Dynamic Action Inference with Recurrent Spiking Neural Networks[⋆]

Manuel Traub[1] [0000−0003−0897−1701], Martin V. Butz[1] [0000−0002−8120−8537], Robert Legenstein[3] [0000−0002−8724−5507], and Sebastian Otte[1] [0000−0002−0305−0463]

[1] Neuro-Cognitive Modeling, Computer Science Department, University of Tübingen, Sand 14, 72076 Tübingen, Germany
{manuel.traub,martin.butz,sebastian.otte}@uni-tuebingen.de

[2] Faculty of Computer Science and Biomedical Engineering, Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria
robert.legenstein@igi.tugraz.at

Abstract. In this paper, we demonstrate that goal-directed behavior unfolds in recurrent spiking neural networks (RSNNs) when intentions are projected onto continuously progressing spike dynamics encoding the recent history of an agent's state. The projections, which can either be realized via backpropagation through time (BPTT) over a certain time window or even directly and temporally local in an online fashion using a biologically inspired inference rule. In contrast to previous studies that use, for instance, LSTM-like models, our approach is biologically more plausible as it fully relies on spike-based processing of sensorimotor experiences. Specifically, we show that precise control of a flying vehicle in a 3D environment is possible. Moreover, we show that more complex mental traces of foresighted movement imagination unfold that effectively help to circumvent learned obstacles.

Keywords: Recurrent spiking neural networks · active inference · temporal gradients

1 Introduction

Recent progress in the field of reinforcement learning (RL) has gained huge attention for learning to play various video games, most notably from the atari console, and more recently even complex strategic online multiplayer games [10, 9, 11]. While these results are impressive, the used training algorithms like proximal policy optimization [14] or similar model free RL approaches require tremendous amounts of learning time (typically thousands of years of simulated learning episodes). This is neither very efficient nor biologically plausible. Moreover, these

[⋆] We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Manuel Traub. Martin Butz is part of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645.

approaches do not foster a deeper understanding of the task at hand. In contrast, cognitive science research, and in particular theories of predictive coding and active inference, suggest that our brain learns a predictive understanding of the world [4, 6, 7].

Previous research already explored how the active inference principle can be implemented in predictive recurrent neural networks (RNNs) to realize the emergence of goal-directed behavior, planning, and environmental interaction [13, 5, 3, 12]. This is achieved as follows: First, a temporal forward model of a system of interest is learned, typically with an LSTM-like RNN [8], from temporally causal streams of available sensorimotor information. Second, motor signals are inferred on-the-fly by backpropagating intentions (such as desired system states) in form of prediction error-induced gradient signal through continuously adapting unrolled imaginations of the anticipated future system dynamics. While such RNNs are a rough simplification of real biological neural networks, they can be trained effectively by using backpropagation through time (BPTT). Recent progress in research on more biologically plausible spiking neural networks (SNNs), however, enables the training of SNNs end-to-end with a biologically more plausible learning rule, called e-prop, which comes close to the performance of BPTT [2].

Of particular interest for this paper is a variant of recurrent SNNs (RSNNs), which is referred to as a *long short-term spiking neural network* (LSNN) [1]. An LSNN consists of two types of spiking neurons: common leaky integrate and fire (LIF) neurons and adaptive LIF (ALIF) neurons. With an adaptive threshold that effectively regulates the firing rates of the latter, they can act as data and gradient highways fostering long-term influences on the network dynamics. In fact, LSNNs unfold impressive learning capabilities which are approaching, and in some cases even surpassing, the performance of LSTMs [1, 2, 16].

For our purposes, the e-prop [2] rule is also highly relevant, as it is ideal for continuous online learning scenarios. E-prop is based on a factorization of the gradient computation into an error-depending learning signal and an activation-depending eligibility trace. While the former is a time-local approximation of the full error signal, the latter can be computed forward in time along the regular forward pass of the network. Thus, e-prop does not require temporally backwards error signal propagation, as standard BPTT does.

In this paper, we bring together active inference-inspired behavior generation and biologically plausible SNNs. Specifically, we demonstrate that goal-directed, anticipatory behavior can emerge from projecting intentions through continuously unfolding spike dynamics onto motor inputs.

2 Action Inference in Recurrent Spiking Neural Networks

Establishing adaptive, goal-directed behavior in a continuous, dynamic control scenario with RSNNs involves essentially two aspects. First, a temporal forward model, that is, a neural approximation of the dynamical system of interest, is learned. Second, action sequences are inferred by means of a goal-based loss function.

2.1 Dynamical System Formulation

We assume a controllable discrete-time dynamical system with time-dependent system states. We follow the formalization from [12, 13] in this section. We furthermore assume that the states are basically separated into perceivable states $\boldsymbol{\sigma}^t \in \mathbb{R}^n$ and unobservable (hidden) states $\boldsymbol{\omega}^t \in \mathbb{R}^m$. The system's dynamics can be influenced via k control commands denoted by $\mathbf{x}^t \in \mathbb{R}^k$. The next system state $(\boldsymbol{\sigma}^{t+1}, \boldsymbol{\omega}^{t+1})$ is determined by a (possibly unknown) state transition function

$$(\boldsymbol{\sigma}^t, \boldsymbol{\omega}^t, \mathbf{x}^t) \xmapsto{\Phi} (\boldsymbol{\sigma}^{t+1}, \boldsymbol{\omega}^{t+1}), \tag{1}$$

which models the forward dynamics of the system. As this process unfolds recursively over time, the next system state depends not only on the current control inputs, but also, in principle, on the entire state history. It is the learning task of the neural forward model to approximate Φ given the current state $\boldsymbol{\sigma}^t$ and current control commands $\mathbf{x}^t$, as well as an internal representation aggregated from the previous system state history $\{\boldsymbol{\sigma}^1, \boldsymbol{\sigma}^2, \dots, \boldsymbol{\sigma}^{t-1}\}$ and corresponding motor commands $\{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{t-1}\}$. Since the neural forward model cannot directly access the unobservable states $\boldsymbol{\omega}^t$ of the system, we will see that our system learns to approximate the missing computational components sufficiently well and thus becomes able to infer goal-directed actions by means of model-predictive control.

2.2 LSNN Forward Model

For learning Φ, we use an LSNN architecture with only one single recurrent hidden layer. We directly inject real valued inputs without any explicit spike encoding, which worked best in preliminary experiments. The hidden layer is composed of both LIF and ALIF neurons in an one-to-one ratio. It should be noted that the original formulation of LSNNs comes from [1]. In terms of notation details, however, the equations in this paper are aligned with [15] (which additionally offers details on the derivation of BPTT for SNNs with LIF and ALIF neurons). We calculate the activation of the hidden neurons as presented in the following:

LIF activation

$$v_j^t = \alpha v_j^{t-1} + \sum_i w_{i,j}^{in} x_i^t + \sum_{j'} w_{j',j}^{rec} z_{j'}^{t-1} - z_j^{t-1} v_{thr} \tag{2}$$

$$z_j^t = \Theta\left(v_j^t - v_{thr}^t\right) \tag{3}$$

ALIF activation

$$v_j^t = \alpha v_j^{t-1} + \sum_i w_{i,j}^{in} x_i^t + \sum_{j'} w_{j',j}^{rec} z_{j'}^{t-1} - z_j^{t-1} v_{j,thr}^{t-1} \tag{4}$$

$$a_j^t = \rho a_j^{t-1} + z_j^{t-1} \tag{5}$$

$$v_{j,thr}^t = v_{thr} + \zeta a_j^t \tag{6}$$

$$z_j^t = \Theta\left(v_j^t - v_{j,thr}^t\right) \tag{7}$$

The voltage v_j of a neuron is modeled as an exponentially decaying sum over the weighted inputs (with the leakage rate α), which gets reset after a spike in z_j occurs by subtracting the value of the spike threshold. For LIF neurons the threshold v_{thr} is constant. For ALIF neurons an adaptive threshold $v_{j,thr}$ is used, which depends on the individual spiking activity of the neurons. Due to this behavior—and its formal structure in particular—ALIF neurons act as data (as well as gradient) highways able to bridge even large temporal gaps, which is essential when confronted with long data sequences [1]. For both LIF and ALIF neurons, the spike output is computed using the non-differentiable Heaviside function denoted by Θ.

The output layer of the network consists of leaky readout neurons, which are modeled according to Equation (2) without the reset term. The entire network is trained using either BPTT or e-prop using the mean squared error (MSE) loss.

2.3 Motor Inference Principle

The inference process uses a sufficiently trained forward model in order to infer control commands by means of BPTT or an e-prop inspired inference algorithm. For this purpose we define a prospective temporal horizon, which determines over how many time steps we will unroll our future projections. During the future projections the network computes in closed loop, that is, it feeds itself with its own predictions of the future development of the perceivable system states. For this, we first randomly initialize a vector of future control commands with the length of the chosen temporal horizon. The other inputs are then based on the network's predictions.

Using this setup, which is visualized in Figure 1, the network calculates for each imaginary future time step a prediction of the perceivable system state given the used motor commands. By using desired targets for these predicted states, at each or only at the last imagined time step, we can backpropagate a prediction error-based gradient signal through the unrolled network (through time). When we map this gradient onto the control inputs, we end up with an input gradient with respect to the discrepancy between the predicted system states and the desired states. By using a gradient descent technique and by repeating the described procedure, we can adaptively optimize the sequence of control inputs effectively pushing the system towards the desired state. Note that we always start from the saved latent state which was produced in the last time step.

In the following, we formally derive the input gradient for the used LSNN network model. Let us refer to the state of a particular neuron j at time step t as $\mathbf{s}_j^t$. For LIF neurons this neuron state is one-dimensional and only contains the voltage v_j^t, whereas for ALIF neurons it is two-dimensional and contains the voltage v_j^t as well as the threshold adaption value a_j^t:

$$\mathbf{s}_{j,LIF}^t \overset{\text{def}}{=} v_j^t \tag{8}$$

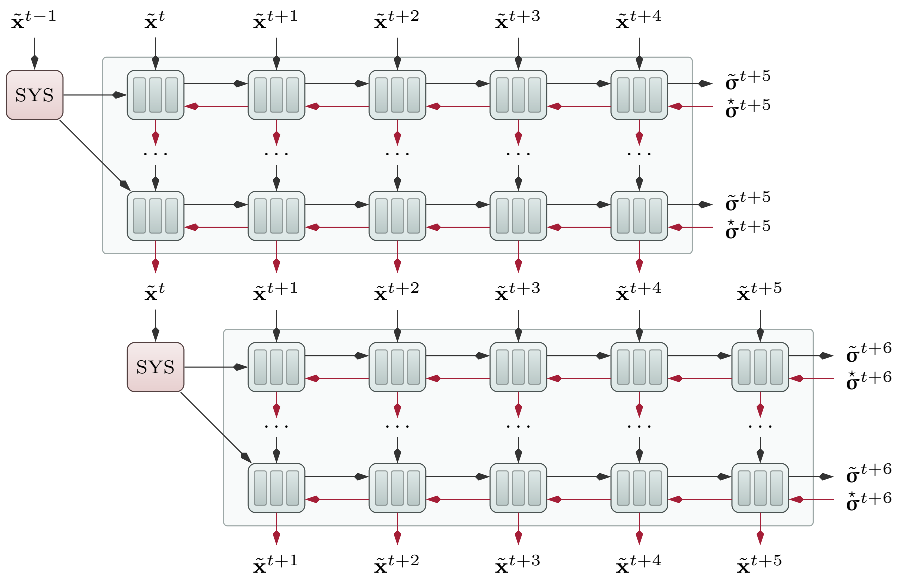
$\tilde{\mathbf{x}}^{t-1}$
$\tilde{\mathbf{x}}^{t}$
$\tilde{\mathbf{x}}^{t+1}$
$\tilde{\mathbf{x}}^{t+2}$
$\tilde{\mathbf{x}}^{t+3}$
$\tilde{\mathbf{x}}^{t+4}$
$\tilde{\mathbf{x}}^{t+5}$
SYS
$\tilde{\boldsymbol{\sigma}}^{t+5}$
$\overset{*}{\boldsymbol{\sigma}}^{t+5}$
$\tilde{\boldsymbol{\sigma}}^{t+6}$
$\overset{*}{\boldsymbol{\sigma}}^{t+6}$

Fig. 1. Illustration of the action inference procedure with LSNNs: Based on the current state $\boldsymbol{\sigma}^t$ of the observed system (SYS), the network imagines the future development of the states $(\tilde{\boldsymbol{\sigma}}^{t+1}, \tilde{\boldsymbol{\sigma}}^{t+2}, \ldots)$ given an initial motor sequence $(\tilde{\mathbf{x}}^t, \tilde{\mathbf{x}}^{t+1}, \ldots)$. During this unrolling, the LSNN is fed with its own predictions. For the LSNN, however, each time step consists of multiple sub time steps (as indicated by the small rectangles within the LSNN nodes). In the last imagined time step, an intention $(\overset{*}{\boldsymbol{\sigma}}^{t+5})$ is injected via a prediction loss function, backpropagated through the unrolled imagined state sequence, and eventually projected onto the control inputs. Using a gradient descent technique and repeating this procedure multiple times within the current time step leads to an adaption of the motor inputs. $\tilde{\mathbf{x}}^{\mathbf{t}}$ is executed to transition to the next time step, where the same procedure starts again.

$$\mathbf{s}^t_{j,ALIF} \overset{\text{def}}{=} \begin{bmatrix} v^t_j & a^t_j \end{bmatrix}^\top \tag{9}$$

The full gradient calculation requires all components within the network's computation chain to be differentiable. As mentioned earlier, however, the Heaviside function does not fulfill this requirement. To overcome this problem, Bellec et al. [2] introduced a pseudo-derivative h^t_j in place for the non-existing derivative of the threshold function:

$$\frac{\partial z^t_j}{\partial \mathbf{s}^t_{j,LIF}} \overset{\text{def}}{=} h^t_{j,LIF} = \lambda \ \max\left(0, 1 - \left|\frac{v^t_j - v_{thr}}{v_{thr}}\right|\right) \tag{10}$$

$$\frac{\partial z^t_j}{\partial \mathbf{s}^t_{j,ALIF}} \overset{\text{def}}{=} \begin{bmatrix} 1 \ , -\zeta \end{bmatrix}^\top h^t_{j,ALIF} = \begin{bmatrix} 1 \ , -\zeta \end{bmatrix}^\top \lambda \ \max\left(0, 1 - \left|\frac{v^t_j - v^t_{j,thr}}{v_{thr}}\right|\right) \tag{11}$$

where the dampening factor λ scales the steepness of the linear segments. With help of this pseudo-derivative we can calculate the partial derivative of the loss with respect to $\mathbf{s}_j^t$:

$$\boldsymbol{\delta}_j^t \overset{\text{def}}{=} \frac{\partial E}{\partial \mathbf{s}_j^t} \tag{12}$$

by applying the chain rule, which is done in the following for LIF and ALIF neurons.

LIF state gradient

$$\boldsymbol{\delta}_j^t = \frac{\partial E}{\partial z_j^t}\frac{\partial z_j^t}{\partial v_j^t} + \frac{\partial E}{\partial v_j^{t+1}}\frac{\partial v_j^{t+1}}{\partial v_j^t} = \frac{\partial E}{\partial z_j^t} h_j^t + \boldsymbol{\delta}_j^{t+1}\alpha \tag{13}$$

ALIF state gradient

$$\boldsymbol{\delta}_j^t = \frac{\partial E}{\partial z_j^t}\frac{\partial z_j^t}{\partial \mathbf{s}_j^t} + \frac{\partial E}{\partial \mathbf{s}_j^{t+1}}\frac{\partial \mathbf{s}_j^{t+1}}{\partial \mathbf{s}_j^t} = \frac{\partial E}{\partial z_j^t}\begin{bmatrix} h_j^t \\ -h_j^t\zeta \end{bmatrix} + \begin{bmatrix} \alpha\delta_{j,v}^{t+1} \\ \rho\delta_{j,a}^{t+1} \end{bmatrix} \tag{14}$$

Using these delta terms we can finally derive the input gradient:

$$\frac{\partial E}{\partial x_i^t} = \sum_j \frac{\partial \mathbf{s}_j^t}{\partial x_i^t}\frac{\partial E}{\partial \mathbf{s}_j^t} = \sum_j w_{i,j}^{in}\boldsymbol{\delta}_j^t \tag{15}$$

As a biologically more plausible and significantly more efficient error-based inference rule, we propose an alternative for BPTT, which is inspired by e-prop [2] and can be computed forward in time. Therefore, we expand the delta term part of the input gradient and discard errors from the future in order to be able to compute the error forward in time. We do this in three different styles:

Symmetric e-prop

$$\frac{\partial E}{\partial x_i^t} = \sum_j w_{i,j}^{in}\boldsymbol{\delta}_j^t \approx \sum_j w_{i,j}^{in}\frac{\partial E}{\partial z_j^t}\frac{\partial z_j^t}{\partial \mathbf{s}_j^t} = \sum_j w_{i,j}^{in} h_j^t \left(\sum_k \boldsymbol{\delta}_k^t w_{j,k}^{out}\right) \tag{16}$$

Sign-based e-prop

$$\frac{\partial E}{\partial x_i^t} \approx \sum_j \operatorname{sgn}(w_{i,j}^{in}) h_j^t \left(\sum_k \boldsymbol{\delta}_k^t \operatorname{sgn}(w_{j,k}^{out})\right) \tag{17}$$

Random feedback e-prop

$$\frac{\partial E}{\partial x_i^t} \approx \sum_j \operatorname{sgn}(w_{i,j}^{in}) h_j^t \left(\sum_k \boldsymbol{\delta}_k^t b_{j,k}^{out}\right) \tag{18}$$

For the symmetric case we calculate the input error of the network only as one single backward-pass through the network, where $\boldsymbol{\delta}_k^t$ is the output error of the output neuron k from the current sub time step. In the two other cases we either use only the sign of the weights or the random feedback weights ($b_{j,k}^{out}$) which were used for training with e-prop.

3 Experiments and Results

The experiments in this paper are based on a simulation of a simple flying vehicle in a minimalistic 3D environment with gravity and air resistance. The vehicle, which we refer to as rocket ball, is equipped with three rocket thrusters arranged around the vehicle with an 120° angle to each other, while pointing downwards in 45° angle. Each of the three thrusters can apply a force to the rocket ball that acts in the respective opposite direction. With this setup we conduct three different kinds of experiments which are described in the subsections below.

For measuring the inference performance of the rocket ball, we use an Euclidean distance error specified as $||E||_2$ which measures the distance between the center of the rocket ball and the center of the target sphere. Here an error of 1 is equivalent to the rocket ball's radius.

3.1 Short-term inference

In the initial set of experiments we tested the network's ability to infer motor commands in a simple setting where it has to steer the rocket ball to a desired goal location within the simulated world. In order to propagate the discrepancy between the currently predicted position delta and the distance to the target, we use BPTT as a baseline and compare it to the different e-prop inspired, biologically more plausible alternatives which all have the advantage that they do not depend on errors from the past and thus can be computed forward in time.

In Figure 2, one can see the results of this experiment, which is a fairly simple task for a trained LSNN and can be accomplished in far less time steps than those the network was trained on. Here the rocket ball quickly accelerates towards the target, then starts to hover around it and by doing so it continues to decrease its mean distance to the target.

The corresponding spike trains also show this behavior where the rocket ball is already pretty close to the target after only 10 time steps and then needs another 20 time steps to get really close to it. Another interesting observation one can make from the spike train, is that the network seems to operate in two cycles per (world) time step, such that two somewhat similar spike patterns occur in each (world) time step.

We also found that the performance of the different proposed e-prop based inference methods is astoundingly close to that of pure BPTT as can be seen in the left plot of Figure 3. While the LSNNs trained and inferred with BPTT initially manage to get closer to the target, after a while, once the rocket ball

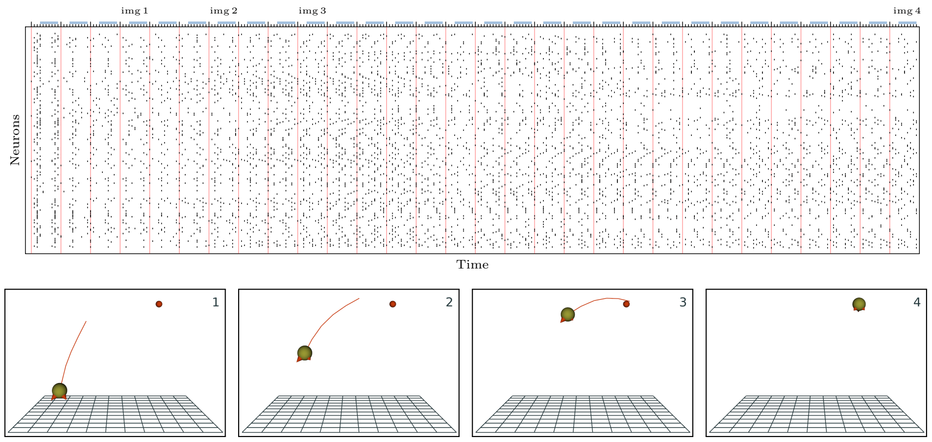

Fig. 2. Short-term inference (**bottom**) and corresponding spike-train (**top**) with BPTT and a time window of 5 time steps. The rocket ball starts on the bottom of the left side of the simulated world and quickly flies towards the desired goal location in the top right side where it then hovers around the goal position.

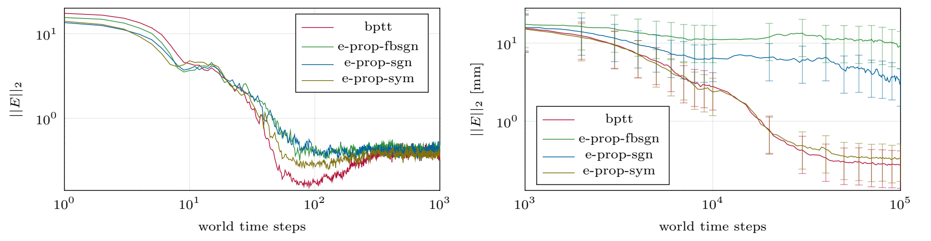

Fig. 3. Evaluation of different e-prop based inference algorithms: **sym**: symmetrical feedback weights, error flows back through the network weights. **sgn**: feedback weights are set to the sign of the corresponding weights. **fbsgn**: uses random feedback weights for propagating back the error from the outputs to the hidden neurons and the sign of the input weights to propagate it to the inputs. **bptt**: uses all network weights to propagate the error through the network and through time into the inputs. The **left** plot compares the median inference error for the different algorithms, using a network that was trained with e-prop. The **right** plot compares the algorithms in a goal-directed learning setup, where the network is trained continuously from scratch with e-prop while performing inference. Here the graph shows the median errors together with the upper and lower quantiles for 10 different runs and inference targets.

hovers around the target, their distance errors are the same as for the LSNNs trained and inferred with e-prop.

3.2 Continuous Goal-Directed Learning

In the second set of experiments we train our model in an online learning setting, where the rocket ball simulation continues indefinitely and the network has only the currently available experience in order to learn, which more closely resembles the learning environment of an agent acting in the real world. More formally, we use e-prop for training the LSNN in a single batch setting, where the network simultaneously trains its weights, while performing short-term motor inference for a total of 100,000 (world) time steps. Like in the first set of experiments, additionally to BPTT based inference, we also test our different proposed biologically plausible approaches, which are capable of online inference.

While the actual e-prop inference method had only a small impact on the overall inference performance, with a pretrained LSNN, in the continuous setting with goal-directed training only the symmetric e-prop variant (e-prop-sym) managed to perform similar to BPTT (see Figure 3 right plot). Using random feedback weights (e-prop-fbsgn) for propagating the inference error completely failed to get the network anywhere close to the target, and also the sign-based (e-prop-sgn) propagation did not reach the target in the simulated 100,000 time steps. For random feedback weights, this might be due to the fact that they initially completely differ from the actual output weights, and only during training potentially align themselves with the output weights. Also using the sign of the forward weights for inference error propagation might not foster enough goal-directed behavior to accelerate learning the forward model in the same way as the actual weights do.

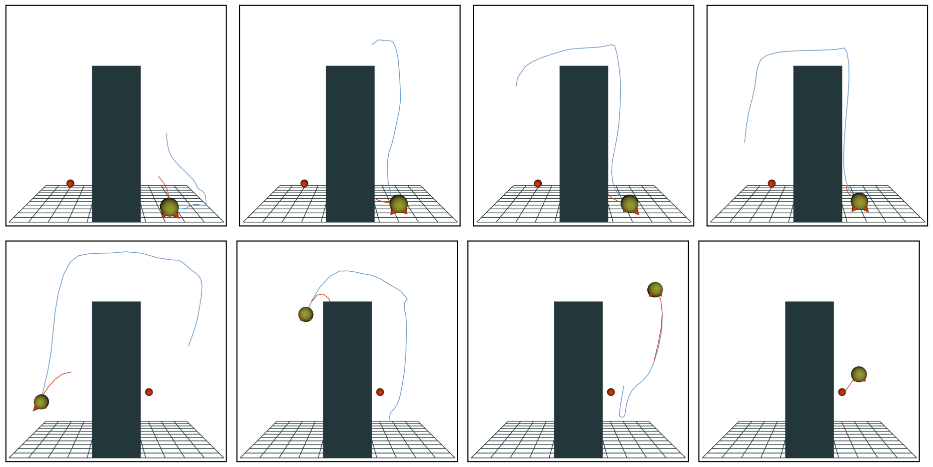

Fig. 4. Top: Emergence of a long-term trajectory within 7 time steps from a randomly initialized trajectory. **Bottom:** Long-term inference with BPTT with a long-term horizon of 30 time steps and a short-term horizon of 5 time steps. While the long-term inference (**blue trajectory**) manages to plan around the obstacle, the short-term inference (**red trajectory**) would collide with it. Once the rocket ball gets close to the target, the long-term inference is disabled in order to allow for a greater precision using only short-term inference.

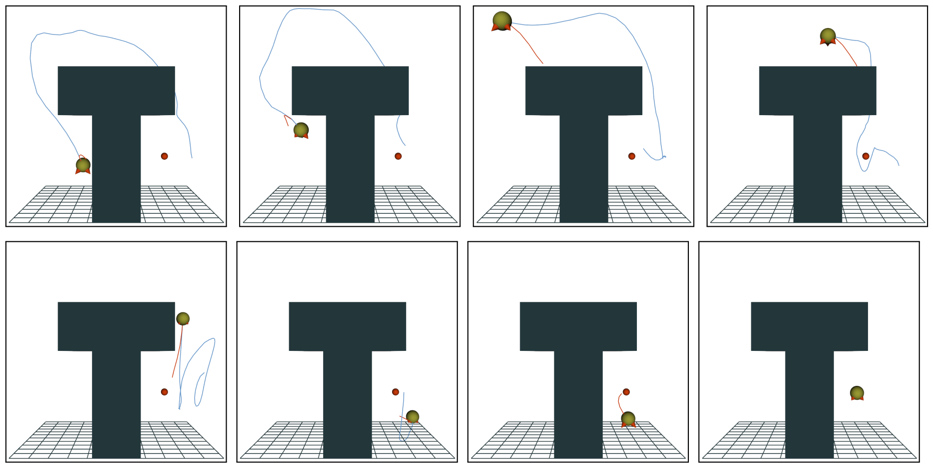

Fig. 5. Long-term inference with BPTT with a long-term horizon of 50 time steps and a short-term horizon of 5 time steps. While the long-term inference (**blue trajectory**) manages to plan around the obstacle, the short-term inference (**red trajectory**) would collide with it. Once the rocket ball gets close to the target, the long-term inference is disabled in order to allow for a greater precision using only short-term inference.

3.3 Long-Term Inference

The last kind of experiments challenge the network's ability to look beyond the immediate future and plan around obstacles. Therefore the closed loop operation is unrolled over an extended period of time steps into the future using only the network's own predictions as inputs. In the last time step, a target for the final prediction position deltas is given and backpropagated through time into the motor inputs from the current time step. Additionally the rocket ball uses axis-aligned distance sensors for which in each imagined time step zero is given as a target for the distance sensor predictions, regularizing the imagined trajectory in a way that the rocket ball avoids obstacles.

This setup then allows the network to perform long-term planning around concave as well as convex obstacles as can be seen in Figure 4 and 5. Here these planning trajectories spontaneously emerge, even when the initial endpoint of the imagined trajectory is behind the obstacle (see Figure 4). This suggests that the network has learned an internal model of the world including the obstacle and that by using the imagined distance sensor values for avoiding the obstacle, it can plan around it. This is not possible when using only short-term inference, where the rocket ball plans to fly directly towards the goal and therefore ends up stuck to the obstacle. Here the gradients towards the goal directly compete with the gradients from the distance sensors, and due to the short planning horizon the gradients from the distance sensors can not alter the trajectory in a way that the endpoint of the planned trajectory ends up behind the obstacle.

While in the majority of simulations a trajectory tends to form around the more challenging T-shaped obstacle and the rocket ball manages to navigate

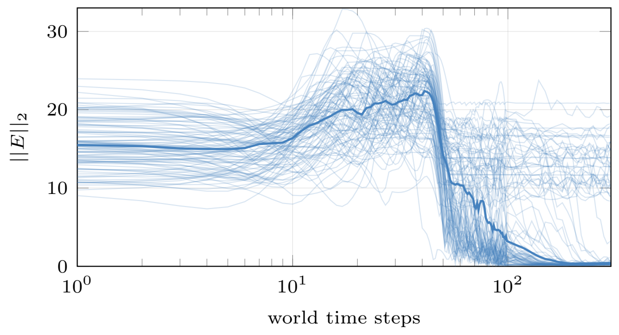

Fig. 6. Long-term inference with the T-shaped obstacle over 300 time steps, using a long-term horizon of 50 and a short-term horizon of 1 time steps. The median inference error (**bold**) is annotated with the individual runs (**thin**). While there are some fail cases, the median error shows that usually the network manages to navigate around the obstacle in within the first 50 time steps.

around it within the first 50 time steps, the long-term trajectories tend to be unstable, which explains the number of fail cases in Figure 6.

4 Conclusion

We have shown that spiking neural networks, and LSNNs in particular, can be trained to learn and memorize a temporal forward model of its environment, including obstacles. Moreover, using a biologically plausible inference algorithm inspired by e-prop, the LSNN can be used to induce short-term and long-term goal-directed action inference in continuous settings—results that are almost identical to BPTT in terms of accuracy, while having the advantage that they do not require the expensive backpropagation of errors through time.

In a continued goal-directed learning setup, the most biologically plausible methods could not compete with BPTT. However an e-prop variant using symmetric feedback weights also performed comparable to pure BPTT showing that LSNNs can not only be trained with a biologically plausible alternative to BPTT, they can also be employed in realistic online reinforcement learning setting, simulating the available experience of an agent in the real world. Here we demonstrated that they can learn and employ goal-directed behavior without the need for any kind of backpropagated error information from the future.

Apart from the more biologically plausible neural network processing mechanisms and local active inference techniques, more capable systems of this type may be implemented in hardware highly efficiently, promising to yield a much lower energy consumption [17]. We furthermore expect to enhance the LSNN with regularization terms, to foster even sparser temporal encodings.

References

1. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 795–805. NIPS'18, Curran Associates Inc., Red Hook, NY, USA (2018)
2. Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., Maass, W.: A solution to the learning dilemma for recurrent networks of spiking neurons. Nature communications **11**(1), 1–15 (2020)
3. Butz, M.V., Bilkey, D., Humaidan, D., Knott, A., Otte, S.: Learning, planning, and control in a monolithic neural event inference architecture. Neural Networks (May 2019)
4. Butz, M.V., Kutter, E.F.: How the mind comes into being: Introducing cognitive science from a functional and computational perspective. Oxford University Press (2016)
5. Butz, M.V., Menge, T., Humaidan, D., Otte, S.: Inferring event-predictive goal-directed object manipulations in reprise. In: International Conference on Artificial Neural Networks. pp. 639–653. Springer (2019)
6. Clark, A.: Surfing uncertainty: Prediction, action, and the embodied mind. Oxford University Press (2015)
7. Friston, K.: The free-energy principle: a rough guide to the brain? Trends in cognitive sciences **13**(7), 293–301 (2009)
8. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (dec 1997)
9. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
10. Nichol, A., Pfau, V., Hesse, C., Klimov, O., Schulman, J.: Gotta learn fast: A new benchmark for generalization in rl. arXiv:1804.03720 (2018)
11. OpenAI, :, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., d. O. Pinto, H.P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S.: Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 (2019)
12. Otte, S., Schmitt, T., Friston, K., Butz, M.V.: Inferring adaptive goal-directed behavior within recurrent neural networks. In: International Conference on Artificial Neural Networks. pp. 227–235. Springer (2017)
13. Otte, S., Stoll, J., Butz, M.V.: Incorporating adaptive rnn-based action inference and sensory perception. In: Artificial Neural Networks and Machine Learning – ICANN 2019. pp. 543–555. No. 11730 in Lecture Notes in Computer Science, Springer International Publishing (Sep 2019)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 (2017)
15. Traub, M., Legenstein, R., Otte, S.: Many-joint robot arm control with recurrent spiking neural networks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021), accepted for publication, preprint available (arXiv:2104.04064)
16. Yin, B., Corradi, F., Bohte, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. arXiv:2103.12593 (2021)
17. Yin, B., Corradi, F., Bohté, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. bioRxiv:2021.03.22.436372 (2021)