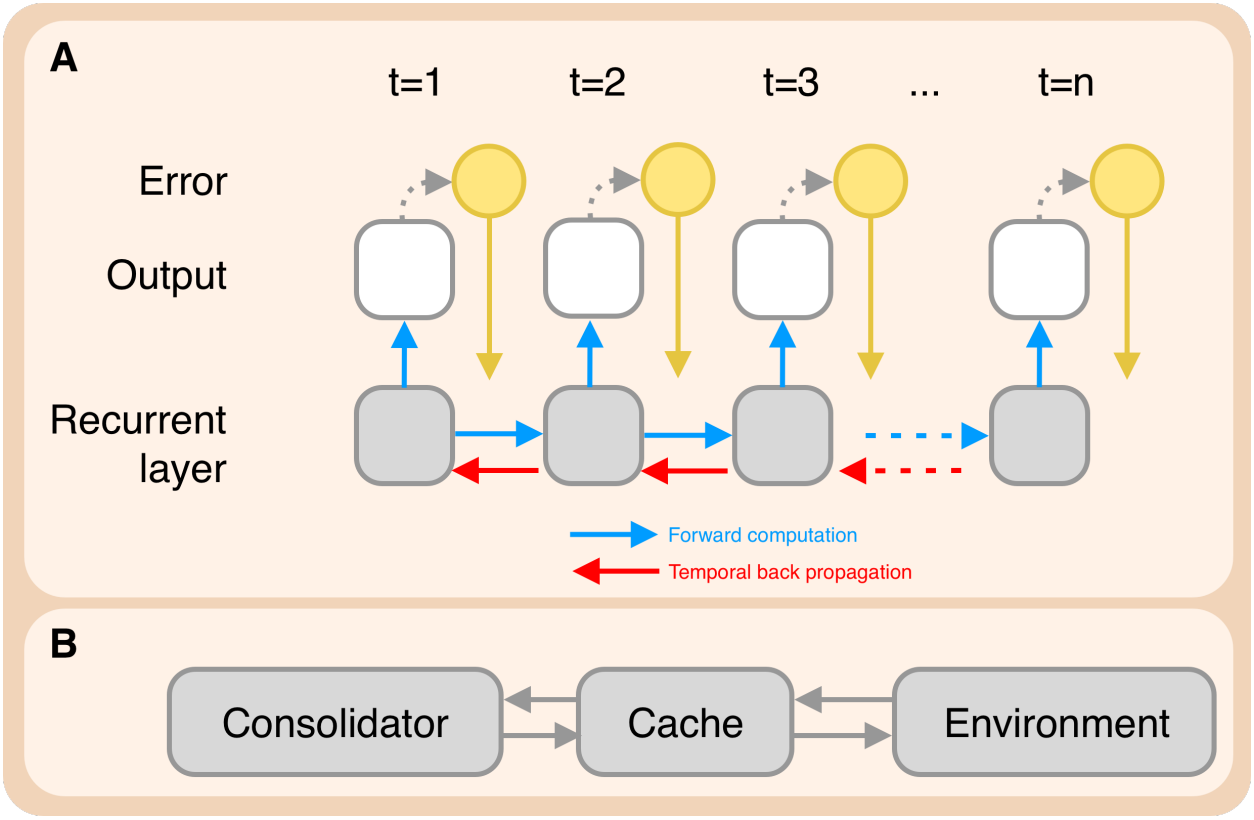

Figure 1: **A** Temporally unfolded computational process of an RNN in BPTT: The recurrent layer at each time step generates an output through the blue projection; error signals (yellow circles) are computed according to the output layer and propagated to the recurrent layer via the yellow projection. To compute the gradient at different time steps, error signals also need to be propagated temporally, i.e. backward in time. **B** Consolidator-Cache model. The Consolidator-Cache system first generates output and gets feedback from the environment. Then the cache network stores sequences and plays them back in reverse order to the consolidator network, which in turn optimizes itself based on the replayed content.

2.1 Models

2.1.1 Consolidator network

Some of the existing model of reverse replay (Haga and Fukai, 2018; Parish et al., 2020) assume neurons are organized in the chained way or fire in order with some spatially shifting signals. Though providing an explanation of how reverse replay works, they are not flexible enough as the expressive power of such chaining neurons are either predetermined by the connectivity structure before learning or the representation is one-hot (only one single neuron or spatially clustered neurons being active at a moment). However, it's well-known that the brain represents information in a highly distributed way and the connectivity structure changes as the learning goes on. To address these problems, we need to design an architecture that is capable of reversing its activity without adding constraints on its spatial connectivity pattern. Taking inspiration from Chang et al. (2017), we studied an RNN that is composed of at least two interconnected populations of neurons A and B (see Figure 2**A**). Neuron group A receives input signals from neuron group B and itself and vice versa. More specifically, we divide the incoming projections for each group into two parts, and the corresponding dynamics equations can be written as follow:

$$\begin{aligned} \tau \frac{dh_A}{dt} &= -h_A + f_A\left(h_A, h_B\right) + g_A\left(h_A, h_B\right) \\ \tau \frac{dh_B}{dt} &= -h_B + f_B\left(h_A, h_B\right) + g_B\left(h_A, h_B\right) \\ y &= k(h_B) \end{aligned} \tag{1}$$

We define the f_* projections as forward connections while the g_* being the backward projections and k being the output projection. In this part we will be focused on the recurrent units in A and B and leave the discussion of output y to later parts, in which the network firing rate dynamics generated by f_* is $H_f = \{(h_A^0, h_B^0), \cdots, (h_A^T, h_B^T)\}$, which is called the forward sequence and represents the normal running or forward replay phase of this RNN. To generate a reversed sequence of H_f, mathematically, one needs to simply flip the sign of derivatives in the dynamics equation from $t = 0$ to $t = T$. If we discretize the dynamics equations into a difference form, it will degenerate to the case of a reversible deep neural network block (Chang et al., 2017), which has been shown to be memory-constant in various tasks as the storage requirement of neural

activity equals the number of units in the network.

The intuition is depicted in Figure 2 **B**. The blue circuit represents a transformation $\oplus$ ($\oplus$ represents the operation that combines A and B together) from A^t, B^t to A^{t+1}, B^{t+1} in the next time step: $A^t \oplus B^t \to A^{t+1}$ and $B^t \oplus A^{t+1} \to B^{t+1}$. Then, as long as there exist another inverse operation $\ominus$ that satisfies $X \oplus Y = Z \Leftrightarrow Z \ominus Y = X$, we can construct a circuit that turns A^{t+1}, B^{t+1} into A^t, B^t: $B^{t+1} \ominus A^{t+1} \to B^t$ and $A^{t+1} \ominus B^t \to A^t$.

In effect, these two operators ($\oplus$ and $\ominus$) reverse the update process without requiring any other constraints: the two variables A and B could be either scalar (signal neuron case) or high dimensional vector (neuron group case). One of the simplest pairs of operators that meet the above conditions consists of addition and subtraction, which is the operator set used by Chang et al. (2017). To map these two operators to the brain seems implausible as it requires flipping the excitability of a synapse in a short time range. However, we can approximately reach the same effect by introducing competition from another group of projections g_*. The goal of the g_* projections is to generate currents that have the same amplitude but with different signs of f_* to cancel $f_* - h_*$, which can be easily implemented with local dendritic propagation and local training (see Eq.2) on g_*'s parameter θ_{g_*}. This detailed balance between f_* and g_* projections thus makes it possible to run the whole system in a backward manner: If the excitability of a trained g_* projection is scaled by a factor of 2, the resulting effective projection will be approximately $h_* - f_*$.

$$\Delta\theta_{g_*} \propto \frac{\partial\big(f_* - h_* - g_*\big)}{\partial\theta_{g_*}} \tag{2}$$

There are many possible choices for f_* and g_* projections. For example, each can be a simple two-layer neural network if we consider the branching structure of dendrites, which has been shown experimentally to be equivalent with multi-layer nonlinear neural networks (Poirazi et al., 2003). In this case, a complex $f_* - h_*$ architecture can be approximated by g_* as long as the complexity of f_* is not higher than g_*. For demonstration, in the following experiments, we choose the simplest form of f_* to be a single layer nonlinear network and g_* to be a combination of one-layer linear and nonlinear networks. Mathematically, this reduces the learning process of backward projection g_* to be a regression process, which is known to be trivial to solve with local learning

rules but is still enough to support complex sequential computation as non-linearity is involved in each time step. Besides, this architecture (Figure 2 A) is scalable and thus is possible to support more complicated computations. If one adds another Group along with A and B, the coupling can be extended, and the backward running can still be preserved.

Figure 2A depicts a network that can produce sequences of activity in reverse, and the same network structure can also be used to propagate error signals in reverse. Figure 2 B shows two network graphs, one which runs activity sequences in reverse, and one which runs error sequences in reverse.

In BPTT (Eq.3), the error signal e_H^t used to update hidden layer connections θ_{f_*} are recursively computed in a mirroring circuit (yellow projections in Fig.2 **A** and **B**) of the forward circuit from "future" to "past", i.e., e_H^t relies on both the future hidden layer error e_H^{t+1} and the transient output error L^t.

$$\begin{aligned} \Delta\theta_{f_*} &\propto \sum_{t=T}^{0} e_H^t X^t \\ e_H^t &= \frac{\partial L^t}{\partial H^t} \\ X^t &= \frac{\partial H^t}{\partial \theta_{f_*}} \end{aligned} \tag{3}$$

Two types of implausibility exist in this process: 1). the weight transport problem, i.e., how to compute the transient component of e_H^t with L^t (Whittington and Bogacz, 2019) and 2). the external storage of activations, i.e., how to compute X_t in Eq.3. On the one hand, for the first issue, Lillicrap et al. (2016) proposed an alternative local error circuit, Feedback Alignment (FA). With fixed random projections, it has been shown to be effective on various deep network architectures and tasks (Nøkland, 2016; Moskovitz et al., 2018). On the other hand, our synaptic competition balance mechanism discribed above addresses the second issue, as it reconstructs the previous network states X_t in a backward manner.

Together these two mechanisms propagate the error temporally back with a simple linear tuning of projection excitability and without any non-local information (the first line in Eq.3). The FA algorithm approximates hidden layer errors e_H^t in Eq.3 with $\widehat{e_H^t}$ at each time step using L^t and a fixed random feedback matrix. (for the specific derivation of $\widehat{e_H^t}$ and its recursive update equation, see the supplementary material) The backward

projections (red connections in Figure. 2 A) in the consolidator network replace X_t with $\widehat{X^t}$ by approximating H^t with $\widehat{H^t}$. i.e., the reconstructed neural activities generated in a reverse replay. The product of $\widehat{e_H^t}$ and $\widehat{X^t}$ is then used to update forward projections θ_{f_*}.

$$\begin{aligned} \Delta\theta_{f_*} &\propto \sum_{t=T}^{0} \widehat{e_H^t}\widehat{X_t} \\ \widehat{X_t} &= \frac{\partial \widehat{H^t}}{\partial \theta_{f_*}} \end{aligned} \tag{4}$$

2.1.2 Cache network

The learning mechanism described so far (Eq.3) is effective when the **consolidator** network is solely determined by its previous states, i.e., without external inputs, as the reverse replay equation will not hold if we add a time-varying terms in Eq.1. This limits the use cases of the **consolidator** as most of the sequence learning tasks involve dealing with temporal inputs. To perform reverse replay in a running **consolidator** that integrates time-varying input sequences I (see Eq 5) through the mapping b, an external storage of sensory sequence inputs I becomes necessary, so that Eq 1 is modified as:

$$\begin{aligned} \tau\frac{dh_A}{dt} &= -h_A + f_A\left(h_A, h_B\right) + g_A\left(h_A, h_B\right) + b(I) \\ \tau\frac{dh_B}{dt} &= -h_B + f_B\left(h_A, h_B\right) + g_B\left(h_A, h_B\right) \end{aligned} \tag{5}$$

In the replay phase, the **consolidator** itself cannot generate the dynamics without knowing b and, by extension, I. Superficially, this brings us back to the original dilemma, i.e., to design another RNN that can run backward. The difference is that it should have the capability to memorize a given sequence after as little as a single exposure, which makes the problem harder. Nevertheless, the fact is that sensory input sequences in most cases are usually in a space that has much fewer dimensions compared with the number of neurons in **consolidator**, and this suggests a solution.

To memorize sequential sensory inputs and play them in a reversed manner, one can simply build point attractors representing inputs in the state space and connect them with directed line attractors (Figure 2**D**). A modified Hopfield RNN (Figure 2**C**) is a perfect match for these desired characteristics. A classical Hopfield network is

capable of building energy basins that lead noisy inputs to corresponding attractors, while a further modification to its learning rule, from $\Delta W \propto I \cdot I^{\intercal}$ to Eq.6 then links one point with another ($I^t \longrightarrow I^{t+1}$) in the state space with a direction pointing to I^i if this trial is rewarded ($r = 1$). This is the general Hopfield weight update equation for the cache network:

$$\Delta W \propto r \cdot \sum_{t,t+1} p^t \cdot (I^{t\intercal} + I^{t+1\intercal}) \tag{6}$$

By linking multiple (I^t, I^{t+1}) pairs from $t = 0$ to $t = T$ with the modified learning rule, a reversed pattern sequence $\{I^T, I^{T-1}, \cdots, I^0\}$ is built.

In the following experiments, we build an RNN (the **cache**) using the above learning rule combined with time-varying weights (Lee, 2002) to increase the stability of transitions between successive sensory patterns (Eq.7). Once I^t is stably transformed to I^{t-1} through W_E , another group of weights W_O will dominate the transition from I^{t-1} to I^{t-2} through the tuning of λ (see in Figure 2**C** the yellow and green projections tuned by two competing interneurons) , which can be viewed as an external periodic control signal or a signal indicating the stability of **cache** activations (Sompolinsky and Kanter, 1986). These signals can be realized though gating interneurons acting on the synapses of pyramidal neurons in CA3. In terms of a physical analogy, one can imagine this as a reciprocating pump, in which W_E drives the system from I^t to I^{t-1}, and then W_O drives the system from I^{t-1} to I^{t-2}, and so on back and forth between W_E and W_O, by the following equation that governs the activity of the cache network:

$$\tau \frac{dI}{dt} = -I + \lambda W_E \phi(I) + (1 - \lambda) W_O \phi(I) \tag{7}$$

The **cache** network thus can learn arbitrary sequences of patterns in a local, stable, and one-shot manner as the weights update rule of the Hopfield network is local and can be computed with only a single exposure to the inputs.

2.1.3 Sequence learning with consolidator and cache together: A fast and statistical learning system

In training a vanilla RNN with BPTT, one needs to perform the following steps:

- Initialize the RNN and run forward with temporally varying inputs.

- Store the inputs sequence, hidden unit activations, output sequence generated and the target sequence to an external memory device.
- After the whole input sequence has been received, compute the error between output and target for the last time step.
- Extract input, output and target pairs from the memory device in a temporally reversed order, propagate the error in a backward manner and compute weight changes simultaneously.
- Apply the accumulated weight changes after finished the backward running phase.

The first point to note here is that the external storage is where the main biological implausibility lies. It's unclear how the brain could store the activity in each cell somewhere else and replay it precisely. However, with **consolidator** and **cache**, this activity memory can be reconstructed dynamically. Notably, the storage size requirement is substantially reduced, as the **consolidator** can reproduce its historical activations as a reverse-play sequence with the help of the **cache**. Consider a case in which **consolidator** has 128 neurons and the channel size of inputs is 16. The standard BPTT needs to store a sequence of 16+128 = 144-dimensional vectors as all input and hidden state vectors need to be preserved in the temporal unfolding process. However, in our model, the system only requires **cache** to store a sequence of 16-d vectors representing only the sequence of input vectors, because the **consolidator** can reconstruct its activity by itself. This means a memory of sensory experience instead of all neural activities is enough to support sequence learning. This approach also matches the empirical findings that the replay of location sequences improve animals' performance in given spatial navigation tasks (Ambrose et al., 2016).

Secondly, we modify the standard learning process in BPTT to fit our model. In BPTT, the input and target channels usually belong to different categories. Taking the classical random dots perceptual decision making task as an example, the input is usually set to the coherence of randomly moving dots' directions and the desired output target is the eye motion direction (Lo and Wang, 2006). This makes the backward running phase more complicated as the system needs to store the desired target and input patterns together and only compute the error signal based on the the difference between generated outputs and desired targets. Instead, the process can be simpli-

fied if there is no categorical difference between desired outputs and inputs. Taking inspiration from predictive coding in sequence learning (Zhang et al., 2019), we view performing cognitive tasks as a process of online sequence prediction: the task-relevant stimuli, action signal, and reward signal are treated equally and are concatenated into an integrated "sensory inputs" vector. Regardless of their structure, various cognitive tasks then can be reduced to the same type of sequence prediction task. Thus the task reduces to predicting the future state at time $t+1$ on the basis of task-relevant variables at time t.

Based on these assumptions and modifications, we propose that learning a specific task can be divided into two phases, with the first one mapped to fast learning and the second one to slower statistical learning, essentially as a consolidation process. At the first stage, the animal explores the task settings and environment randomly, generating both rewarded and unrewarded "sensory sequences" involving all task relevant variables. During this initial phase, the **cache** memorizes "sensory sequences" that are rewarded at the end of each trial, which can be learned in a one-shot fashion as it is a Hopfield network in principle (see Eq.6).

In the second phase, the **cache** starts reverse replay, sending signals to the **consolidator** and thus trains it. This means a target for the **cache** at time t is actually inputs for both the **consolidator** and **cache** at time $t+1$, so that the **cache** does not need to store a target sequence separately. Then the **consolidator** in the second phase optimizes its forward projections f_* according to the targets provided by **cache** and its own reconstructed reversed activations. Once its forward projections are changed, the backward projections g_* will be adjusted accordingly to cancel f_*. Notice that the adjustment of f_* and g_* (Eq.4 and Eq.2) could occur simultaneously as the learning of backward projection is an online process. Consequently, the knowledge about the rewarded sensory experience is transferred from **cache** to **consolidator** via fast learning at first and then statistical learning later. Besides, as the **consolidator** can go back to states it experienced, it can also perform forward replay using projections f_*, which could be used to explore possible future outcome when the model is in an intermediate state (Van Der Meer and Redish, 2010; Pfeiffer and Foster, 2013). To sum up, we view this process as an implementation of Buzsaki's two-phase model (Lörincz and Buzsáki, 2000) for training long-term memories as the interplay between **consolidator**

and **cache** in two phases simulates the entorhinal-hippocampal communication.

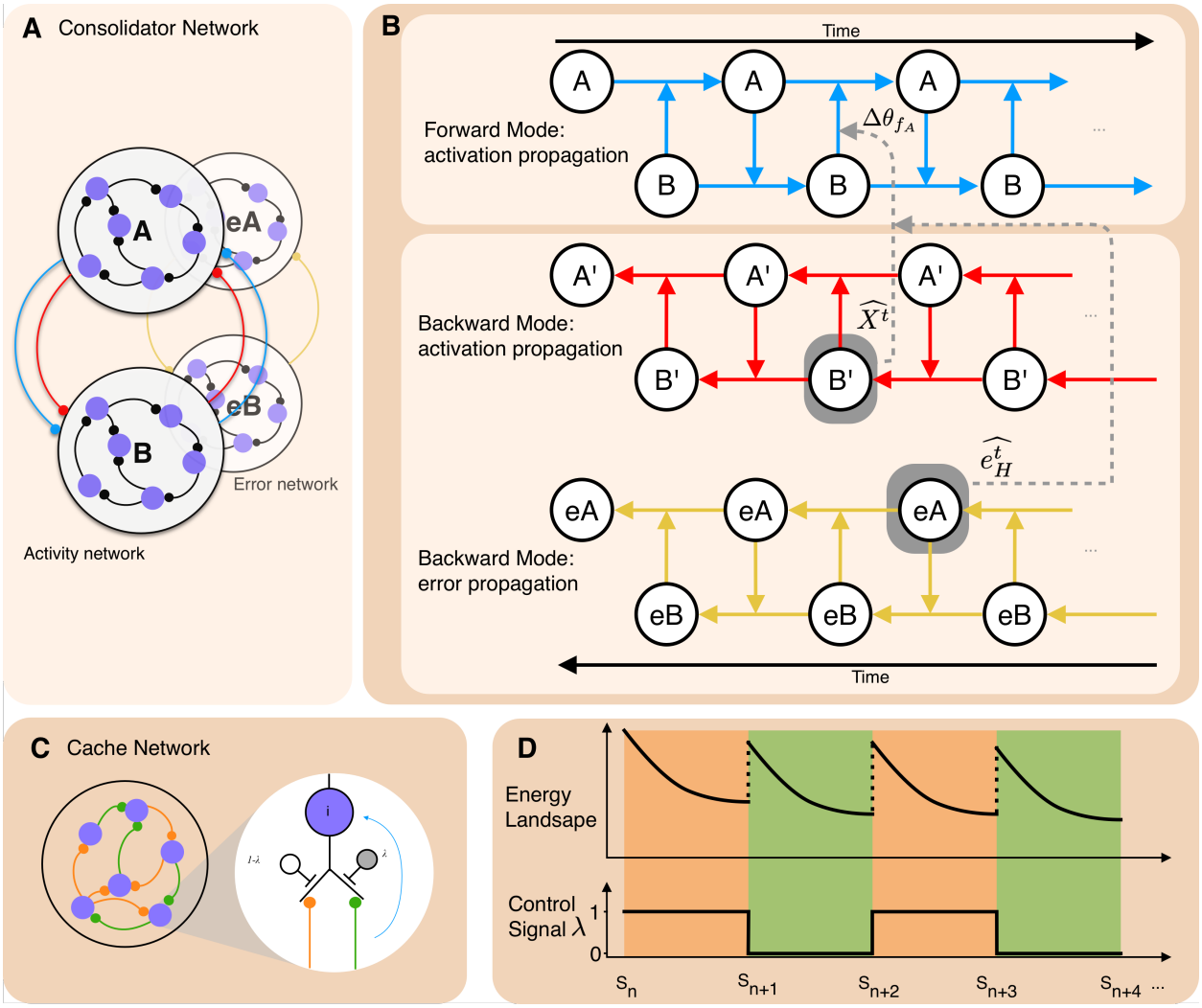

Figure 2: Schematic structure of **consolidator** and **cache**. **A** Projections of neuronal group A and B in the **consolidator** network. The consolidator network is composed of an activity network and an error network. The activity network Red ones represent f_* while blue ones represent g_*. Notice that the direction of sequence play (forward or reverse) is not fixed by the arrow but determined by the competition between f_* and m, g_*. **B** The temporal unfolding of forward and backward computation in **consolidator**. When projections f_* is stronger, the whole network operates in the forward mode. In this case, the network updates its activations in both group A and B, generates outputs and compute error signals accordingly. When the g_* connection is stronger, the whole network turns into backward mode. The neuronal group A and B generate reversed activation sequence and thus can be utilized to propagate error signals for the previous time steps, bypassing the temporal credit assignment issue caused by non-locality.

Figure 2: **C** Projections in the **cache**. In the **cache**, each neuron receives projections from both W_E (orange) and W_O (green) (**B** left). A detailed description of connection pattern in a single neuron i can be found in **B** right.The final incoming weight of neuron i is determined by $\lambda \in (0, 1)$, which can be viewed as a result of competing oscillating interneurons tuning inputs from W_E and W_O synaptic inputs. **D** A schematic description of state transitions in **cache**. By periodically switching between W_O and W_E via the control signal λ (lower panel), the network operates in two set of weights. Each of them builds state attractors between successive states S_n and S_{n+1}(energy landscape slopes between two edges of the same phase). Different successive state pairs are connected in a chaining way and thus forming a long states sequence (upper panel).

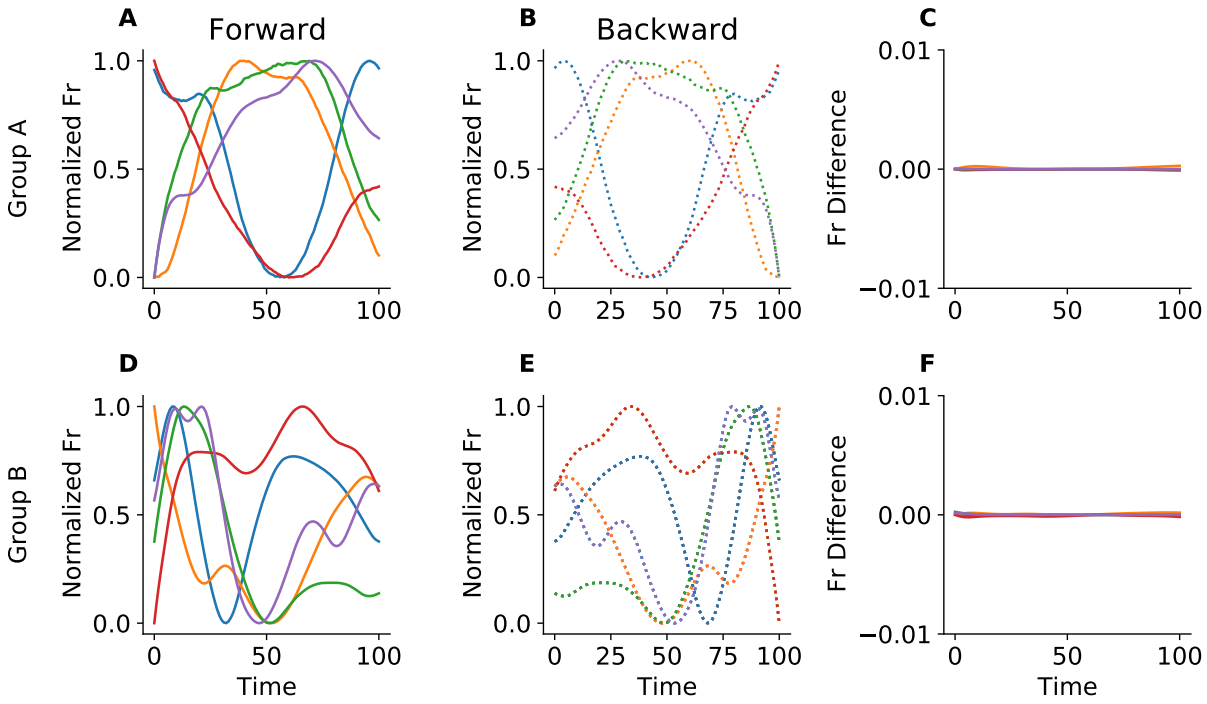

Figure 3: Normalized firing rates of neuronal group A and B in **consolidator** (For each group, first 5 neurons are selected, while overall **consolidator** has 128 neurons). **A**,**D**: Solid lines represent firing rates in forward running **B**,**E**: Dashed lines represent firing rates during backward running. **C**,**F**: Difference of firing rates between forward and backward running (flipped). Signs of symmetry is shown clearly in comparisons between **A** and **B** pair and **D** and **E** pair.

2.2 Performance

We first tested if backward projections in the **consolidator** trained through the local learning rule is capable of reconstructing the neural activation sequence in a flipped order without external signals. With setting f_* to be a one-layer nonlinear transformation with tanh being the corresponding input-output function, the g_* is defined as the sum of autapses and another one-layer nonlinear transformation. After 4 epochs of training, the backward projection is capable of reconstructing the reversed activation for about 100 time steps for both Group A and B (Figure 3).

Next, we examined the **consolidator**'s capability in input-output mapping(Figure 4). A random binary vector data stream is generated to serve as inputs to the **consol-**

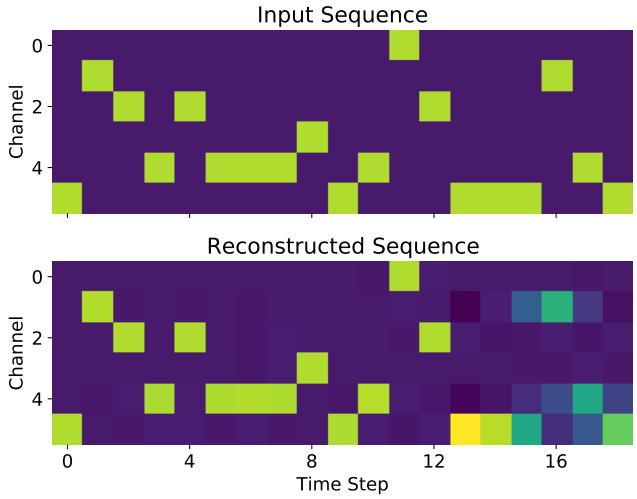

Figure 4: Sequence memorization task for the **consolidator**. The **consolidator** is trained to recall elements in the next step using data from the current time step. The top row represents the input data stream and the bottom row represents the sequence generated by **consolidator**.

idator. At each time step, the **consolidator** is trained to generate inputs in the next time step using data at the current time step as a cue. This experiment shows that the **consolidator** can learn to map input patterns to outputs at a given time despite using shared weights across multiple time steps. This implies that the temporal credit assignment problem is solved, and that Eq. 4 provides a good approximation of the full error gradient, as otherwise weights changes computed at different time steps may interfere with each other.

To further test the memorization capacity, we compared the **consolidator** with BPTT and two online RNN learning algorithms, Random Feedback Local Learning (RFLO) (Murray, 2019) and Echo State Networks (ESN), on a character prediction task $a^n b^n$ (Figure 5 **A**). In each trial of $a^n b^n$, a random number n of characters a are fed into the network first. Then it follows a line break character and another repeated n number of characters b and the second line break character. The network is trained to predict the next character given the input until that time step. To successfully perform the task, a network must both memorize character positions and count the number of a and b presented, which turns out to be harder as the number n increases.

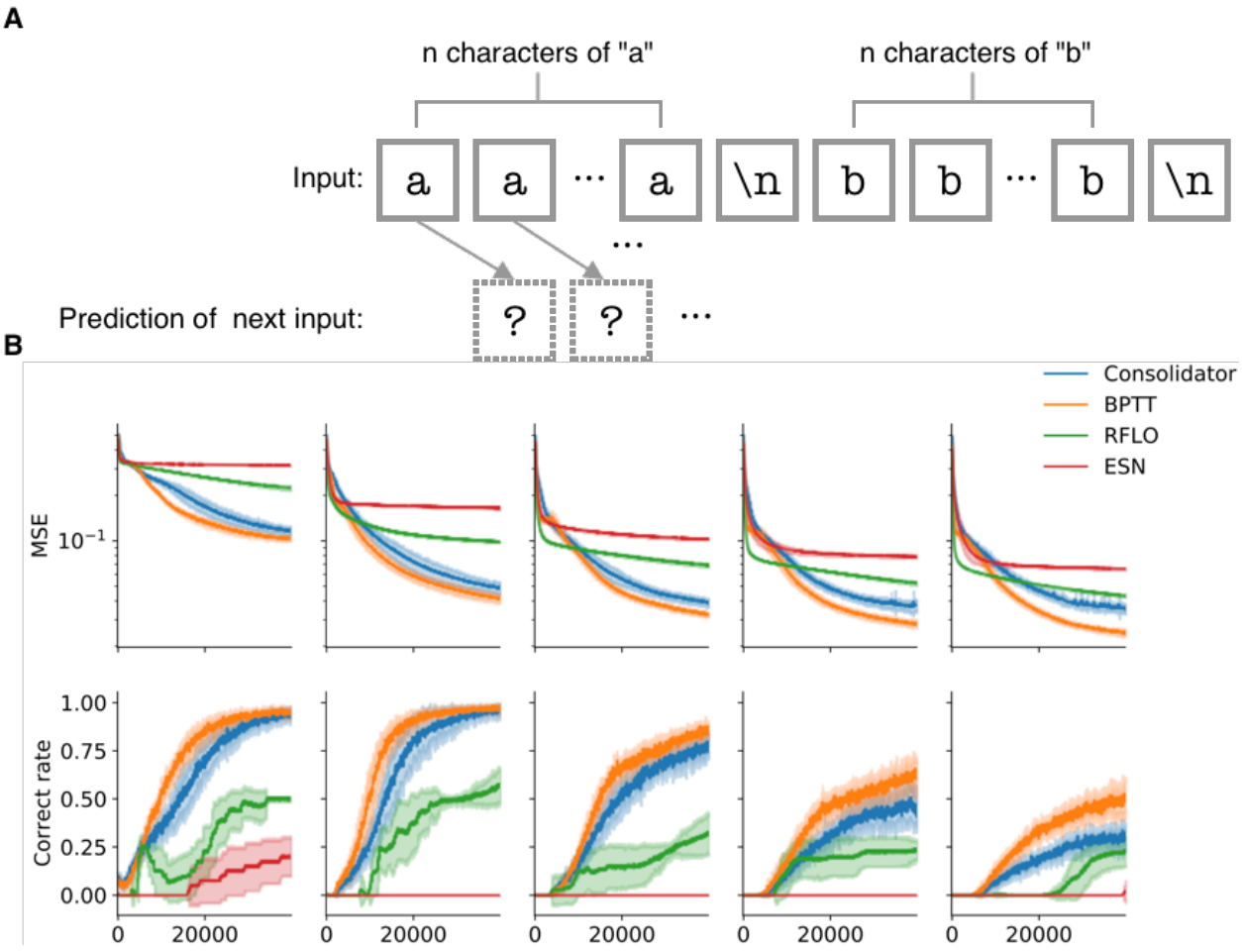

Figure 5: **A** A schematic view of $a^n b^n$ task. Each model is trained to predict the next character based on the previous inputs. **B** Performance Comparison on the $a^n b^n$ task. The upper rows show averaged loss curves while the lower rows show prediction accuracies. From the leftmost to the rightmost column, the range of n in each dataset is linearly increased from a bin including $(1, 4)$ to a bin including $(21, 24)$, thus the corresponding sequence length ranges from $(4, 10)$ to $(44, 50)$. Orange lines represent the **consolidator** with transpose of forward matrices as backward matrices. Blue lines represent **consolidator** with fixed random backward matrices, implementing feedback alignment. Green lines represent RFLO, and red lines represent ESN.

To setup a baseline, we first tested the **consolidator** but replaced the random error feedback matrices in it with the transpose of the corresponding forward matrices, a scenario functionally equivalent with BPTT (orange lines in Figure 5 **B**). This case can also be viewed as a variant of weight mirrors proposed by (Akrout et al., 2019), which substitutes the transpose of forward matrix with an evolving matrix gradually converging to the transpose with no non-local information. Then we also trained the standard **consolidator** (blue lines in Figure 5 **B**), RFLO (green lines) and ESN (red lines) on this task. In principle, BPTT captures the complete exact temporal gradient, but RFLO only computes the partial gradient of one step backward in time, and the ESN has no weight updates in its recurrent layer at all (Jaeger, 2002). Thus, we hypothesized that BPTT may have the best performance and RFLO should be better than ESN. The **consolidator** can also compute the complete temporal error gradient, but the propagated error signal is approximated via the feedback alignment matrix. Thus, we expected the **consolidator** performance to be close to BPTT and better than RFLO (Performance: BPTT $\geq$ Consolidator $>$ RFLO $>$ ESN). The results match our prediction and are shown in Figure 5 **B**. The BPTT and **consolidator** systematically outperform other algorithms. Trained RFLO networks can reach a correct rate with an upper bound at around 0.5, which reflects that the error gradient in RFLO is essentially limited to one step backward in time (Marschall et al., 2020), while the **consolidator** can propagate the error gradient backward multiple steps in time. By comparison, the ESN can hardly generate any proper outputs when the sequence length is longer than 10 (right four columns in Figure 5 **B**). This shows the advantage of multiple time step temporal error signal propagation over online error minimization, even with the constraint of no external storage of neural activation history. As the sequence length increases, all models show some performance drop (bottom row of Figure 5 **B**) and the divergence between **consolidator** with Backpropagation (orange lines) and Feedback Alignment (blue lines) also gradually increases. This shows the limits of feedback alignment temporal propagation across multiple time steps.

After finishing these basic tests for the **consolidator**, we turned to the **cache** to test whether it is capable of memorizing input sequences in a one-shot manner. For this test, the **cache** receives a randomly generated binary vector sequence once only and adjusts its recurrent connections via Eq. 6 with two types of control signals. In the case

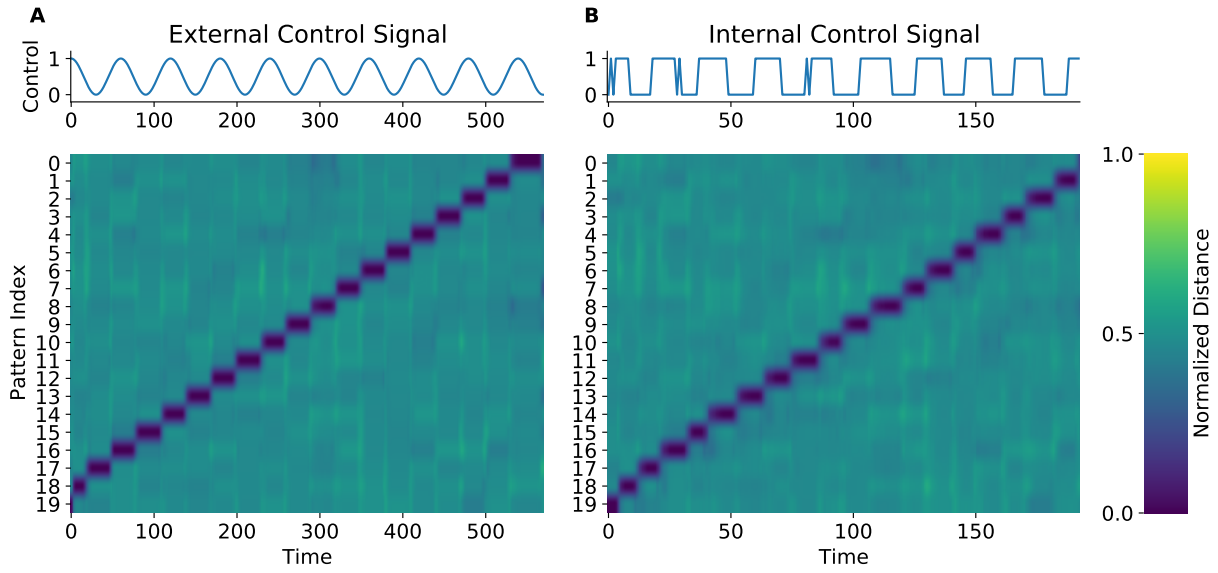

Figure 6: One shot sequence memorization task for **cache**. **A** An **cache** trained to recall the random binary sequence with an external periodic signal. Top: the oscillating external control signal λ, Bottom: The relative hamming distance between **cache**'s activation and all patterns. A larger pattern index represents a pattern that appears later in the given sequence. **B** Same as **A**, except that it uses an internally generated control signal.

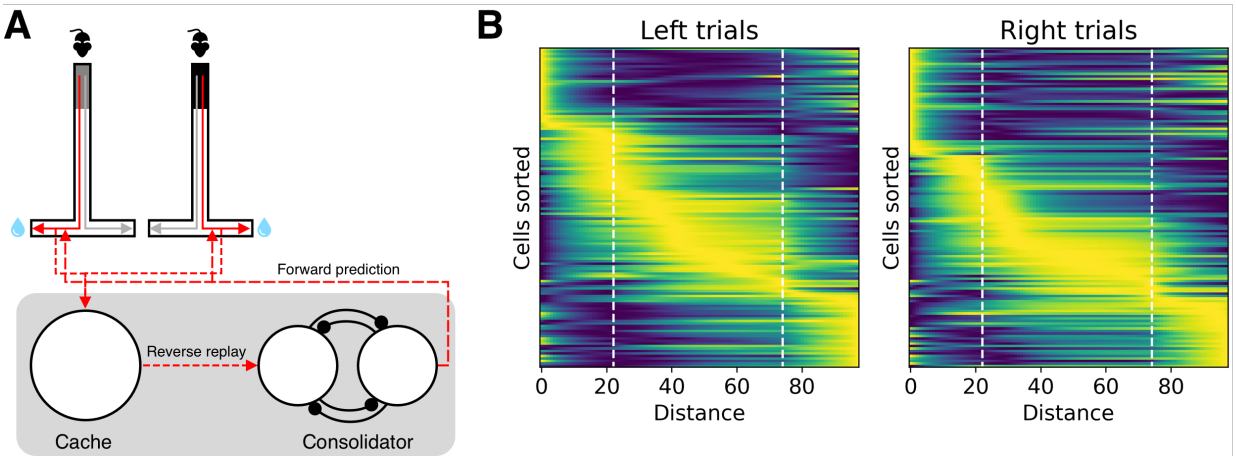

Figure 7: T-maze task trained with **consolidator** and **cache**. **A** Task structure and training paradigm. Upper: Animal makes a decision to run and then turn to left or right according to the cue type (black or grey block in the top of the T-maze) to get reward. Lower: The system first selectively receives rewarded sensory sequences (red trajectories in the T-maze) and stores it in the **cache**, which then performs reverse replay, providing reversed inputs sequence to the **consolidator**. The **consolidator** then is trained to generate rewarding predictions. **B** Place representations in the hidden unit firing rates of the **consolidator**. For left and right rewarded trials, neurons are sorted according to the distance between the start point and positions with highest firing rates. The first vertical dashed line represents the distance at which the cue ends, while the second one indicates where the left/right decision point lies.

of the external oscillating control signal (see Figure.6 A top panel), the pace of **cache** to recall previous patterns is controlled by its period, while in the internal control signal case (see Figure.6 B top panel), a discrete signal generated by the stability of the **cache** is used, and the pace thus is controlled by the **cache** itself. In the latter case, the **cache** changes its transitions between different successive pattern pairs once it reaches a stable state, similar to the temporal average integral of neural activities in (Sompolinsky and Kanter, 1986). In both cases, the **cache** successfully retrieved the binary input sequence in a reversed order.

Lastly, we test the learning capability of **consolidator** and **cache** as a learning system {**consolidator**, **cache**}. We trained this system to perform a T-maze naviga-

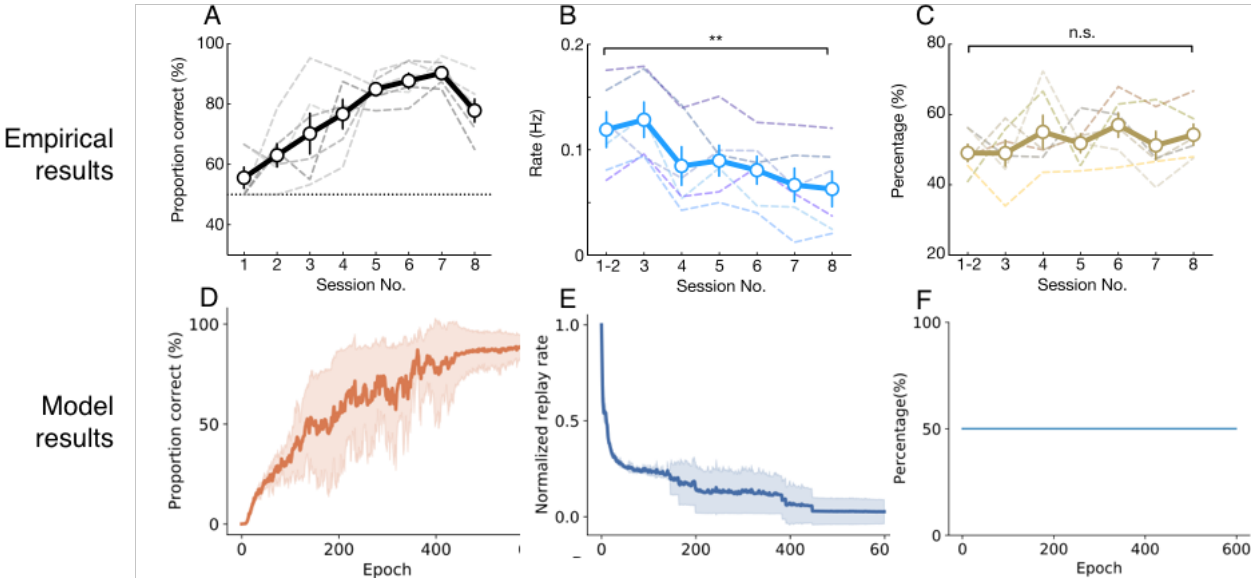

Figure 8: Model behavior in T-maze navigation task in comparison with results from Shin et al. (2019). Top row: **A**, **B** and **C** are adapted from Fig.4 in (Shin et al., 2019). Bottom row: **D**, **E** and **F** are results from the the **consolidator-cache** model. **A**,**D**: Throughout training the model and animal performance both steadily increase. **B**,**E**: During the training process, the replay rate gradually decreases as the performance increases. **C**,**F**: **C** Animals have relatively balanced forward and reverse replay in the all training sessions regardless of performance. **F** In the system of the cache and consolidator, the replay rate is balanced by definition as forward replay involves learning of backward circuits and backward replay trains forward circuits.

tion task, in which the animal needs to make decisions to either turn to the left or right end of the horizontal branch to get a reward based on the visual cue at the beginning of the maze (see Figure 7). Similar to the training paradigm discussed previously, we set the sensory inputs to both **consolidator** and **cache** as a concatenated binary vector (o^t, a^t, r^t) data stream, with o^t for visual observation, a^t for action and r^t for the presence of the reward.

To explore how the model might map on to the learning process in the hippocampus, we mapped and modified several concepts in the machine learning context to animal task training. We first define an uninterrupted phase of replay events as an epoch of training RE_l, in which the system continuously samples from past rewarded experience and reactivates them to train the **consolidator** (Figure 7 **A**). Each training epoch RE_l is then followed by an exploration stage EX_l. During EX_l, the system explores in the environment and makes predictions (forward replays) about future states based on the trained parameters in RE_{l-1}. Some of these generated sequences are rewarded and stored as potential training samples in RE_l. The whole process is therefore divided into blocks made by $\{(EX_1, RE_1), (EX_2, RE_2), \cdots, (EX_l, RE_l)\}$. As the training goes on, the prediction error in each block gradually decreases. As the mean prediction error decreases, the number of replay events in each epoch likewise decreases until its performance converges to a master level.

More specifically, in EX_l, the **consolidator** first generates sensory sequences $S = \{(o^0, a^0, r^0), (o^1, a^1, r^1), \cdots, (o^t, a^t, r^t)\}$ according to its own action units (outputs) at each time step, as part of its prediction of S^t represents the action it will execute in the next time step. Then the **cache** selects input patterns with r^T's that are rewarded and memorizes them through the learning rule tuned by reward amplitude. Once this one-shot process (the first phase) is finished, the RE_l simulating offline learning will start. The **cache** then plays rewarded S's in a reversed fashion, sending inputs to the **consolidator**, which correspondingly performs backward running with the reversed S's.

As the repeated backward running goes on, the **consolidator** optimizes its forward projections f_* according to the reversed prediction error (Eq.4) and adjusts its backward projections g_* with the change of f_* (Eq.2). After training, the {**consolidator**, **cache**} system successfully mastered the task with a correct performance rate above

90% (see Figure. 8 **D**). The performance rise is driven by a series of gradually decreasing replay epochs of training (Figure. 8 **E**). Besides, since this system requires training for both forward and backward circuits in the consolidator, the replay rate for both directions are balanced by definition (Figure. 8 **F**).

These results account for the empirical finding that as the animal gets familiar with the task, the replay events occur less often Shin et al. (2019) (Figure. 8 **A**, **B** and **C**).

To investigate the representation of tasks in the system, we sorted neurons' normalized firing rates according to the distance between their positions with highest firing rate and the start point. Similar to some previous work (Ziv et al., 2013; Driscoll et al., 2017), the results (Figure 7 **B**) show that place cells-like structure emerged after training, and some neurons are biased to crucial positions such as the end of the cue and the turning point.

3 Discussion

The massive success of deep learning models and their similarities with biological neural networks in the behavior and dynamics level has captured the attention of the neuroscience community, with one of the most crucial questions being how BPTT (or more generally, full error gradient learning in recurrent networks) might be implemented in the brain since its connectivity is both hierarchical (multi-layer structure) and recurrent (lateral connection). In addition to the constraint of only local information at synapses, as a seemingly obvious biological constraint in the brain, most of the work trying to solve this problem has focused on the online side (Whittington and Bogacz, 2019; Murray, 2019) as a workaround.

Here we have presented the R2N2 model as a novel approach to the question from another side, an integrated learning system composed of a combined fast (**cache**) and statistical (**consolidator**) sequential learning model. Unlike the online alternatives, our model's underlying principle is that it has a backward phase to compute and assign credits to synapses in recurrent projections without violating the locality constraint. We have shown that the model is capable of running itself in a reversed order and shows improved performance relative to other models based on recurrent weight updates computed in the backward phase in various tasks. In addition, by applying this model

to a rat navigation task we show that it captures several observed phenomena from previous experiments. Computationally, we show the importance of reverse replays as observed in the hippocampus - they may be essential for computing the error feedback signal through time to support supervised sequence learning, in a manner that is similar to BPTT. In addition, the **cache** also memorizes sequences in a one-shot manner, which provides training samples for **consolidator** in the backward phase. The ability to compute the full error feedback signal through time may account for the advantage over previous localist supervised sequence learning models such as echo state network(Maass et al., 2002; Jaeger, 2002) and RFLO (Murray, 2019), as it extends the error gradient farther back in time.

A possible neural realization of this **consolidator-cache** system might be the entorhinal-hippocampus communication system. First, there is empirical evidence showing it is the backward running phase (reverse replay), rather than the forward running phase (forward replay), in the hippocampus during immobilization that is crucial for the animal's later performance after experiencing the task environment (Ambrose et al., 2016). Some other recent studies even show that prolonged reverse replay enhances task performance Fernández-Ruiz et al. (2019), while destroyed reverse replay leads to failures in task performance (Michon et al., 2019). These observations imply the existence of offline backward learning in recurrent neuronal networks, which is conceptually isomorphic with the temporal unfolding process in BPTT. However, many of the existing models (Haga and Fuka, 2018; Evangelista et al., 2020) for reverse and forward replay do not account for sequence learning. They are limited in that they are built on handcrafted attractor connectivity patterns and thus usually have only one or few spatially clustered neurons being active at each moment, which puts a limit (N=number of neurons) on their learning capacity. Instead, the **consolidator** in our model builds connectivity matrices for the reverse replay of arbitrary neuronal activation pattern sequences, which is more flexible and biologically realistic considering the high dimensional nature of spiking activities in the brain. Though in our simulation the speed of forward replay and reverse replay is the same, previous experiments usually show temporally compressed reverse replay in the hippocampus (Diba and Buzsáki, 2007). This could be implemented in the model with faster temporal decay constants in Eq.1.

The **cache** could be implemented in CA3 pyramidal neurons with recurrent lateral excitatory projections, which have the same arbitrary spatial association and pattern completion capability. With a trainer providing reversed sequence samples, the **consolidator**, which could be a circuit in the entorhinal cortex receiving inputs from CA3, could learn statistics in the data stream and solidify the short term memory in the hippocampus to provide longer-lasting memories. The second phase of learning might be triggered and tuned by reward-related signals to ensure the sequence being replayed and learned is rewarded and beneficial for the system, which has been found to be the case in the hippocampus (Ambrose et al., 2016).

Our model bears some similarity to the complementary learning systems (CLS) framework regarding the role of the hippocampus. Typically the hippocampus is cast as the fast learner and the cortex the slower learner (McClelland et al., 1995), and more recently the role of replays has been incorporated into the framework (Kumaran et al., 2016). Our model suggests that the cache and consolidator functions (analogous to fast and statistical learning, respectively) may both be carried out within the hippocampus, as well as between the hippocampus and neocortex. Recent work has similarly argued that both fast and statistical learning may take place within the hippocampus, with the entorhinal cortex to CA1 pathway providing statistical learning, and the pathway from CA1 to dentate gyrus to CA3 providing fast learning (Schapiro et al., 2017). The R2N2 model is consistent with this anatomical delineation but does not exclude other possible functional mappings.

More generally, the R2N2 model suggests the importance of internal clock signals, as the **consolidator** and **cache** each oscillate to generate state updates. This is consistent with a number of hippocampal cell types which show either greatest or smallest activity levels at the peak of the theta cycle or a ripple (Klausberger and Somogyi, 2008).

In this phase, since the **consolidator** is gradually trained to perform reverse replay in a nearly perfect way, we further speculate that the **consolidator** could, in turn, train other similar **consolidator** instances in the cortex in a "bootstrapped" manner to implement distributed knowledge representation across distinct brain regions.

In summary, this article provides a new possible approach for biological RNNs to learn sequential tasks. This model can memorize sequences in a one-shot way and transfer the experience to long-lasting synaptic changes through the reverse replay. When it

comes to cognitive tasks, the **consolidator-cache** model treats different types of tasks under a unified sequence prediction framework and solves it with rewards as a signal for reverse replay. The whole process is based on competitions between different synaptic projections, i.e., the competition between f_* and g_* in the **consolidator** and W_O and W_E in the **cache**, which does not require any non-local information or weight symmetries. These assumptions therefore generate several experimentally testable predictions for future research of sequence learning. First, an imbalanced synaptic projection (e.g., decreased excitatory level in one projection) between different neural assemblies may lead to impaired reverse replay since in our model the reverse replay in the **consolidator** relies on competition of different projections between neuronal groups. Second, as in the **cache** an internally generated pseudo-periodic signal is responsible for the transition between firing pattern attractors, one may expect to see induced reverse replay with external periodic signals acting on the gating neurons for the CA3 network or corrupted reverse replay with interrupted reverse replay with aperiodic perturbations.

4 Materials and methods

In Figure 3, the number of neurons of each neuronal group in **consolidator** is 64. In total 128 neurons are used. The time constant τ is set to 10 in this simulation and is the same in all other experiments across different architectures unless specified differently. The learning rate of backward projections in this simulation is set to 5×10^{-3}. For the simulation of Figure 4, we use the same number of neurons and as in Figure 3. The learning rate for all projections are set to 5×10^{-3}. The input sequence used to test the network's capability to associate temporal changing inputs is a 6 dimensional randomly generated one-hot vector sequences with length 20.

In the performance test across different algorithms on $a^n b^n$ task (Figure 5 **A**), the number of neurons in ESN and RFLO network is set to 64. Since **consolidator** consists of two groups of neurons, we set each group to 32. The learning rate for all layers in RFLO and output layer and input layer in ESN is set to 1×10^{-2}. The learning rate used in **consolidator** with both Backpropagation and Feedback Alignment is set to 1×10^{-3} as higher learning rates failed to converge.

In the simulation of T-maze navigation (Figure 7), **consolidator** is initialized with

64 neurons in each neuronal group. The learning rate is set to 1×10^{-3}.

In the sequence recall test for **cache** (Figure 6), we initialized the network with 500 neurons. The learning rate is set to 2×10^{-3} for both externally and internally controlled cases.

References

Akrout, M., C. Wilson, P. C. Humphreys, T. Lillicrap, and D. Tweed (2019). Deep Learning without Weight Transport. (NeurIPS).

Ambrose, R. E., B. E. Pfeiffer, and D. J. Foster (2016). Reverse Replay of Hippocampal Place Cells Is Uniquely Modulated by Changing Reward. *Neuron 91*(5), 1124–1136.

Chang, B., L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham (2017). Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*.

Depasquale, B., C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott (2018). full-FORCE : A Target-Based Method for Training Recurrent Networks arXiv : 1710 . 03070v1 [cs . NE] 9 Oct 2017. pp. 1–20.

Diba, K. and G. Buzsáki (2007, oct). Forward and reverse hippocampal place-cell sequences during ripples. *Nature Neuroscience 10*(10), 1241–1242.

Driscoll, L. N., N. L. Pettit, M. Minderer, S. N. Chettih, and C. D. Harvey (2017). Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell 170*(5), 986–999.

Eichenlaub, J. B., B. Jarosiewicz, J. Saab, B. Franco, J. Kelemen, E. Halgren, L. R. Hochberg, and S. S. Cash (2020). Replay of Learned Neural Firing Sequences during Rest in Human Motor Cortex. *Cell Reports 31*(5), 107581.

Evangelista, R., G. Cano, C. Cooper, D. Schmitz, N. Maier, and R. Kempter (2020). Generation of sharp wave-ripple events by disinhibition. *Journal of Neuroscience 40*(41), 7811–7836.

Fernández-Ruiz, A., A. Oliva, E. F. de Oliveira, F. Rocha-Almeida, D. Tingley, and G. Buzsáki (2019). Long-duration hippocampal sharp wave ripples improve memory. *Science 364*(6445), 1082–1086.

Haga, T. and T. Fuka (2018). Recurrent network model for learning goal-directed sequences through reverse replay. *eLife 7*, 1–31.

Haga, T. and T. Fukai (2018). Recurrent network model for learning goal-directed sequences through reverse replay. *Elife 7*, e34171.

Hemberger, M., M. Shein-Idelson, L. Pammer, and G. Laurent (2019). Reliable Sequential Activation of Neural Assemblies by Single Pyramidal Cells in a Three-Layered Cortex. *Neuron 104*(2), 353–369.e5.

Jaeger, H. (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, Volume 5. GMD-Forschungszentrum Informationstechnik Bonn.

Klausberger, T. and P. Somogyi (2008, Jul). Neuronal diversity and temporal dynamics: The unity of hippocampal circuit operations. *Science 321*(5885), 53–57.

Kumaran, D., D. Hassabis, and J. L. McClelland (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences 20*(7), 512–534.

Lee, D.-L. (2002). Pattern sequence recognition using a time-varying hopfield network. *IEEE Transactions on Neural Networks 13*(2), 330–342.

Lillicrap, T. P., D. Cownden, D. B. Tweed, and C. J. Akerman (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications 7*, 1–10.

Lillicrap, T. P., A. Santoro, L. Marris, C. J. Akerman, and G. Hinton (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 1–12.

Lo, C.-C. and X.-J. Wang (2006). Cortico–basal ganglia circuit mechanism for a decision threshold in reaction time tasks. *Nature neuroscience 9*(7), 956–963.

Lörincz, A. and G. Buzsáki (2000). Two-phase computational model training long-term memories in the entorhinal-hippocampal region. *Annals of the New York Academy of Sciences 911*(1), 83–111.

Maass, W., T. Natschläger, and H. Markram (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation 14*(11), 2531–2560.

Marschall, O., K. Cho, and C. Savin (2020). A unified framework of online learning algorithms for training recurrent neural networks. *Journal of Machine Learning Research 21*(135), 1–34.

McClelland, J. L., B. L. McNaughton, and R. C. O'Reilly (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review 102*(3), 419.

Michon, F., J.-J. Sun, C. Y. Kim, D. Ciliberti, and F. Kloosterman (2019). Post-learning hippocampal replay selectively reinforces spatial memory for highly rewarded locations. *Current Biology 29*(9), 1436–1444.

Moskovitz, T. H., A. Litwin-Kumar, and L. Abbott (2018). Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*.

Murray, J. M. (2019). Local online learning in recurrent networks with random feedback. *eLife 8*, 1–25.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *arXiv preprint arXiv:1609.01596*.

Parish, G., S. Michelmann, S. Hanslmayr, and H. Bowman (2020). Modelling the replay of dynamic memories from cortical alpha oscillations with the syncfire/desync model. *bioRxiv*.

Pfeiffer, B. E. and D. J. Foster (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature 497*(7447), 74–79.

Poirazi, P., T. Brannon, and B. W. Mel (2003). Pyramidal neuron as two-layer neural network. *Neuron 37*(6), 989–999.

Schapiro, A. C., N. B. Turk-Browne, M. M. Botvinick, and K. A. Norman (2017). Complementary learning systems within the hippocampus: a neural network modelling approach to reconciling episodic memory with statistical learning. *Philosophical Transactions of the Royal Society B: Biological Sciences 372*(1711), 20160049.

Schuck, N. W. and Y. Niv (2019). Sequential replay of nonspatial task states in the human hippocampus. *Science 364*(6447).

Shin, J. D., W. Tang, and S. P. Jadhav (2019). Dynamics of awake hippocampal-prefrontal replay for spatial learning and memory-guided decision making. *Neuron 104*(6), 1110–1125.

Sompolinsky, H. and I. Kanter (1986). Temporal association in asymmetric neural networks. *Physical review letters 57*(22), 2861.

Tallec, C. and Y. Ollivier (2017). Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043*.

Van Der Meer, M. A. and A. D. Redish (2010). Expectancies in decision making, reinforcement learning, and ventral striatum. *Frontiers in neuroscience 3*, 6.

Vaz, A. P., J. H. Wittig, S. K. Inati, and K. A. Zaghloul (2020, mar). Replay of cortical spiking sequences during human memory retrieval. *Science 367*(6482), 1131–1134.

Whittington, J. C. and R. Bogacz (2019). Theories of Error Back-Propagation in the Brain.

Zhang, Z., H. Cheng, and T. Yang (2019). A recurrent neural network model for flexible and adaptive decision making based on sequence learning. *bioRxiv*, 555862.

Ziv, Y., L. D. Burns, E. D. Cocker, E. O. Hamel, K. K. Ghosh, L. J. Kitch, A. El Gamal, and M. J. Schnitzer (2013). Long-term dynamics of ca1 hippocampal place codes. *Nature neuroscience 16*(3), 264.

5 Supplementary Material

5.1 Hidden layer error computation

In our implementation, the specific process to update the postsynpatic component, e_H^t, in Eq.3 is discretized as a difference equation from time $t = 0$ to $t = T$. Consider a simple case in which from $(0, T)$ the task of the network is only to generate a proper output O_T at the last time step $t = T$ and the overall loss function is then defined as $\frac{1}{2}(O^T - O^{\star T})^2$ with $O^{\star T}$ being the desired output. If neuronal group B is connected to the output layer O through a linear mapping W_O, then we have

$$\begin{aligned} e_B^T &= W_O^\top \cdot (O^T - O^{\star T}) \\ e_A^T &= 0 \end{aligned} \tag{8}$$

for the last time step.

For all other previous steps $t \in (0, T-1)$, we have two coupling equations to compute the error signals for both A and B iteratively:

$$\begin{aligned} e_B^t &= \frac{1}{\tau} W_A^\top e_A^{t+1} + (1 - \frac{1}{\tau}) e_B^{t+1} \\ e_A^t &= \frac{1}{\tau} W_B^\top e_B^t + (1 - \frac{1}{\tau}) e_A^{t+1} \end{aligned} \tag{9}$$

We include a leaky term in the the above update equations to maintain consistency with the leaky nature of the activity updates in Eq. 1.

With the help of Feedback alignment algorithm, the $W_O^\top$, $W_A^\top$ and $W_B^\top$ in the above equation can further be replaced by random fixed matrices β_O, β_A and β_B, for $t = T$, then we have:

$$\begin{aligned} \widehat{e_B^T} &= \beta_O \cdot (O^T - O^{\star T}) \\ \widehat{e_A^T} &= 0 \end{aligned} \tag{10}$$

and for $t \in (0, T-1)$:

$$\begin{aligned} \widehat{e_B^t} &= \frac{1}{\tau} \beta_A \widehat{e_A^{t+1}} + (1 - \frac{1}{\tau}) \widehat{e_B^{t+1}} \\ \widehat{e_A^t} &= \frac{1}{\tau} \beta_B \widehat{e_B^t} + (1 - \frac{1}{\tau}) \widehat{e_A^{t+1}} \end{aligned} \tag{11}$$

If we generalize the case of only generating output O^T at T to outputs at each time step $O^1, O^2, \cdots, O^T$, the corresponding e_A and e_B will be similar with each time step

to the neuronal group connected to the output layer receiving an extra non-zero output error term used in Eq. 10

$$\begin{aligned}\widehat{e_B^t} &= \frac{1}{\tau}\beta_A \widehat{e_A^{t+1}} + (1-\frac{1}{\tau})\widehat{e_B^{t+1}} + \beta_O \cdot (O^{t+1} - O^{\star_{t+1}}) \\ \widehat{e_A^t} &= \frac{1}{\tau}\beta_B \widehat{e_B^t} + (1-\frac{1}{\tau})\widehat{e_A^{t+1}}\end{aligned} \tag{12}$$

When connecting with the **Cache** network, since we're taking a sequence prediction paradigm, $O^{\star_1}, O^{\star_2}, \cdots, O^{\star_T}$ will be the activation sequence of the **Cache**, which can perform reverse replay itself to provide the reversed desired output sequence required in Eq. 12.