A solution to the learning dilemma for recurrent networks of spiking neurons

Guillaume Bellec[1,°], Franz Scherr[1,°], Anand Subramoney[1],

Elias Hajek[1], Darjan Salaj[1], Robert Legenstein[1], Wolfgang Maass[1,*]

[1]Institute of Theoretical Computer Science, Graz University of Technology, Inffeldgasse 16b, Graz, Austria

° Equal contributions.
* To whom correspondence should be addressed; E-mail: maass@igi.tugraz.at.

Abstract

Recurrently connected networks of spiking neurons underlie the astounding information processing capabilities of the brain. But in spite of extensive research, it has remained open how learning through synaptic plasticity could be organized in such networks. We argue that two pieces of this puzzle were provided by experimental data from neuroscience. A new mathematical insight tells us how they need to be combined to enable network learning through gradient descent. The resulting learning method – called *e-prop* – approaches the performance of *BPTT* (backpropagation through time), the best known method for training recurrent neural networks in machine learning. But in contrast to *BPTT*, *e-prop* is biologically plausible. In addition, it elucidates how brain-inspired new computer chips – that are drastically more energy efficient – can be enabled to learn.

Introduction

Networks of neurons in the brain differ in at least two essential aspects from deep networks in machine learning: They are recurrently connected by synapses, forming a giant number of loops, and they communicate via asynchronously emitted stereotypical electrical pulses, called spikes, rather than bits or numbers that are produced in a synchronized manner by each layer. Models that capture primary information processing capabilities of spiking neurons in the brain are well known, and we consider the arguably most prominent one: leaky integrate-and-fire (LIF) neurons, where spikes that arrive from other neurons through synaptic connections are multiplied with the corresponding synaptic weight, and are linearly integrated by a leaky membrane potential. The neuron fires – i.e., emits a spike – when the membrane potential reaches a firing threshold.

An important open problem is how recurrent networks of spiking neurons (RSNNs) can learn, i.e., how their synaptic weights can be modified by local rules for synaptic plasticity so that the computational performance of the network improves. In deep learning this problem is solved for feedforward networks through gradient descent for a loss function E that measures imperfections of current network performance (LeCun et al., 2015). Gradients of E are propagated backwards through all layers of the feedforward networks to each synapse through a process called backpropagation. Recurrently connected networks can compute more efficiently because each neuron can participate several times in a network computation, and they are able to solve tasks that require integration of information over time and a suitable timing of network outputs according to task demands. But since a synaptic weight can affect the network computation at several time points during a computation, its impact on the loss function (see Fig. 1A) is more indirect, and learning through gradient descent becomes

substantially more difficult in a recurrent network. In machine learning one had solved this problem 30 years ago by unrolling a recurrent network into a virtual feedforward network, see Fig. 1B, and applying the backpropagation algorithm to it (Fig. 1C). This learning method for recurrent neural networks is called backpropagation through time ($BPTT$).

We show that with a careful choice of the pseudo-derivative for handling the discontinuous dynamics of spiking neurons one can apply this learning method also to RSNNs, yielding the by far best performing learning algorithm for such networks (see Huh and Sejnowski (2018) for related preceding results). But the dilemma is that $BPTT$ requires storing the intermediate states of all neurons during a network computation, and to merge these in a subsequent offline process with gradients that are computed backwards in time (see Fig. 1C and Movie S2). This makes it very unlikely that $BPTT$ is used by the brain (Lillicrap and Santoro, 2019). This dilemma is exacerbated by the fact that neurons in the brain have a repertoire of additional internal dynamic processes on slower time scales that are not reflected in the LIF model, but which are likely to contribute to the superior capabilities of RSNNs in the brain to compute in the temporal domain. In fact, even in machine learning one uses special types of neuron models, called LSTM (Long Short-Term Memory) units, in order to handle such tasks. But any neuron model that has additional internal processes, and hence more hidden variables that capture their current state, makes learning in a recurrent network of such neurons even more difficult.

We present an approach for solving this dilemma: *e-prop* (Fig. 1D and 1E, see Movie S3). It can be applied not only to RSNNs, but also to recurrent networks of LSTM units and most other types of recurrent neural networks. We focus on the application of *e-prop* to RSNNs that have, besides LIF neurons, also a more sophisticated form of LIF neurons, called ALIF neurons. An ALIF neuron has a second hidden variable besides its membrane potential: an adaptive firing threshold. The firing threshold of an ALIF neuron increases through each ot its spikes and decays back to a resting value between spikes. This models firing rate adaptation, a well known feature of a fraction of neurons in the brain (Allen Institute: Cell Types Database, 2018) that dampens their firing activity. We refer to an RSNN that contains a fraction of ALIF neurons as a Long short-term memory Spiking Neural Network (LSNN), because we show that ALIF neurons provide a qualitative jump in temporal computing capabilities of RSNNs, allowing RSNNs to approach for the first time the performance of LSTM networks in machine learning for temporal processing tasks.

E-prop is motivated by two streams of experimental data from neuroscience that can be seen as providing hints how the brain solves the learning dilemma for RSNNs:

i) The dynamics of neurons in the brain is enriched by continuously ongoing updates of traces of past activity on the molecular level, for example in the form of calcium ions or activated CaMKII enzymes(Sanhueza and Lisman, 2013). These traces in particular record events where the presynaptic neuron fired before the postsynaptic neuron, which is known to induce Hebbian-like STDP (spike timing dependent plasticity) if followed by a top-down learning signal (Cassenaer and Laurent, 2012; Yagishita et al., 2014; Gerstner et al., 2018). We refer to local traces of this type as eligibility traces in our learning model.

ii) In the brain there exists an abundance of top-down signals such as dopamine and acetylcholine, to name only a few, that inform local populations of neurons about sub-optimal performance of brain computations. Interestingly some of these signals are of a predictive nature, e.g. they predict upcoming rewards in the case of dopamine or movement errors in the case of the error-related negativity (ERN), see MacLean et al. (2015). Furthermore both dopamine signals (Engelhard et al., 2019; Roeper, 2013) and ERN-related neural firing (Sajad et al., 2019) are reported to be specific for a target population of neurons, rather than global. We refer to such top-down signals as learning signals in our learning model.

Our re-analysis of the mathematical basis of gradient descent in recurrent neural networks in equ. (1) tells us how eligibility traces and learning signals need to be combined to produce network learning through gradient descent – without backprogation of signals through time or retrograde through synaptic connections. We will show that the resulting new learning method, *e-prop*, approximates the performance of $BPTT$ for RSNNs, thereby providing a solution to the learning dilemma for RSNNs. We demonstrate this on tasks for supervised

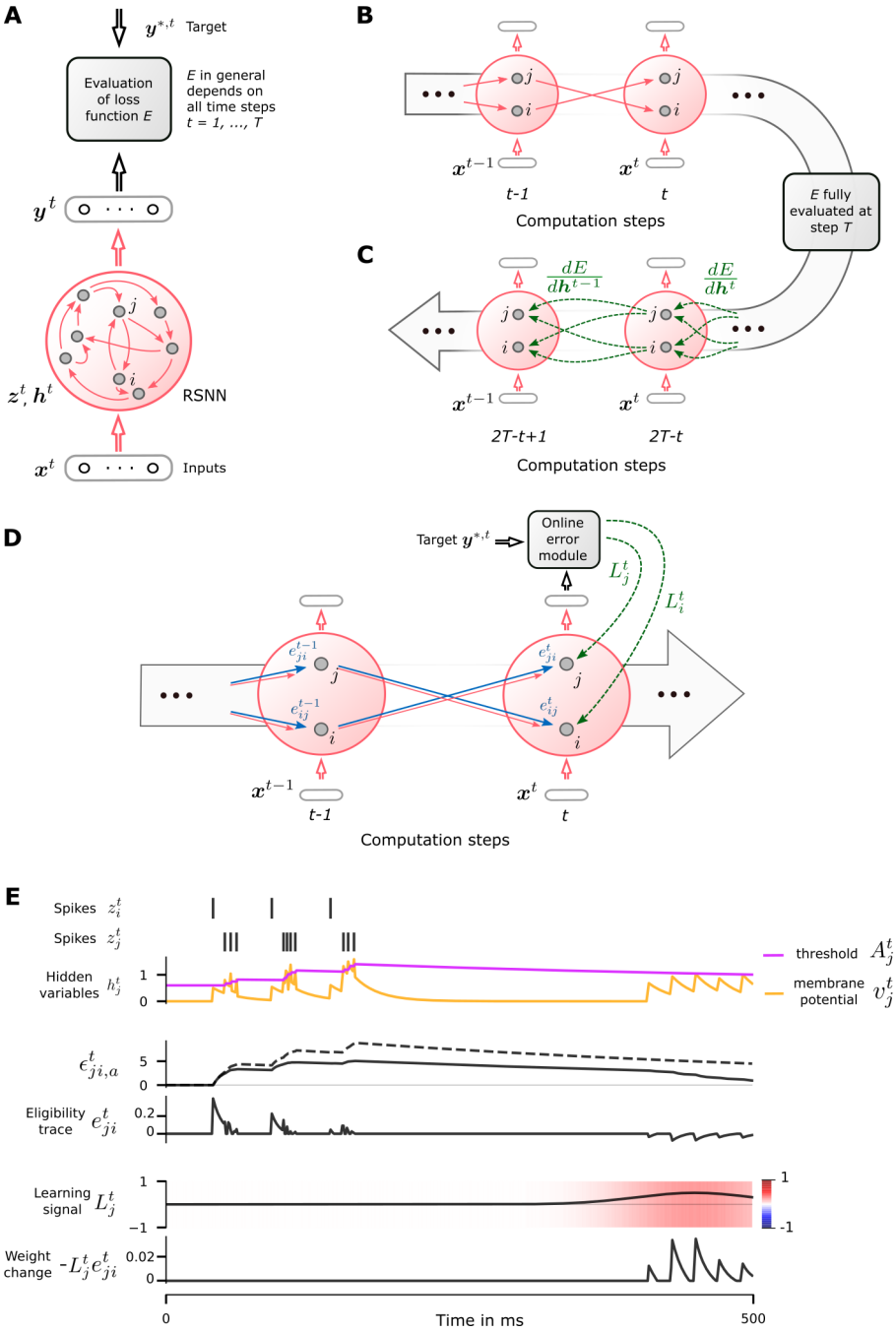

Figure 1: (Caption on the next page.)

Figure 1: **Schemes for RSNNs, *BPTT*, and *e-prop*.** **A**) RSNN with network inputs x, neuron spikes z, and output targets y^*, for each time step t of the RSNN computation. Output neurons y provide a low-pass filter of network spike z. **B**) BPTT computes gradient in the unrolled version of the network. It has a copy of all neurons of the RSNN for each time step t. A synaptic connection from neuron i to neuron j of the RSNN is replaced by an array of feedforward connections, one for each time step t, that goes from the copy of neuron i in the layer for time step t to a copy of neuron j in the layer for time step $t+1$. All synapses in this array have the same weight: the weight of this synaptic connection in the RSNN. **C**) Loss gradients of *BPTT* are propagated backwards in time and retrograde across synapses in an offline manner, long after the forward computation has passed a layer. **D**) Online learning dynamics of *e-prop*. Feedforward computation of eligibility traces is indicated in blue. These are combined with online learning signals according to equ. (1). **E**) Illustration of the dynamics of ALIF neurons and e-prop. Observable variables (spikes) $\boldsymbol{z}^t$ and hidden variables of an ALIF neuron, slow factor $\epsilon^t_{ji,a}$ (equation (22)) of the eligibility trace e^t_{ji} (equation (23)) of the synapse from neuron i to neuron j, as well as a learning signal L^t_j and the resulting online weight change proposed by e-prop. In this case a late activation of a learning signal, such as dopamine in the experiments of Yagishita et al. (2014), it transforms the eligibility trace into the modification of the synaptic weight. The dashed curve above the plot of $\epsilon^t_{ji,a}$ shows an easily computable approximation (see equation (24)) of $\epsilon^t_{ji,a}$ as low-pass filter of STDP-inducing spiking events that can be used for an approximation of *e-prop*.

learning (Fig. 2,3) and reinforcement learning (Fig. 4). None of these tasks were previously known to be solvable by RSNNs.

The previously described learning dilemma for RSNNs also affects the development of new, brain inspired computing hardware, which aims at a drastic reduction in the energy consumption of computing and learning. Resulting new designs of computer chips, such as Intel's Loihi (Davies et al., 2018), are usually focused on RSNN architectures. On-chip learning capability for these RSNNs in the hardware is essential. Although it does not matter here whether the learning algorithm is biologically plausible, the excessive storage and offline processing demands of BPTT make this option unappealing for such novel computing hardware also. Hence a corresponding learning dilemma exists also there. *E-prop* does not contain any features that make it unlikely to be implementable on such neuromorphic chips, thereby promising a solution also for this learning dilemma.

Results

Mathematical basis for *e-prop*

Spikes are modeled as binary variables z^t_j that assume value 1 if neuron j fires at time t, otherwise value 0. It is common to let t vary over small discrete time steps, e.g. of 1ms length. The goal of network learning is to find synaptic weights W that minimize a given loss function E. E may depend on all or a subset of the spikes in the network. E measures in the case of regression or classification learning the deviation of the actual output y^t_k of each output neuron k at time t from its given target value $y^{*,t}_k$ (Fig. 1A). In reinforcement learning (RL), the goal is to optimize the behavior of an agent in order to maximize obtained rewards. In this case, E measures deficiencies of the current agent policy to collect rewards.

The gradient $\frac{dE}{dW_{ji}}$ for the weight W_{ji} of the synapse from neuron i to neuron j tells us how this weight should be changed in order to reduce E. The key observation for *e-prop* (see proof in Methods) is that this gradient can be represented as a sum over the time steps t of the RSNN computation: A sum of products of learning signals L^t_j (specific for the post-synaptic neuron j of the corresponding synapse) and synapse-specific eligibility traces e^t_{ji}:

$$\frac{dE}{dW_{ji}} = \sum_t L^t_j \; e^t_{ji} \; . \tag{1}$$

The ideal value of L^t_j is the derivative $\frac{dE}{dz^t_j}$, which tells us how the current spike output z^t_j of

neuron j affects E. In contrast, the eligibility trace e_{ji}^t does not depend on E, but on the internal dynamics of neuron j and the spikes of neuron i. It tells us how a change of the weight W_{ji} would affect the spike output z_j^t when considering the history of the interaction of i and j and ignoring other connections in the network (see equation (S2)).

We view (1) as a program for online learning: In order to reduce E, change at each step t of the network computation all synaptic weights W_{ji} proportionally to $-L_j^t e_{ji}^t$ (see Fig. 1E for an illustration). There is no need to explicitly compute or store the sum (1), or to wait for later signals. Hence *e-prop* is an online learning method in a strict sense (see Fig. 1D and Movie S3). In particular, there is no need to unroll the network as for *BPTT*. Furthermore, in contrast to the previously known real time recurrent learning algorithm (RTRL, see Williams and Zipser (1989) and Methods), which substantially increases the required number of multiplications as function of network size, *e-prop* is – up to a constant factor – not more costly than the RSNN computation itself. This is obviously an important issue both for biological plausibility and neuromorphic implementations.

Since the ideal value $\frac{dE}{dz_j^t}$ of the learning signal L_j^t also captures influences which the current spike output z_j^t of neuron j may have on E via future spikes of other neurons, its precise value is in general not available at time t. We replace it by an approximation that ignores these indirect influences: Only currently arising errors at the output neurons k of the RSNN are taken into account, and are routed with neuron-specific weights B_{jk} to the network neurons j, (see Fig. 2A):

$$L_j^t = \sum_k B_{jk} \underbrace{(y_k^t - y_k^{*,t})}_{\substack{\text{error of output } k \\ \text{at time } t}} \quad . \tag{2}$$

Although this signal L_j^t only captures errors that arise at the current time step t, it is combined in equation (1) with an eligibility trace e_{ji}^t that may reach far back into the past of the target neuron j (see Fig.1E). In this way *e-prop* alleviates the need to propagate signals backwards in time.

There are several strategies for choosing the weights B_{jk} for this online learning signal. In *symmetric e-prop* we set it equal to the corresponding output weight W_{kj}^{out} from neuron j to output neuron k. This learning signal is closest to the theory, and would be theoretically optimal in the absence of recurrent connections. Biologically more plausible are two variants that avoid weight sharing: If all network neurons j are connected to output neurons k, we let B_{jk} evolve in *adaptive e-prop* through a simple local plasticity rule that mirrors the plasticity rule applied to W_{kj}^{out}. In *random e-prop* the values of the weights B_{jk} are randomly chosen and remain fixed, similar to broadcast alignment for feedforward networks (Lillicrap et al., 2016; Nøkland, 2016). Resulting synaptic plasticity rules (see Methods) look very similar to previously proposed plasticity rules (Gerstner et al., 2018). In particular they involve postsynaptic depolarization as one of the factors, similarly as the data-based rule in Clopath et al. (2010), see section S6 in the supplement for an analysis.

We finally would like to mention that the Learning-to-Learn approach can be used to train a separate neural network to generate – instead of the previously considered options – tailormade learning signals for a limited range of potential learning tasks. This variation of *e-prop* enables for example one-shot learning of new arm movements (Bellec et al., 2019).

Comparing the performance of *e-prop* and *BPTT* on a common benchmark task

The speech recognition task TIMIT (Garofolo et al., 1993) is one of the most commonly used benchmarks for temporal processing capabilities of different types of recurrent neural networks and different learning approaches Greff et al. (2017). It comes in two versions. Both use, as input, acoustic speech signals from sentences that are spoken by 630 speakers from 8 dialect regions of the USA (see the top of Fig. 2B for a sample segment). In the simpler version, used for example in Greff et al. (2017), the goal is to recognize which of 61 phonemes is spoken in each 10 ms time frame ("frame-wise classification"). In the harder version from Graves et al. (2013), which achieved an essential step toward human-level performance in speech-to-text transcription, the goal is to recognize the sequence of phonemes in the entire spoken sentence independently of their timing ("sequence transcription"). *E-prop* approximates the performance of *BPTT* on LSNNs for both versions of TIMIT very well, as shown in Fig. 2C.

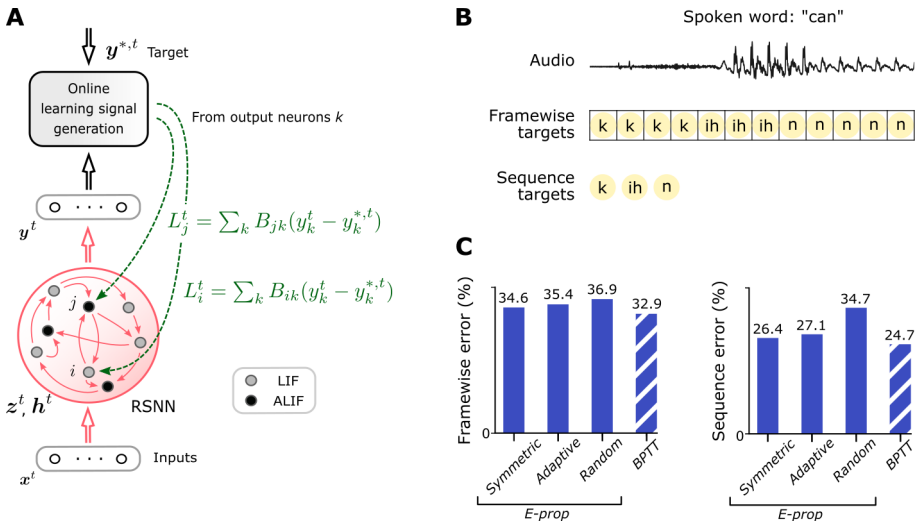

Figure 2: **Comparison of the performance of *BPTT* and *e-prop* on TIMIT. A**) Network architecture for *e-prop*, illustrated for an LSNN consisting of LIF and ALIF neurons. **B**) Input and target output for the two versions of TIMIT. **C**) Performance of *BPTT* and the three versions of *e-prop* for LSNNs consisting of 800 neurons for framewise targets and 2400 for sequence targets.

For the more difficult version of TIMIT we trained as in Graves et al. (2013) a complex LSNN consisting of a feedforward sequence of three recurrent networks. Our results show that *e-prop* can also handle learning for such more complex network structures very well. In Fig. S2 we show for comparison also the performance of LSTM networks. These data show that for both versions of TIMIT the performance of LSNNs comes rather close to that of LSTM networks. This has previously not been demonstrated for any type of RSNN with any learning method on a real-world benchmark task for temporal processing. Furthermore LSNNs could solve the frame-wise classification task without any neuron firing more frequently than 12Hz (spike count taken over 32 spoken sentences), demonstrating that they operate in an energy efficient spike-coding, rather than a rate-coding regime. The FORCE method of Nicola and Clopath (2017) is the best performing previously known learning method for RSNNs. However this learning method was not argued to be biologically realistic, since the plasticity rule for each synaptic weight required knowledge of the current values of all other synaptic weights in the RSNNs. It was applied in Nicola and Clopath (2017) to supervised learning of several pattern generation task. We show in Figs. S1 and S5 that RSNNs can learn such tasks also with *e-prop*, hence without the biologically unrealistic feature of FORCE. We show in Fig S2 that *e-prop* can not only be applied to RSNNs, but also to LSTM networks – and many other types of recurrent networks – that fit under the quite general model discussed in Methods. Furthermore, *e-prop* approximates the performance of BPTT very well for LSTM networks as well (Fig. S2).

E-prop performance for a task where temporal credit assignment is difficult

A hallmark of cognitive computations in the brain is the capability to go beyond a purely reactive mode, to integrate diverse sensory cues over time, and to wait until the right moment arrives for an action. A large number of experiments in neuroscience analyze neural coding after learning for such tasks. But it had remained unknown how one can model the underlying learning processes in RSNNs of the brain. We wondered whether *e-prop* can fill this void. As an example we consider the task that was studied in the experiments of Morcos and Harvey (2016) and Engelhard et al. (2019). There a rodent learnt to run along a linear track in a virtual environment, where it encountered several visual cues on the left and right, see Fig. 3A and Movie S2. Later, when it arrived at a T-junction, it had to decide whether to turn left or right. It was rewarded when it turned to that side from which it had previously received the

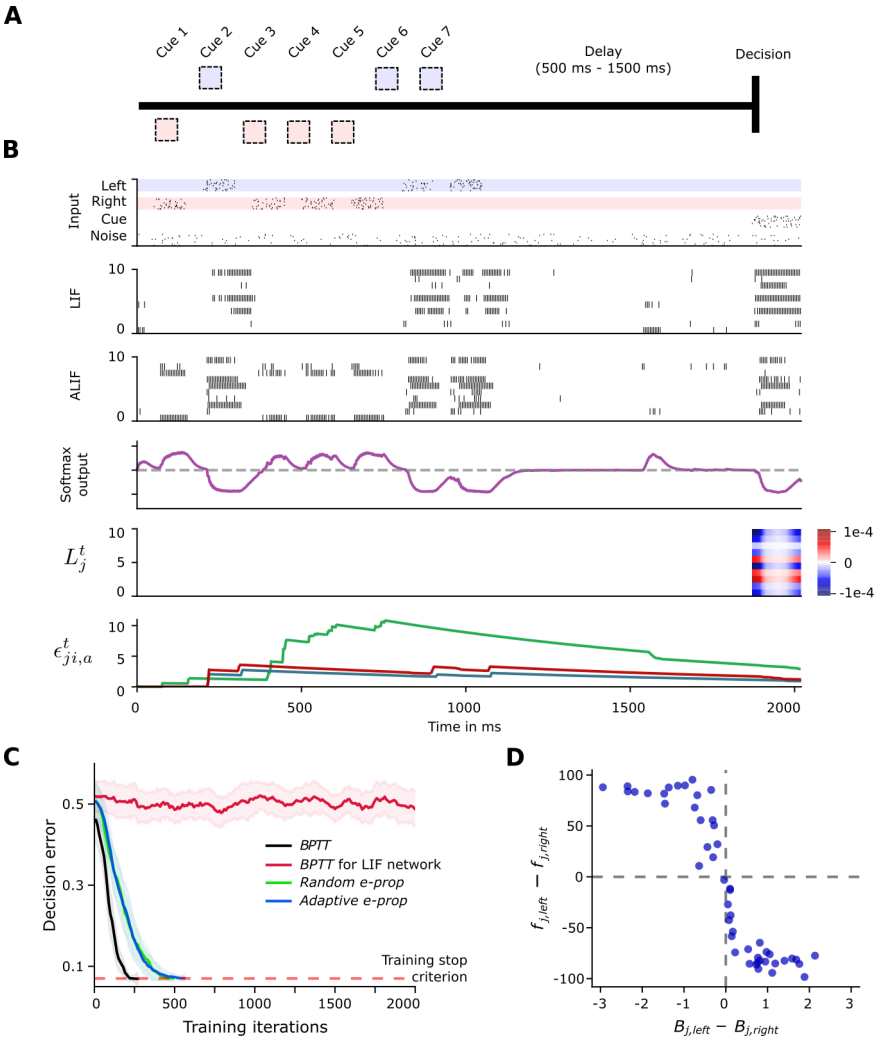

Figure 3: **Solving a task with difficult temporal credit assignment by *e-prop*.** **A**) Setup of corresponding rodent experiments of Morcos and Harvey (2016) and Engelhard et al. (2019), see Movie S2. **B**) Input spikes, internal spiking activity of 10 out of 50 sample LIF neurons and 10 out of 50 sample ALIF neurons, softmax output, sample learning signals and samples of slow components of eligibility traces in the bottom row. **C**) Learning curves for *BPTT* and two *e-prop* versions. **D**) Correlation between the broadcast weights B_{jk} for k = left/right for learning signals in *random e-prop* and sensitivity to "left" and "right" input components after learning. $f_{\mathrm{j,left}}$ ($f_{\mathrm{j,right}}$) is the resulting average firing rate of neuron j during presentation of left (right) cues after learning.

majority of visual cues. This task is not easy to learn since the subject needs to find out that it does not matter on which side the last cue was, or in which order the cues were presented. Instead, the subject has to learn to count cues separately for each side and to compare the two resulting numbers. Furthermore the cues need to be processed long before a reward is given. We show in Fig. S4 that LSNNs can learn this task through *reward-based e-prop*. But since the solution provided by *e-prop* to the temporal credit assignment problem is easier to explain with the supervised learning variation of this task, we discuss below the case where a teacher tells the subject at the end of each trial what would have been the right decision. This still yields a really challenging scenario for *e-prop* since non-zero learning signals L_j^t arise only during the last 150ms of a trial (Fig. 3B). Hence all synaptic plasticity of *e-prop* has to take place during these last 150ms, long after the relevant computations on input cues had been carried out. The result of training an LSNN with *BPTT* and *e-prop* for solving this task is shown in Fig. 3C (illustrated in Movies S3 and S4). Whereas this task can not even be solved by *BPTT* with a regular RSNN that has no adapting neurons (red curve), all 3 previously discussed variations of *e-prop* can solve it if the RSNN contains adapting neurons. We also explain in section S2.4 how this task can be solved for sparsely connected LSNNs when biologically inspired stochastic rewiring (Kappel et al., 2018) is integrated into *e-prop*.

But how can the neurons in the LSNN learn to record and count the input cues if all the learning signals are identically 0 until the last 150ms (5th row of Fig. 3B)? The solution is indicated in the bottom row of Fig. 3B: The slow component $\epsilon^t_{ji,a}$ (equation (22)) of the eligibility traces e_{ji} of adapting neurons j decays with the long time constant of firing rate adaptation (see equation (27) and Movie S4), that typically lies in the range of seconds. Since these traces stretch from the beginning of the trial into its last phase, they enable assignment of credit to firing events that happened over 1000 ms ago. Fig. 3D provides insight into the functional role of the broadcast weights of *random e-prop* in this context: The difference of these weights determines for each neuron j whether it learns to respond in the first phase of a trial more to cues from the left or right. This observation suggests that neuron-specific learning signals for RSNNs have the advantage that they can create a variety of feature detectors for task-relevant network inputs. Hence a suitable weighted sum of these feature detectors is able to cancel remaining errors at the network output, similarly as in the case of feedforward networks (Lillicrap et al., 2016).

Reward-based e-prop

Deep RL has recently produced really powerful results in machine learning and AI through clever applications of *BPTT* to RL (Mnih et al., 2016). We found that one of the arguably most powerful RL methods within the range of deep RL approaches that are not directly biologically implausible, policy gradient in combination with actor-critic, can be implemented with *e-prop*. This yields the biologically plausible RL algorithm *reward-based e-prop*. The LSNN learns through *reward-based e-prop* both an approximation to the value function and a stochastic policy. Neuron-specific learning signals are combined in *reward-based e-prop* with a global signal that transmits reward prediction errors (Fig. S3). In contrast to the supervised case where the learning signals depend on the deviation from an external target signal, the learning signals here are emitted when an action is taken and they express here how much this action deviates from the action mean that is currently proposed by the network. We show in Methods that *reward-based e-prop* yields local reward-based rules for synaptic plasticity that are in many aspects similar to ones that have previously been discussed in the literature (Gerstner et al., 2018). But those previously proposed rules estimated gradients of the policy essentially by correlating the noisy output of network neurons with rewards, which is known to be inefficient due to noisy gradient estimates. In contrast, *reward-based e-prop* computes policy- and value-gradients by approximating *BPTT*, which is one of the pillars of modern deep RL.

We tested *reward-based e-prop* on a task that captures the essence of numerous learning experiments in systems neuroscience: A delayed goal-directed movement has to be learnt, consisting of a sequence of many 2-dimensional continuous motor commands, each of them being only loosely linked to rewards. We chose a setup where the agent first receives a spatial goal cue (Fig. 4A), then has to control the angles of a two-joint arm during a delay so that its tip remains – in spite of motor noise that result from the stochastic policy – within a center

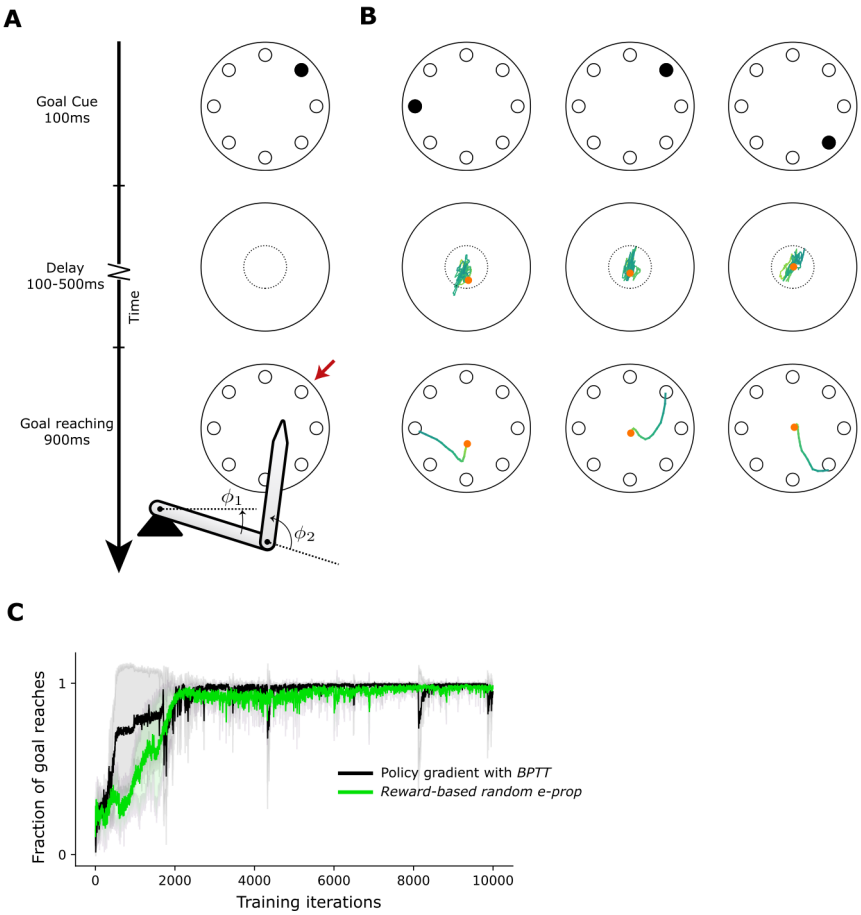

Figure 4: **Application of *e-prop* to RL. A**) Scheme of the delayed arm movement task. The red arrow points to the formerly visible goal. The arm always starts moving from the center of the circle. **B**) Resulting arm movement in three sample trials after learning. The orange dot indicates the position of the tip of the arm at the end of the delay period. **C**) Performance of *reward-based random e-prop* and of a control where *e-prop* is replaced by *BPTT*, both for an LSNN consisting of 350 LIF and 150 ALIF neurons. Solid curves show the mean over 5 different runs, and shaded area indicates 1 standard deviation.

region (indicated by a dotted circle) in order to avoid small negative rewards, until it receives a go-cue (see Movie S5). The agent then has to move the tip of the arm to the location of the initial goal cue in order to receive a reward. Note that no forward- or inverse model of the arm was given to the LSNN, it had to learn those implicitly. This task had so far been beyond the reach of biologically plausible learning, for any type of neural network model.

Three sample trials after learning are shown in Fig. 4B (and in Movie S6). Fig. 4C shows that *reward-based e-prop* is able to solve this demanding RL task about as well as policy gradient with biologically implausible *BPTT*. We conjecture that variants of *reward-based e-prop* will be able to solve most RL tasks that can be solved by online actor-critic methods in machine learning.

Discussion

We propose that in order to understand the computational function and neural coding of higher brain areas, one needs to understand the organization of the plasticity mechanisms that install and maintain the computational functions of the underlying RSNNs. So far BPTT was the only candidate for that, since no other learning method provided sufficiently powerful computational function to RSNN models. But since *BPTT* is not viewed to be biologically realistic (Lillicrap and Santoro, 2019), it does not help us to understand the organization of synaptic plasticity in RSNNs of the brain. *E-prop* offers a solution to this dilemma, since it does not require biologically unrealistic mechanisms, but still enables RSNNs to learn difficult computational tasks almost as well as *BPTT*. Furthermore it enables RSNNs to solve these tasks in an energy efficient sparse firing regime, rather than resorting to rate coding. In particular, we have shown in Fig. 3 and 4 that *e-prop* enables us to model for the first time the learning processes in RSNNs of the brain that underlie the emergence of complex behaviors in key experiments of systems neuroscience.

E-prop relies on two types of signals that are abundandly available in the brain, but whose precise role for learning have not yet been understood: eligibility traces and learning signals. Since *e-prop* is based on a transparent mathematical principle, it provides a normative model for both types of signals, as well as for synaptic plasticity rules. In particular, it suggests a new rule for the organization of eligibility traces: that the time constant of the eligibility trace for a synapse is correlated with the time constant for the history-dependence of the firing activity of the postsynaptic neuron. It also suggests that the experimentally found diverse time constants of the firing activity of populations of neurons in different brain areas (Runyan et al., 2017) are correlated with their capability to handle corresponding ranges of delays in temporal credit assignment for learning. Finally, *e-prop* theory suggests that learning signals for different populations of neurons should be diverse, rather than uniform and global (see section S6.2), and should be correlated with the impact which the activity of these neurons has on the quality of the learnt behavior.

Apart from these consequences of *e-prop* for research in neuroscience and cognitive science, *e-prop* also provides an interesting new tool for approaches in machine learning where *BPTT* is replaced by approximations in order to improve computational efficiency. For example, the combination of eligibility traces from *e-prop* with synthetic gradients from Jaderberg et al. (2016) substantially improves performance of LSTM networks for difficult machine learning problems such as the copy-repeat task and the Penn Treebank word prediction task (Bellec et al., 2019).

Finally, *E-prop* suggests a viable new approach for on-chip learning of RSNNs on neuromorphic chips. Whereas *BPTT* is not within the reach of current neuromorphic chip designs, an implementation of *e-prop* appears to offer no serious hurdle. Since we have shown in Fig. 2 that *e-prop* enables RSNNs to learn to understand speech, and in Fig. 4 that *e-prop* enables reward-based learning of the control of complex arm movements, *e-prop* promises to support a qualitative jump in on-chip learning capabilities of neuromorphic chips.

Methods

To exhibit the theory around *e-prop* and preceding related work, we structure the methods section in the following way:

- Comparison of *e-prop* with other online learning methods for recurrent neural networks (RNNs)
- Network models
- Conventions
- Mathematical basis for *e-prop*
- Eligibility traces
- Eligibility traces for concrete neuron models
- Derivation of the synaptic plasticity rules resulting from *e-prop*
- *Reward-based e-prop*: application of *e-prop* to policy gradient RL.

Comparison of *e-prop* with other online learning methods for recurrent neural networks (RNNs)

In this section we compare *e-prop* with other learning algorithms implementing gradient descent in RNNs without BPTT. A well-known alternative to *BPTT* is real time recurrent learning (RTRL). RTRL was derived for networks of rate-based (sigmoidal) neurons in Williams and Zipser (1989). There, the loss gradients are computed forward in time by multiplying the full Jacobian $\mathbf{J}^t_{kk'} = \frac{d\mathbf{h}^t_k}{d\mathbf{h}^{t-1}_{k'}}$ of the network dynamics with the tensor $\frac{d\mathbf{h}^t_k}{dW_{ji}}$ that computes the dependency of the state variables with respect to the parameters: $\frac{d\mathbf{h}^t_k}{dW_{ji}} = \sum_{k'} \mathbf{J}^t_{kk'} \cdot \frac{d\mathbf{h}^{t-1}_{k'}}{dW_{ji}} + \frac{\partial \mathbf{h}^t_k}{\partial W_{ji}}$ (see equation (12) in Williams and Zipser (1989)). Denoting with n the number of neurons, this requires $O(n^4)$ multiplications, which is computationally prohibitive. Unbiased Online Recurrent Optimization (Tallec and Ollivier, 2018) (UORO) used an unbiased estimator of $\mathbf{J}^t_{kk'}$ of rank one that can be computed online. The authors report that the variance of this estimator increases with the network size and simulations were only carried out for a network size up to 64. Another unbiased estimator of $\mathbf{J}^t_{kk'}$ (Mujika et al., 2018) based on Kronecker factors solved this issue and made it possible to approach the performance of *BPTT* on harder tasks. Yet this method requires $O(n^3)$ operations per time step, which is one order more than UORO, *e-prop* or *BPTT*.

In *e-prop*, the eligibility traces are just $d \times d$ matrices (d being the dimension of $\mathbf{h}^t_j$), since they are restrictions of the full Jacobian $\mathbf{J}^t_{kk'}$ to the internal dynamics of a neuron ($k = k'$). As a consequence, only $O(n^2)$ multiplications are required for the forward propagation of eligibility traces. Hence their computation is not more costly than *BPTT* or the simulation of the RNN.

The learning rule called Superspike (Zenke and Ganguli, 2018) was derived by applying RTRL in spiking neural networks without recurrent connections. In the absence of these connections RTRL is practicable and the resulting learning rule uses eligibility traces similar to those arising in *e-prop* with LIF neurons. Two other algorithms, Roth et al. (2019) and Murray (2019), were introduced to train recurrent neural networks of sigmoidal units by approximating RTRL with another form of eligibility traces. Random Feedback Local Online (RFLO) learning(Murray, 2019) is equivalent to *random e-prop* in the particular case of leaky sigmoidal neurons for regression tasks. But the performance of RFLO was not compared to *BPTT* on published benchmarks for RNNs, or for spiking neurons. In contrast to the eligibility traces in *e-prop*, the eligibility traces in kernel RNN learning (keRNL)(Roth et al., 2019) are viewed as components of an estimator of the tensor $\mathbf{J}^t_{kk'}$, and are not related to the specific definition of the neuron model. This approach requires non-local communication within the RNN, which we wanted to avoid in *e-prop*. In contrast to *e-prop*, none of the papers above (Zenke and Ganguli, 2018; Murray, 2019; Roth et al., 2019) derived a theory or a definition of eligibility traces that can be applied to neuron models with a non-trivial internal dynamics, such as adaptive neurons or LSTM units, that appear to be essential for solving tasks with demanding temporal credit assignment of errors.

Network models

To exhibit the generality of the *e-prop* approach, we define the dynamics of recurrent neural networks using a general formalism that is applicable to many recurrent neural network models, not only to RSNNs and LSNNs. Also non-spiking models such as LSTM networks fit under this formalism (see Section S4.3 in the Supplement). The network dynamics is summarized by the computational graph in Fig. 5. It uses the function M to define the update of the hidden state: $\mathbf{h}_j^t = M(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$, and f to define the update of the observable state: $z_j^t = f(\mathbf{h}_j^t, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$ (f simplifies to $z_j^t = f(\mathbf{h}_j^t)$ for LIF and ALIF neurons).

RSNNs. RSNNs are recurrently connected networks of leaky integrate-and-fire (LIF) neurons. Each LIF neuron has a one dimensional internal state h_j^t that consists only of the membrane potential v_j^t. The observable state $z_j^t \in \{0, 1\}$ is binary, indicating a spike ($z_j^t = 1$) or no spike ($z_j^t = 0$) at time t. The dynamics of the LIF model is defined by the equations:

$$v_j^{t+1} = \alpha v_j^t + \sum_{i \neq j} W_{ji}^{\text{rec}} z_i^t + \sum_i W_{ji}^{\text{in}} x_i^{t+1} - z_j^t v_{\text{th}} \tag{3}$$

$$z_j^t = H\left(v_j^t - v_{\text{th}}\right), \tag{4}$$

where $x_i^t = 1$ indicates a spike from the input neuron i at time step t ($x_i^t = 0$ otherwise) and W_{ji}^{rec} (W_{ji}^{in}) is the synaptic weight from network (input) neuron i to neuron j. The decay factor α in (3) is given by $e^{-\delta t/\tau_m}$, where δt is the discrete time step size (1 ms in our simulations) and $\tau_m = 20$ ms is the membrane time constant. H denotes the Heaviside step function.

Due to the term $-z_j^t v_{\text{th}}$ in equation (3), the neurons membrane potential is reduced by a constant value after an output spike, which relates our model to the spike response model(Gerstner et al., 2014). To introduce a simple model of neuronal refractoriness, we further assume that z_j^t is fixed to 0 after each spike of neuron j for a short refractory period of 2 to 5ms depending on the simulation.

LSNNs. LSNNs are recurrently connected networks that consist of LIF neurons and of adaptive LIF (ALIF) neurons. An ALIF neuron has a time-dependent threshold adaptation a_j^t. As a result, their internal state is a 2 dimensional vector $\mathbf{h}_j^t \stackrel{\text{def}}{=} [v_j^t, a_j^t]$. Their threshold potential A_j^t increases with every output spike and decreases exponentially back to the baseline threshold v_{th}. This can be described by

$$A_j^t = v_{\text{th}} + \beta a_j^t \ , \tag{5}$$

$$z_j^t = H(v_j^t - A_j^t) \ , \tag{6}$$

with a threshold adaptation according to

$$a_j^{t+1} = \rho a_j^t + z_j^t \ , \tag{7}$$

where the decay factor ρ is given by $e^{-\delta t/\tau_a}$, and τ_a is the adaptation time constant that is typically chosen to be in the range of the time span of the length of the working memory that is a relevant for a given task. This is a very simple model for a neuron with spike frequency adaptation. We refer to (Gerstner et al., 2014; Pozzorini et al., 2015; Gouwens et al., 2018) for experimental data and other neuron models.

In relation to the more general formalism represented in the computational graph in Fig. 5, equations (3) and (7) define $M(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$, and equations (4) and (6) define $f(\mathbf{h}_j^t)$.

Gradient descent for RSNNs. Gradient descent is problematic for spiking neurons because of the step function H in equation (4). We overcome this issue as in (Esser et al., 2016; Bellec et al., 2018): the non-existing derivative $\frac{\partial z_j^t}{\partial v_j^t}$ is replaced in simulations by a simple non-linear function of the membrane potential that is called the pseudo-derivative. Outside of the refractory period, we choose a pseudo-derivative of the form $\psi_j^t = \frac{1}{v_{\text{th}}} \gamma_{\text{pd}} \max\left(0, 1 - \left|\frac{v_j^t - A_j^t}{v_{\text{th}}}\right|\right)$ where $\gamma_{\text{pd}} = 0.3$. During the refractory period the pseudo derivative is set to 0.

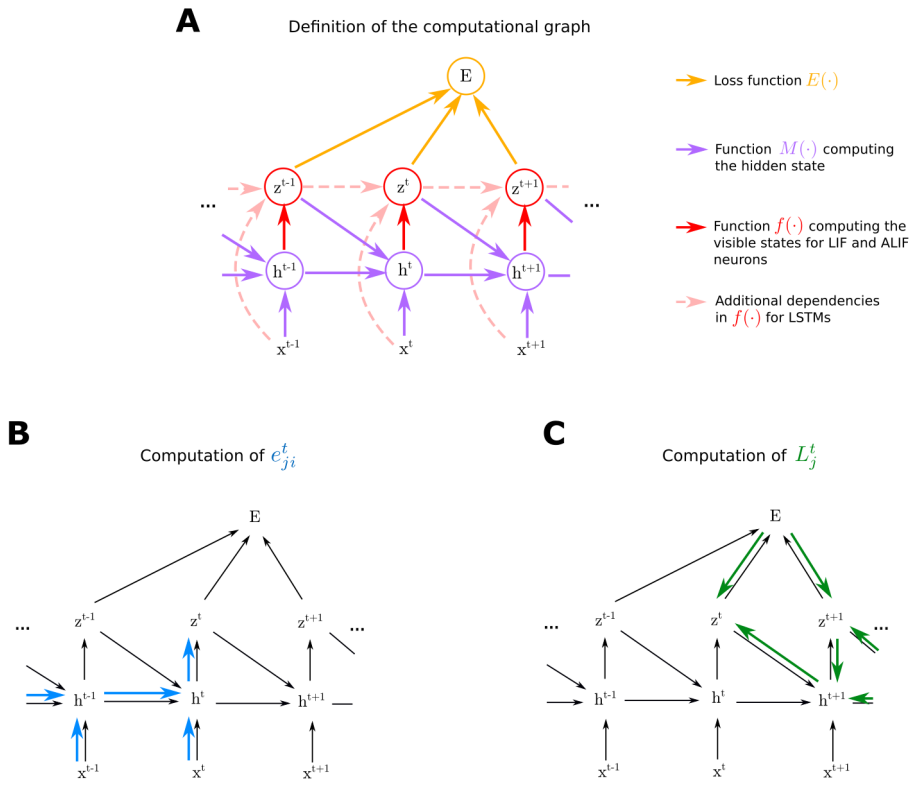

Figure 5: **Computational graph and gradient propagations A)** Assumed mathematical dependencies between hidden neuron states $\mathbf{h}_j^t$, neuron outputs $\mathbf{z}^t$, network inputs $\mathbf{x}^t$, and the loss function E through the mathematical functions $E(\cdot)$, $M(\cdot)$, $f(\cdot)$ are represented by coloured arrows. **B-C)** The gradient computation can be represented in similar graphs, where coloured arrows represent partial derivatives. **B)** Following equation (19), the derivatives involved in the computation of eligibility traces e_{ji}^t are shown in blue in the case where i is an input neuron. **C)** Unlike the eligibility traces, the ideal learning signals required to back-propagate gradients as represented here with green arrows.

Network output and loss functions. We assume that network outputs y_k^t are real-valued and produced by leaky output neurons (readouts), which are not recurrently connected:

$$y_k^t = \kappa y_k^{t-1} + \sum_j W_{kj}^{\text{out}} z_j^t + b_k^{\text{out}} , \tag{8}$$

where $\kappa \in [0, 1]$ defines the leak and b_k^{out} denotes the output bias. The leak factor κ is given for spiking neurons by $e^{-\delta t/\tau_{out}}$, where τ_{out} is the membrane time constant. Note that for non-spiking neural networks (such as for LSTM networks), temporal smoothing of the network observable state is not necessary. In this case, one can use $\kappa = 0$.

The loss function E quantifies the network performance. We assume that it depends only on the observable states $E(\mathbf{z}^1, \ldots, \mathbf{z}^T)$. For instance, for a regression problem we define E as the mean square error $E = \frac{1}{2}\sum_{t,k}(y_k^t - y_k^{*,t})^2$ between the network outputs y_k^t and target values $y_k^{*,t}$. For classification or RL tasks the loss function E has to be re-defined accordingly.

Conventions

Notation for derivatives. We distinguish the total derivative $\frac{dE}{d\mathbf{z}^t}(\mathbf{z}^1, \ldots, \mathbf{z}^T)$, which takes into account how E depends on $\mathbf{z}_t$ also indirectly through influence of $\mathbf{z}^t$ on the other variables $\mathbf{z}^{t+1}, \ldots, \mathbf{z}^T$, and the partial derivative $\frac{\partial E}{\partial \mathbf{z}^t}(\mathbf{z}^1, \ldots, \mathbf{z}^T)$ which quantifies only the direct dependence of E on $\mathbf{z}^t$.

Analogously $\frac{\partial M}{\partial h}$ denotes for $\mathbf{h}_j^t = M(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$, the partial derivative of M with respect to $\mathbf{h}$. It only quantifies the direct influence of $\mathbf{h}_j^t$ on $\mathbf{h}_j^{t-1}$ and it does not take into account the dependency of $\mathbf{h}_j^t$ on $\mathbf{h}_j^{t-1}$ via the observable states $\mathbf{z}^t$. To improve readability we also use the following abbreviations: $\frac{\partial \mathbf{h}_j^t}{\partial \mathbf{h}_j^{t-1}} \stackrel{\text{def}}{=} \frac{\partial M}{\partial h}(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$, $\frac{\partial \mathbf{h}_j^t}{\partial W_{ji}} \stackrel{\text{def}}{=} \frac{\partial M}{\partial W_{ji}}(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$, and $\frac{\partial z_j^t}{\partial \mathbf{h}_j^t} \stackrel{\text{def}}{=} \frac{\partial f}{\partial h}(\mathbf{h}_j^t, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$.

Notation for temporal filters. For ease of notation we use the operator $\mathcal{F}_\alpha$ to denote the low-pass filter such that, for any time series x_t:

$$\mathcal{F}_\alpha(x^t) = \alpha \mathcal{F}_\alpha(x^{t-1}) + x^t , \tag{9}$$

and $\mathcal{F}_\alpha(x^0) = x^0$. In the specific case of the time series z_j^t and e_{ji}^t, we simplify notation further and write $\bar{z}_j^t$ and $\bar{e}_{ji}^t$ for $\mathcal{F}_\alpha(z_j)^t$ and $\mathcal{F}_\kappa(e_{ji})^t$

Mathematical basis for *e-prop*

We provide here the proof of the fundamental equation (1) for *e-prop*

$$\frac{dE}{dW_{ji}} = \sum_t \frac{dE}{dz_j^t}\, e_{ji}^t . \tag{10}$$

This equation shows that the total derivative of the loss function E with respect to the synaptic weights $\boldsymbol{W}$ can be written as a product of learning signals L_j^t and eligibility traces e_{ji}^t for the "ideal" learning signal $L_j^t = \frac{dE}{dz_j}$. The eligibility traces are defined at the end of the proof below.

We start from a factorization of the loss gradient that arises in equation (12) of (Werbos, 1990) to describe *BPTT* in recurrent sigmoidal neural networks. Using our notation, this classical factorization of loss gradient can be rewritten as:

$$\frac{dE}{dW_{ji}} = \sum_{t'} \frac{dE}{d\mathbf{h}_j^{t'}} \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}} . \tag{11}$$

We now show how one can derive from this to the new factorization (10) of the loss gradient that underlies *e-prop*. $\frac{dE}{d\mathbf{h}_j^{t'}}$ can be expressed recursively as a function of the same derivative

at the next time step $\frac{dE}{d\mathbf{h}_j^{t'+1}}$ by applying the chain rule at the node $\mathbf{h}_j^t$ for $t = t'$ of the computational graph shown in Figure 5C:

$$\frac{dE}{d\mathbf{h}_j^{t'}} = \frac{dE}{dz_j^{t'}}\frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \frac{dE}{d\mathbf{h}_j^{t'+1}}\frac{\partial \mathbf{h}_j^{t'+1}}{\partial \mathbf{h}_j^{t'}} \tag{12}$$

$$= L_j^{t'}\frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \frac{dE}{d\mathbf{h}_j^{t'+1}}\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}, \tag{13}$$

where we defined the learning signal $L_j^{t'}$ as $\frac{dE}{dz_j^{t'}}$. The resulting recursive expansion ends at the last time step T of the computation of the RNN, i.e., $\frac{dE}{d\mathbf{h}_j^{T+1}} = 0$. If one substitutes the recursive formula (13) into the definition of the loss gradients (11), one gets:

$$\frac{dE}{dW_{ji}} = \sum_{t'}\left(L_j^{t'}\frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \frac{dE}{d\mathbf{h}_j^{t'+1}}\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}\right)\cdot\frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}} \tag{14}$$

$$= \sum_{t'}\left(L_j^{t'}\frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \left(L_j^{t'+1}\frac{\partial z_j^{t'+1}}{\partial \mathbf{h}_j^{t'+1}} + (\cdots)\frac{\partial \boldsymbol{h}_j^{t'+2}}{\partial \boldsymbol{h}_j^{t'+1}}\right)\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}\right)\cdot\frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}\,. \tag{15}$$

The following equation is the main equation for understanding the transformation from *BPTT* into *e-prop*. The key idea is to collect all terms which are multiplied with the learning signal L_j^t at a given time t. These are only terms that concern events in the computation of neuron j up to time t, and they do not depend on other future losses or variable values. We collect them into an eligibility trace e_{ji}^t for each neuron j and i, which can be computed locally in an online manner.

To this end, we write the term in parentheses in equation (15) into a second sum indexed by t and exchange the summation indices to pull out the learning signal L_j^t. This expresses the loss gradient of E as a sum of learning signals L_j^t multiplied by some factor indexed by ji, which we define as the eligibility trace $e_{ji}^t \in \mathbb{R}$ and eligibility vectors $\boldsymbol{\epsilon}_{ji}^t \in \mathbb{R}^d$, which have the same dimension as the hidden states $\mathbf{h}_{ji}^t$

$$\frac{dE}{dW_{ji}} = \sum_{t'}\sum_{t\geq t'} L_j^t\frac{\partial z_j^t}{\partial \mathbf{h}_j^t}\frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}}\cdots\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}\cdot\frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}} \tag{16}$$

$$= \sum_t L_j^t\frac{\partial z_j^t}{\partial \mathbf{h}_j^t}\underbrace{\sum_{t'\leq t}\frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}}\cdots\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}\cdot\frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}}_{\stackrel{\text{def}}{=}\boldsymbol{\epsilon}_{ji}^t}\,. \tag{17}$$

Here, we use the identity matrix for $\frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}}\cdots\frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}}$ if $t = t'$. After defining the eligibility vector $\boldsymbol{\epsilon}_{ji}^t$, we also define

$$e_{ji}^t \stackrel{\text{def}}{=} \frac{\partial z_j^t}{\partial \mathbf{h}_j^t}\cdot\boldsymbol{\epsilon}_{ji}^t\,, \tag{18}$$

so that equation (17) proves the factorization of *e-prop* in (1).

Eligibility traces

Online computation of eligibility traces. The eligibility vectors as defined in (17) can be computed recursively for efficiency and in order to avoid the back-propagation of signals through time:

$$\boldsymbol{\epsilon}_{ji}^t = \frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}}\cdot\boldsymbol{\epsilon}_{ji}^{t-1} + \frac{\partial \mathbf{h}_j^t}{\partial W_{ji}}\,, \tag{19}$$

where $\cdot$ denotes the dot product. The eligibility traces can be computed with their definition in equation (18).

Derivation of eligibility traces for concrete neuron models

The eligibility traces for LSTMs are provided in the supplementary materials. Below we provide the derivation of eligibility traces for spiking neurons.

Eligibility traces for LIF neurons. We compute the eligibility trace of a LIF neuron without adaptive threshold (equation (3)). Here the hidden state $\mathbf{h}_j^t$ consists just of the membrane potential v_j^t and we have $\frac{\partial \boldsymbol{h}_j^{t+1}}{\partial \boldsymbol{h}_j^t} = \frac{\partial v_j^{t+1}}{\partial v_j^t} = \alpha$ and $\frac{\partial v_j^t}{\partial W_{ji}} = z_i^{t-1}$ (for a derivation of the eligibility traces taking the reset into account we refer to section S1.2). Using these derivatives and equation (19), one obtains that the eligibility vector is the low-pass filtered pre-synaptic spike-train,

$$\boldsymbol{\epsilon}_{ji}^{t+1} = \mathcal{F}_\alpha(z_i^t) \stackrel{\text{def}}{=} \bar{z}_i^t \ . \tag{20}$$

and following equation (18), the eligibility trace is:

$$e_{ji}^{t+1} = \psi_j^{t+1} \bar{z}_i^t \ . \tag{21}$$

For LIF neurons as well as for ALIF neurons in the following section the derivation applies to the input connections by substituting the network spikes z_i^{t-1} by the input spikes x_i^t (the time index switches from $t-1$ to t because the hidden state $\mathbf{h}_j^t = M(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \boldsymbol{W})$ is defined as a function of the input at time t but the preceding recurrent activity). For simplicity we have focused on the case where transmission delays between neurons in the RSNN are just 1ms. If one uses more realistic length of delays d, this $-d$ appears in equations (21)–(23) instead of -1 as the most relevant time point for pre-synaptic firing (see Section S1.3). This moves resulting synaptic plasticity rules closer to experimentally observed forms of STDP.

Eligibility traces for ALIF neurons. The hidden state of an ALIF neuron $\mathbf{h}_j^t = [v_j^t, a_j^t]$ is a two dimensional vector to capture the state of the adaptive threshold a_j^t besides the membrane potential v_j^t. Hence a two dimensional eligibility vector $\boldsymbol{\epsilon}_{ji}^t \stackrel{\text{def}}{=} [\epsilon_{ji,v}^t, \epsilon_{ji,a}^t]$ is associated with each weight, and the matrix $\frac{\partial \boldsymbol{h}_j^{t+1}}{\partial \boldsymbol{h}_j^t}$ is a 2×2 matrix. The derivatives $\frac{\partial a_j^{t+1}}{\partial a_j^t}$ and $\frac{\partial a_j^{t+1}}{\partial v_j^t}$ capture the dynamics of the adaptive threshold. Hence to derive the computation of eligibility traces we substitute the spike z_j in equation (7) by its definition given in equation (6). With this convention one finds that the diagonal of the matrix $\frac{\partial \boldsymbol{h}_j^{t+1}}{\partial \boldsymbol{h}_j^t}$ is formed by the terms $\frac{\partial v_j^{t+1}}{\partial v_j^t} = \alpha$ and $\frac{\partial a_j^{t+1}}{\partial a_j^t} = \rho - \psi_j^t \beta$. Above and below the diagonal, one finds respectively $\frac{\partial v_j^{t+1}}{\partial a_j^t} = 0$, $\frac{\partial a_j^{t+1}}{\partial v_j^t} = \psi_j^t$. One can finally compute the eligibility traces using equation (18). The component of the eligibility vector associated with the membrane potential remains the same as in the LIF case and only depends on the presynaptic neuron: $\epsilon_{ji,v}^t = \bar{z}_i^{t-1}$. For the component associated with the adaptive threshold we find the following recursive update:

$$\epsilon_{ji,a}^{t+1} \quad = \quad \psi_j^t \bar{z}_i^{t-1} + (\rho - \psi_j^t \beta) \epsilon_{ji,a}^t \ , \tag{22}$$

and this results in an eligibility trace of the form:

$$e_{ji}^t \quad = \quad \psi_j^t \left(\bar{z}_i^{t-1} - \beta \epsilon_{ji,a}^t \right). \tag{23}$$

Recall that the constant $\rho = \exp(-\frac{\delta t}{\tau_a})$ arises from the adaptation time constant τ_a, which typically lies in the range of hundreds of milliseconds to a few seconds in our experiments, yielding values of ρ between 0.995 and 0.9995. The constant β is typically of the order of 0.07 in our experiments.

To provide a more interpretable form of eligibility trace that fits into the standard form of local terms considered in 3-factor learning rules (Gerstner et al., 2018), one may drop the term $-\psi_j^t \beta$ in equation (22). This approximation $\widehat{\epsilon}_{ji,a}^t$ of equation (22) becomes an exponential trace of the post-pre pairings accumulated within a time window as large as the adaptation adaptation time constant:

$$\widehat{\epsilon}_{ji,a}^{t+1} \quad = \quad \mathcal{F}_\rho \left(\psi_j^t \bar{z}_i^{t-1} \right) \ . \tag{24}$$

The eligibility traces are computed with equation (22) in most experiments but the performance obtained with *symmetric e-prop* and this simplification were indistinguishable on the evidence accumulation task of Fig. 3.

Synaptic plasticity rules resulting from *e-prop*

An exact computation of the ideal learning signal $\frac{dE}{dz_j^t}$ in equation (1) requires to back-propagate gradients through time (see Fig. 5C). To compute the loss gradients with *e-prop* we replace it with the partial derivative $\frac{\partial E}{\partial z_j^t}$ which can be computed online. Implementing the weight updates with gradient descent and learning rate η, all the following plasticity rules are derived from the formula

$$\Delta W_{ji}^{\text{rec}} = -\eta \sum_t \frac{\partial E}{\partial z_j^t} e_{ji}^t \,. \tag{25}$$

Note that the weight updates derived for the recurrent weights W_{ji}^{rec} also applies to the inputs weights W_{ji}^{in}. For the output weights and biases the derivation does not rely on the theory of *e-prop*, and the weight updates can be found in the Section S3.1.

Case of regression tasks. In the case of a regression problem with targets $y_k^{*,t}$ and outputs y_k^t defined in equation (8), we define the loss function $E = \frac{1}{2}\sum_{t,k}(y_k^t - y_k^{*,t})^2$ which results in a partial derivative of the form $\frac{\partial E}{\partial z_j^t} = \sum_k W_{kj}^{\text{out}} \sum_{t' \geq t}(y_k^{t'} - y_k^{*,t'})\kappa^{t'-t}$. This seemingly provides an obstacle for online learning, because the partial derivative is a weighted sum over future errors. But this problem can be resolved as one interchange two sum indices in the expression of the weight updates (see section S3.1). It results that the sum over future events transforms into a low-pass filtering of the eligibility traces $\bar{e}_{ji}^t = \mathcal{F}_\kappa(e_{ji}^t)$, and the resulting weight update can be written as

$$\Delta W_{ji}^{\text{rec}} = -\eta \sum_t \underbrace{\Big(\sum_k B_{jk}(y_k^t - y_k^{*,t})\Big)}_{=L_j^t} \bar{e}_{ji}^t \,. \tag{26}$$

Here, B_{jk} denote broadcast weights in analogy to (Lillicrap et al., 2016), where we note that $B_{jk} = W_{kj}^{\text{out}}$ as the ideal values.

Case of classification tasks. We assume that K target categories are provided in the form of a one-hot encoded vector $\boldsymbol{\pi}^{*,t}$ with K dimensions. We define the probability for class k predicted by the network as $\pi_k^t = \text{softmax}_k(y_1^t, \ldots, y_K^t) = \exp(y_k^t)/\sum_{k'}\exp(y_{k'}^t)$, and the loss function for classification tasks as the cross-entropy error $E = -\sum_{t,k}\pi_k^{*,t}\log\pi_k^t$. The plasticity rule resulting from *e-prop* reads (see derivation in Section S3.1):

$$\Delta W_{ji}^{\text{rec}} = -\eta \sum_t \underbrace{\Big(\sum_k B_{jk}(\pi_k^t - \pi_k^{*,t})\Big)}_{=L_j^t} \bar{e}_{ji}^t \,. \tag{27}$$

Reward-based e-prop: application of *e-prop* to policy gradient RL

For reinforcement learning, the network interacts with an external environment. Based on the observations $\boldsymbol{x}^t$ that are perceived, the network has to commit to actions $\boldsymbol{a}^{t_0}, \cdots, \boldsymbol{a}^{t_n}, \cdots$ at certain decision times $t_0, \cdots, t_n, \cdots$. Each action $\boldsymbol{a}^{t_n}$ is sampled from a probability distribution $\pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})$ which is also referred to as the policy of the RL agent. The policy is defined as function of the network output $\boldsymbol{y}^{t_n}$, and is chosen here to be a vector of Gaussians with means $\boldsymbol{y}^t$ and variance σ^2 (see section S5.1 for discrete actions). At any time t the environment can provide a positive or negative reward r^t.

The goal of reinforcement learning is to maximize the expected sum of discounted rewards. That is, we want to maximize the expected return at time $t = 0$ $\mathbb{E}[R^0]$, where the return at time t is defined as $R^t = \sum_{t' \geq t}\gamma^{t'-t}r^{t'}$ with $\gamma \leq 1$ and the expectation is taken over the

agent actions $\boldsymbol{a}^t$, the rewards r^t and the observation from the environment $\boldsymbol{x}^t$. We approach this optimization problem using the actor-critic variant of the policy gradient algorithm which applies gradient ascent to maximize $\mathbb{E}[R^0]$. The form of the estimated gradient relies on a corollary of the policy gradient theorem shown in section 13.3 in (Sutton and Barto, 2018): the gradient $\frac{d\mathbb{E}\left[R^0\right]}{dW_{ji}}$ is proportional to $\mathbb{E}\left[\sum_{t_n} R^{t_n} \frac{d \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})}{dW_{ji}}\right]$ which is easier to compute because the expectation can be estimated by an average over one or many trials. Following this strategy, we define the per-trial loss function E_π as a function of the sequences actions $\boldsymbol{a}^{t_0}, \cdots, \boldsymbol{a}^{t_n}, \cdots$ and rewards $r^0, \cdots, r^T$ sampled during this trial:

$$E_\pi(\boldsymbol{z}^0, \cdots, \boldsymbol{z}^T, \boldsymbol{a}^{t_0}, \cdots \boldsymbol{a}^{t_n}, \cdots, r^0, \cdots, r^T) \quad \stackrel{\text{def}}{=} \quad -\sum_n R^{t_n} \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n}) \ , \tag{28}$$

and it results from the policy gradient theorem (Sutton and Barto, 2018) that:

$$\frac{d\mathbb{E}\left[R^0\right]}{dW_{ji}} \quad \propto \quad \mathbb{E}\left[\sum_{t_n} R^{t_n} \frac{d \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})}{dW_{ji}}\right] = -\mathbb{E}\left[\frac{dE_\pi}{dW_{ji}}\right] \ . \tag{29}$$

Intuitively, given a trial with high rewards, policy gradient changes the network output $\boldsymbol{y}$ to increase the probability of the actions $\boldsymbol{a}^{t_n}$ that occurred. Note in particular that $\boldsymbol{a}^{t_n}$ and r^{t_n} are samples of stochastic variables and their values do not depend on the parameters W_{ji} (only the probabilities $\pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})$ depend on the parameters). Hence, E_π can be viewed as a function of the observable states $\boldsymbol{z}^0, \cdots, \boldsymbol{z}^T$ from the view point of the optimization algorithm and E_π is a well defined loss. In practice, the gradient $\frac{dE_\pi}{dW_{ji}}$ is known to have high variance and the efficiency of the learning algorithm can be improved using the actor-critic variant of the policy gradient algorithm. It involves the policy π (the actor) and an additional output neuron V^t which predicts the value function $\mathbb{E}[R^t]$ (the critic). The actor and the critic are learnt simultaneously by defining the loss function

$$E = E_\pi + c_V E_V \ , \tag{30}$$

where $E_\pi = -\sum_n R^{t_n} \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})$ measures the performance of the stochastic policy π, and $E_V = \sum_t \frac{1}{2}(R^t - V^t)^2$ measures the accuracy of V^t. Using that V^t is independent of the action $\boldsymbol{a}^t$ one can show that $0 = \mathbb{E}\left[V^{t_n} \frac{d \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})}{dW_{ji}}\right]$, and use that to define an estimator $\widehat{\frac{dE}{dW_{ji}}}$ of the loss gradient with reduced variance

$$-\frac{d\mathbb{E}\left[R^0\right]}{dW_{ji}} + c_V \frac{d\mathbb{E}\left[E_V\right]}{dW_{ji}} \quad \propto \quad \mathbb{E}\left[\frac{dE}{dW_{ji}}\right] \tag{31}$$

$$= \quad \mathbb{E}\Bigg[\underbrace{-\sum_{t_n}(R^{t_n} - V^{t_n}) \frac{d \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})}{dW_{ji}} + c_V \frac{dE_v}{dW_{ji}}}_{\stackrel{\text{def}}{=} \widehat{\frac{dE}{dW_{ji}}}}\Bigg] \ . \tag{32}$$

Until now this derivation follows the classical definition of the actor-critic variant of policy gradient, and the gradient $\widehat{\frac{dE}{dW_{ji}}}$ has be computed with BPTT. To derive *reward-based e-prop* we follow instead the generic online approximation of *e-prop* as in equation (25) and approximate $\widehat{\frac{dE}{dW_{ji}}}$ as $\widehat{\frac{dE}{dz_j}} e^t_{ji}$ with

$$\widehat{\frac{\partial E}{\partial z^t_j}} \quad = \quad -\sum_n (R^{t_n} - V^{t_n}) \frac{\partial \log \pi(\boldsymbol{a}^{t_n}|\boldsymbol{y}^{t_n})}{\partial z^t_j} + c_V \frac{\partial E_V}{\partial z^t_j} \ . \tag{33}$$

We derive below the resulting synaptic plasticity rule for the case of multiple continuous actions as needed to solve the task of Fig. 4. For the case of a single discrete action as used in Fig. S4 we refer to section S5.1.

Case of continuous actions. This task is more difficult when there is a delay between the action and the reward or, even harder, when a sequence of many actions lead together to a delayed reward. There the loss function E cannot be computed online because the evaluation of R^{t_n} requires knowledge of future rewards. To overcome this, we introduce temporal difference errors $\delta^t = r^t + \gamma V^{t+1} - V^t$ (see Fig. S3), and use the equivalence between the forward and backward view in reinforcement learning (Sutton and Barto, 2018) to arrive at the following synaptic plasticity rules for a general actor-critic algorithm with *e-prop* (see Section S5.1):

$$\Delta W_{ji}^{\text{rec}} = -\eta \sum_t \delta^t \mathcal{F}_\gamma\left(L_j^t\ \bar{e}_{ji}^t\right) \quad \text{for} \tag{34}$$

$$L_j^t = -c_V B_j^V + \sum_k B_{jk}^a \frac{y_k^t - a_k^t}{\sigma^2}\ , \tag{35}$$

where we define the term $y_k^t - a_k^t$ to have value zero when no action is taken at time t. The combination of reward prediction error and neuron-specific learning signal was also used in a plasticity rule for feedforward networks inspired by neuroscience (Roelfsema and Holtmaat, 2018), here it arises from the approximation of *BPTT* by *e-prop* in RSNNs solving RL problems. Note that the filtering $\mathcal{F}_\gamma$ requires an additional eligibility trace per synapse. This arises from the temporal difference learning in RL(Sutton and Barto, 2018). It depends on the learning signal and does not have the same function as the eligibility trace e_{ji}^t.

Acknowledgments

This research/project was supported by the Human Brain Project (Grand Agreement number 785907) and the SYNCH project (Grand Agreement number 824162) of the European Union. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research. Computations were carried out on the Human Brain Project PCP Pilot Systems at the Juelich Supercomputing Centre, which received co-funding from the European Union (Grand Agreement number 604102) and on the Vienna Scientific Cluster (VSC).

We thank Thomas Bohnstingl, Wulfram Gerstner, Christopher Harvey, Martin Vinck, Jason MacLean, Adam Santoro, Christopher Summerfield, and Yuqing Zhu for helpful comments on an earlier version of the manuscript.

Authors contributions GB, FS, AS and WM conceived the work, GB, FS, AS, EH and DS carried out experiments and all authors contributed to the writing of the paper.

References

Allen Institute: Cell Types Database (2018). © 2018 Allen Institute for Brain Science. Allen Cell Types Database, cell feature search. Available from: celltypes.brain-map.org/data.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. In *NeurIPS 32*.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv:1901.09049 [cs]*. arXiv: 1901.09049.

Cassenaer, S. and Laurent, G. (2012). Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature*, 482(7383):47.

Clopath, C., Büsing, L., Vasilaki, E., and Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neuroscience*, 13(3):344–52.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99.

Engelhard, B., Finkelstein, J., Cox, J., Fleming, W., Jang, H. J., Ornelas, S., Koay, S. A., Thiberge, S. Y., Daw, N. D., Tank, D. W., et al. (2019). Specialized coding of sensory, motor and cognitive variables in vta dopamine neurons. *Nature*, page 1.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., di Nolfo, C., Datta, P., Amir, A., Taba, B., Flickner, M. D., and Modha, D. S. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *PNAS*, 113(41):11441–11446.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1993). DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93.

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. (2018). Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 12.

Gouwens, N. W., Berg, J., Feng, D., Sorensen, S. A., Zeng, H., Hawrylycz, M. J., Koch, C., and Arkhipov, A. (2018). Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature communications*, 9(1):710.

Graves, A., Mohamed, A.-R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE TNNLS*, 28(10):2222–2232.

Huh, D. and Sejnowski, T. J. (2018). Gradient descent for spiking neural networks. In *NeurIPS*, pages 1433–1443.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. (2016). Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*.

Kappel, D., Legenstein, R., Habenschuss, S., Hsieh, M., and Maass, W. (2018). A dynamic connectome supports the emergence of stable computational function of neural circuits through reward-based learning. *eNeuro*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276.

Lillicrap, T. P. and Santoro, A. (2019). Backpropagation through time and the brain. *Current Opinion in Neurobiology*, 55:82–89.

MacLean, S. J., Hassall, C. D., Ishigami, Y., Krigolson, O. E., and Eskes, G. A. (2015). Using brain potentials to understand prism adaptation: the error-related negativity and the p300. *Frontiers in human neuroscience*, 9:335.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937.

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

Mujika, A., Meier, F., and Steger, A. (2018). Approximating real-time recurrent learning with random kronecker factors. In *NeurIPS*, pages 6594–6603.

Murray, J. M. (2019). Local online learning in recurrent networks with random feedback. *eLife*, 8:e43299.

Nicola, W. and Clopath, C. (2017). Supervised learning in spiking neural networks with force training. *Nature Communications*, 8(1):2208.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In *NIPS*, pages 1037–1045.

Pozzorini, C., Mensi, S., Hagens, O., Naud, R., Koch, C., and Gerstner, W. (2015). Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS computational biology*, 11(6):e1004275.

Roelfsema, P. R. and Holtmaat, A. (2018). Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180.

Roeper, J. (2013). Dissecting the diversity of midbrain dopamine neurons. *Trends in neurosciences*, 36(6):336–342.

Roth, C., Kanitscheider, I., and Fiete, I. (2019). Kernel rnn learning (kernl). *ICLR*.

Runyan, C. A., Piasini, E., Panzeri, S., and Harvey, C. D. (2017). Distinct timescales of population coding across cortex. *Nature*, 548:92–96.

Sajad, A., Godlove, D. C., and Schall, J. D. (2019). Cortical microcircuitry of performance monitoring. *Nature Neuroscience*, 22(2):265.

Sanhueza, M. and Lisman, J. (2013). The camkii/nmdar complex as a molecular memory. *Molecular brain*, 6(1):10.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

Tallec, C. and Ollivier, Y. (2018). Unbiased online recurrent optimization. *ICLR*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Yagishita, S., Hayashi-Takagi, A., Ellis-Davies, G. C., Urakubo, H., Ishii, S., and Kasai, H. (2014). A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620.

Zenke, F. and Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541.