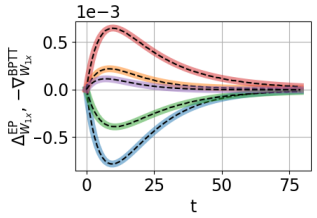
We thank the reviewers for their time, thorough reading of our paper and extremely useful comments that we much appreciated reading. We address the points that were raised below.

Reviewer 1. We have now improved figure quality, playing with transparency of the plain lines so that the dashed lines are now visible. We also have computed the wall-clock time of each algorithm on MNIST training and added it as an extra column to Table 1 to highlight the benefits of our discrete-time setting (P) compared to the real-time setting (EB), where #-h stands for the number of hidden layers used: EB-1h: 1 hrs 33 mins, EB-2h: 16 hrs 4 mins, P-1h: 17 mins, P-2h: 1 hr 56 mins, P-3h: 8 hrs 27 mins, P-conv: 8 hrs 58 mins. To emphasize these results in the conclusion, we have amended, l. 205: "no degradation of accuracy is seen between using the prototypical (P) rather than the energy-based (EB) setting, although the prototypical setting requires three to five times less time steps in the first phase (T) and cuts the simulation time by a factor five to eight."

Our RMSE(f,g) metric, defined in Appendix C.2.3, is normalized by the max of $||f||$ and $||g||$ so that it is invariant upon rescaling functions f and g. Our RMSE is therefore not the usual Root Mean Squared Error, and stands for "Relative Mean Squared Error". We clarified this by changing our acronym to RelMSE. The suggested cos metric is also meaningful, but lacks an important property in our context: the cos between f and g is maximal if and only if f and g are proportional, whereas we aim at reaching equality (Theorem 1). In contrast, our RelMSE metric is such that $\mathrm{RelMSE}(f,g)=0 \Leftrightarrow f(t)=g(t)$.



We also bring a precision about the notation $\partial\mathcal{L}/\partial\theta_t$. We chose this notation (the index t must be on θ, not on $\mathcal{L}$) to highlight the difference with RNNs trained with sequential data. In this situation, the total loss reads $\mathcal{L}=\sum_t \mathcal{L}_t$ with the sum index carried by the losses, and so does the gradient of the total loss. However in our setting: $\mathcal{L}_t=0$ for $t<T$, so that $\mathcal{L}=\mathcal{L}_T$. Then, the gradient $\partial\mathcal{L}_T/\partial\theta$ can itself be decomposed as $\partial\mathcal{L}_T/\partial\theta=\sum_{k=1}^{T}\mathcal{L}_T/\partial\theta_k$, with the sum index being carried by the parameters. We have added a clarification in our revised manuscript about this notation choice, in the paragraph preceding Eq.4.

Concerning RTRL, as suggested, we have amended the text, l. 124: "Other algorithms such as RTRL and UORO [Tallec and Ollivier, 2017] also compute the gradients by forward-time dynamics". Finally we have corrected, l. 41: "Originally, EP was introduced in the context of leaky-integrate neurons [Scellier and Bengio, 2017, 2019]. Computing their dynamics involves long simulation times, hence limiting EP training experiments to small neural networks".

Reviewer 3. To clarify the role of the conditions to be satisfied by the transition function F and as also suggested by Reviewer 4, we have now added the following verbal outline for Theorem 1, l. 118 just before the theorem statement: "The convergence requirement enables to derive the equations satisfied by the neural and weight updates (Lemma 4). Then, the existence of a primitive function ensures that these equations are equal to those satisfied by the gradients of BPTT (Corollary 3), with same initial conditions, yielding the desired equality (Theorem 1)". For completeness, Appendix A (in the paragraph between Lemma 4 and its proof) and B.2 respectively explain more precisely why we need such restrictions on F and that convergence is assumed without theoretical guarantee.

As suggested, we have clarified the position of our study with respect to standard RNN approaches in the BPTT background subsection, l. 95 and with Appendix A completed accordingly: "As detailed in Appendix A, the stability of the steady state implies that the gradients $\nabla_\theta^{\mathrm{BPTT}}(t)$ decay (i.e. vanish) exponentially fast, which ensures that $\frac{\partial\mathcal{L}}{\partial\theta}=\sum_{t=0}^{T-1}\nabla_\theta^{\mathrm{BPTT}}(t)$ converges, even if $T\to\infty$. In the context of convergent RNNs with a static input, vanishing gradients of BPTT are consequently not a problem and need not be addressed with LSTMs, as it is the case when learning from temporal data with RNNs".

Reviewer 4. The case of non symmetric weights is very interesting. It has been investigated in a real-time setting by Scellier et al. in their generalization of EP to vector-field dynamics (2018). As they show, learning still works in this setting, with forward and backward weights aligning throughout learning. We expect the same results in a discrete-time setting, with accelerated simulations, as we now have added it in the conclusion: "our discrete-time setting could also potentially accelerate simulations where the weights are not tied [Scellier, 2018]".

With regards to sequence tasks, we have added the following to the conclusion, l.219: "These results highlight that, in principle, EP can reach the same performance as BPTT on benchmark tasks - for RNN models *with fixed input*. One limitation of our theory however is that it has yet to be adapted to sequential data: such an extension requires to capture and learn correlations between successive equilibrium states corresponding to different inputs".

As suggested, we have added titles and legends to clarify each experimental set-up presented. For the sake of clarity, we have also specified the weight update implemented in the experiments, l. 169: "We study this architecture (...) with corresponding weight updates (14) and (16)". Moreover, we have also clarified when the effective weight update is being performed, l.113: "so that the effective weight update is performed at the end of the second phase." To improve the theory section, as also recommended by Reviewer 3, we have now provided a verbal outline of the proof of Theorem 1 - see answer to Reviewer 3. Finally, due to time and computational constraints inherent to hyperparameter tuning, we have not yet investigated what is the deepest network our algorithm can train, but we are addressing this now and aim to include these results in the camera-ready version.