Short-Term Plasticity Neurons Learning to Learn and Forget

Hector Garcia Rodriguez [1 2] Qinghai Guo [3] Timoleon Moraitis [1]

Abstract

Short-term plasticity (STP) is a mechanism that stores decaying memories in synapses of the cerebral cortex. In computing practice, STP has been used, but mostly in the niche of spiking neurons, even though theory predicts that it is the optimal solution to certain dynamic tasks. Here we present a new type of recurrent neural unit, the STP Neuron (STPN), which indeed turns out strikingly powerful. Its key mechanism is that synapses have a state, propagated through time by a self-recurrent connection-within-the-synapse. This formulation enables training the plasticity with backpropagation through time, resulting in a form of learning to learn and forget in the short term. The STPN outperforms all tested alternatives, i.e. RNNs, LSTMs, other models with fast weights, and differentiable plasticity. We confirm this in both supervised and reinforcement learning (RL), and in tasks such as Associative Retrieval, Maze Exploration, Atari video games, and MuJoCo robotics. Moreover, we calculate that, in neuromorphic or biological circuits, the STPN minimizes energy consumption across models, as it depresses individual synapses dynamically. Based on these, biological STP may have been a strong evolutionary attractor that maximizes both efficiency and computational power. The STPN now brings these neuromorphic advantages also to a broad spectrum of machine learning practice. Code is available at https://github.com/NeuromorphicComputing/stpn.

[1]Huawei Technologies – Zurich Research Center, Switzerland [2]University College London, United Kingdom [3]Advanced Computing & Storage Lab, Huawei Technologies, Shenzhen, China. Correspondence to: Timoleon Moraitis <timoleon.moraitis@huawei.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1. Introduction

1.1. Biological vs artificial neural networks

Biological neural networks are the source of inspiration for some of the most successful machine learning (ML) models, namely the artificial neural networks (ANNs) that underpin Deep Learning. Despite the success of ANNs, artificial intelligence (AI) models still pale in comparison to animals and humans in several respects. Widely acknowledged limitations include (a) the vast number of training episodes required before mastering a task, (b) difficulties in dynamically changing tasks, (c) the ad hoc task-specificity of ANN architectures, and (d) the high energy demands of running the algorithms on computers. Similarly, despite their biological inspiration at an abstract level, ANNs lack many of the computational operations that the known neuronal biophysics implies. It is conceivable that this latter disparity in implementation also underlies the former mismatch in performance. In particular, much of the complexity within biological neurons (a) is dedicated to synaptic plasticity for learning, (b) is governed by dynamically changing chemical concentrations, (c) is maintained across brain areas and species, and (d) accomplishes extreme energy efficiency. It is hard to ignore that these four biological properties that are largely missing from ANNs have a one-to-one correspondence to the four aforementioned limitations of ANNs. Based on this, our high-level ambition here is to explore whether these biological properties and the associated advantages can be brought to Deep Learning by a new rendition of a particular neuromorphic mechanism, i.e. short-term plasticity (STP) of synapses. Our approach is closely related to concepts from multiple subfields (Fig. 1.A), which we review in this section.

1.2. Neuromorphic Computing

Our goal here is aligned with the field of neuromorphic computing, which has been porting biophysical mechanisms from experimental neuroscience into models and emulating them in electronic circuits to improve neural computation (Indiveri & Horiuchi, 2011; Indiveri, 2021; Sarwat et al., 2022a;b). However, the focus has been mostly on the specific property of spiking neuronal activations, in so-called spiking neural networks (SNNs) (Maass, 1997; Ponulak & Kasinski, 2011). In addition, the field's aim has been over-

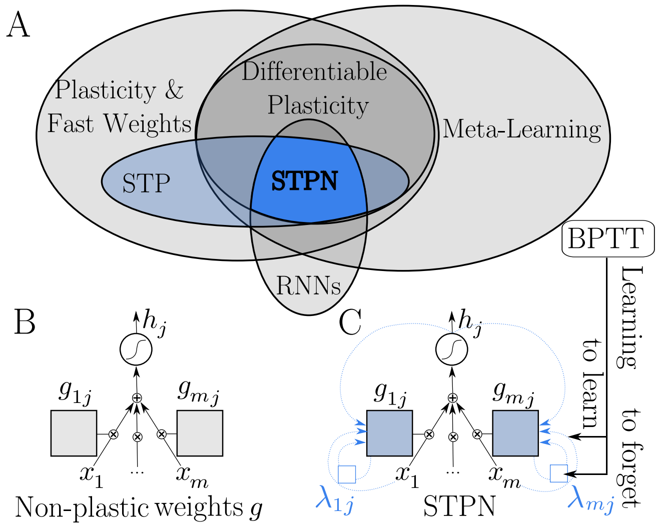

Figure 1. (A) Our STPN model in the literature's context. (B) Simple connectionist neuron, with synaptic efficacies g. (C) The STPN. Plasticity through recurrent parameters, e.g. λ.

whelmingly to increase energy *efficiency* of the state of the art (SOTA), rather than its *proficiency* such as classification accuracy or achieved reward in useful tasks. Nevertheless, recent results show that advantages of neuromorphic mechanisms are not limited to efficiency. If other properties than spikes are considered, such as STP, neuromorphic models can in fact also lead to more proficient models (Moraitis et al., 2020), while remaining compatible with efficient electronics (Sarwat et al., 2022a).

1.3. Plasticity and Short-Term Plasticity

The term "plasticity" refers to the rules that determine how the efficacy, i.e. strength, of synaptic connections changes in the brain or in model networks. The term is often reserved for local rules, i.e. when the changes of a synapse depend on signals from the pre-synaptic and/or the post-synaptic neuron that the synapse connects, and potentially a third signal, such as concentration of neuromodulators (Gerstner et al., 2018; Pogodin & Latham, 2020; Sarwat et al., 2022b). Such plasticity is generally considered the underlying principle of learning in the brain, and a possible path to life-long ML. One type of plasticity rule is STP (Zucker, 1989; Tsodyks & Markram, 1997; Chamberlain et al., 2008; Mongillo et al., 2008), i.e. a type of plasticity with strong biological evidence, whose effect is constrained in time. On the other hand, if each plastic change is persistent, the rule is a type of long-term plasticity. For example, Hebb postulated that biological weights are updated proportionally to pre- and post-synaptic activations (Hebb, 1949). Despite its simplicity, networks with Hebbian-like plasticity and without supervision can be used to optimize models such as Bayesian mixtures (Nessler et al., 2009) and to solve tasks such as handwritten digit classification fast and with robustness to adversarial attacks (Moraitis et al., 2021). Extensions of such plasticity can be both insightful and powerful, playing a key role in the ML SOTA (Nessler et al., 2009; Scellier & Bengio, 2019; Löwe et al., 2019; Limbacher & Legenstein, 2020; Millidge et al., 2020; Illing et al., 2021). Particularly relevant to the present manuscript are cases where a plasticity rule causes synaptic changes at a different timescale with respect to another learning rule. This concept of "subordinate" or "interleaved" changes in synapses has been termed dynamic weights (Feldman, 1982), fast weights (Hinton & Plaut, 1987; Schmidhuber, 1992; 1993; Tieleman & Hinton, 2009; Ba et al., 2016; Schlag & Schmidhuber, 2017), or simply plasticity or learning (Bengio et al., 1990; Moraitis et al., 2018b; Miconi et al., 2018; 2020; Moraitis et al., 2020; Miconi, 2021). Interestingly, some of these approaches that use associative plastic updates have been shown to be equivalent to attention mechanisms (Ba et al., 2016), and even to models like linear transformers (Schlag et al., 2021).

The short-term aspect of STP can be modelled by splitting the synaptic efficacy G that weights a presynaptic input, into a long-term weight W and an additive or multiplicative short-term component F, e.g. $G = W + F$. Subsequently, an update rule, which depends on local variables, increments F, which otherwise decays exponentially with time, implying a type of learning followed by forgetting. The literature has shown diverse functions emerging from the various forms of STP. For example, STP can apply temporal filtering on synaptic inputs (Rosenbaum et al., 2012), or underpin biophysical models of working memory (Mongillo et al., 2008; Szatmáry & Izhikevich, 2010; Fiebig & Lansner, 2017). If realized as the fast-weight counterpart to a concurrent slower plasticity, it can focus learning from sequences on multiple timescales of the input (Moraitis et al., 2018a;b). It also instils a long short-term memory to recurrently connected neural networks (RNNs), resulting in properties and performance similar to Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) units (Bellec & Salaj et al., 2018). In all these cases, STP was studied in the context of SNNs. The reason is that the increased biological plausibility of the spiking activations is a more useful model for many neuroscientists, and may bring energy efficiency advantages to neuromorphic engineering. However, SNNs are also harder to analyse mathematically or train practically, which limits the potential improvements from STP in useful ML tasks.

Nevertheless, two recent models with STP did outperform others without in specific tasks, even though the STP-equipped models were SNNs. Namely, first, STP led to improved learning of restricted Boltzmann machines from unbalanced data (Leng et al., 2018). Second, and in closer relation to our present study, a very simple SNN learned without supervision from the standard static MNIST dataset of handwritten digits (LeCun et al., 1998), but was tested on the classification of frames of a video of digits with moving occlusions (Moraitis et al., 2020). Surprisingly, owing to STP's dynamics at the input synapses, the SNN outperformed simple supervised convolutional neural networks and LSTMs, even trained on the video dataset with temporal context. Moraitis et al. (2020) also included a mathematical proof that neural networks with STP at their input synapses are in fact the optimal model for certain dynamic data. The key enabling principle is that input synapses with STP memorize not only dataset-wide features in the long-term weights W, but also recent features that are relevant to the immediate future in the short-term component F. However, the practical demonstrations of STP's advantages remain limited due to the reliance on spiking neurons. It should be noted that some of the non-spiking models with fast weights, particularly those co-authored by Hinton (Hinton & Plaut, 1987; Tieleman & Hinton, 2009; Ba et al., 2016). Hinton & Plaut (1987); Tieleman & Hinton (2009) did include STP-like decaying dynamics in the fast subordinate weight changes, however training of the STP's parameters was not reported. In Ba et al. (2016), backpropagation trained an RNN's conventional slow weights end-to-end, while additional fast weights were updated by a separate non-learnable rule, also during inference. This system was able to solve tasks where attention to the recent inputs is important. However, the authors did not include STP in input synapses, but only in recurrent connections between hidden units. Here, we hypothesize that STP's potential for Deep Learning can be realized by enabling STP also at the input synapses, as in the theoretically supported proposal of Moraitis et al. (2020).

In order to fulfil the promise of optimally adapting plasticity, one important element is missing from those previous studies. That is the optimization of the STP rule for each synapse and through learning, as opposed to fixed STP based on chosen hyper-parameters and uniform for all connections. How to realize this last ingredient is not immediately obvious. One recent work (Tyulmankov et al., 2022) did train a model equipped with the Hebbian STP mechanism of Moraitis et al. (2020). However, that model does not support recurrent connections between neurons, uses uniform plasticity parameters across all synapses, and focuses on simple associative tasks of random binary inputs. Therefore, the key advance was that the uniform hyperparameter in the earlier work was hand-tuned as opposed to optimized through backpropagation. Here we target advanced tasks, through fully recurrent models, and through the training of individual synapses' STP. To achieve this, we implement STP in a synapse as a sub-connection within that synapse, thus resulting in a formulation of STP Neurons (STPNs) as a novel type of recurrent unit (see Section 2). Through this formulation, the learning-and-forgetting function of STP becomes itself trainable, drawing links to the category of algorithms that are meta-learning, i.e. learning to learn.

1.4. Learning to learn

Meta-learning is a paradigm that applies machine learning to improve further learning in new domains. This has been considered analogous to the nesting of biological timescales of evolution, development, life-long skill learning, and learning for temporary objectives. In fact, direct empirical evidence for meta-plasticity, i.e. changes in plasticity, in the brain has been observed (Abraham & Bear, 1996). More generally, a large body of ML literature on meta-learning exists, with various approaches and applications (Schmidhuber et al., 1996; Thrun & Pratt, 1998; Bellec & Salaj et al., 2018; Hospedales et al., 2020). For instance, it has been shown that, through minimal modifications to how the training data is provided, RNNs learn to learn, where the inner learning loop consists in changes of recurrent state rather than changes of weights (Hochreiter et al., 2001; Wang et al., 2016). Clearly, some of the most relevant meta-learning methods with respect to our work here are those where learning to learn consists in learning the parameters of a plasticity rule that controls the changes of weights within the inner loop. Both evolutionary (Soltoggio et al., 2018) and gradient-based (Bengio et al., 1990) algorithms have long been described for such plasticity-rule meta-learning purpose. However, backpropagation-based end-to-end training of the individual-synapse plasticity parameters along with the other parameters of a neural network has only recently been demonstrated (Miconi et al., 2018). The trainable plasticity in that work outperformed non-plastic neural networks and has been followed up with extensions that confirm its advantages (Miconi et al., 2019; Beaulieu et al., 2020; Miconi, 2021). However, thus far, none of these has incorporated STP. It has not been obvious how to learn the spontaneous temporal dynamics of synaptic efficacy decay, i.e. how to learn the forgetting aspect of STP, for each synapse. As a matter of fact, in models without STP, fast weights are long-term, i.e. they persist through time, unless the fast-weight-update mechanism applies a learning increment or decrement. As a result, they lack a dedicated forgetting mechanism, and forgetting must be handled by the mechanism intended for learning. We believe that demonstrating the importance of learning to forget would be an important contribution to the meta-learning field, as it could possibly relate to the challenge of catastrophic forgetting in continual deep learning. Catastrophic forgetting refers to the phenomenon of neglecting a previously learned task because of learning a new one, and it is one of the important motivations for meta-learning research (Vuorio et al., 2018; Ren et al., 2018; Flennerhag et al., 2019; Javed & White, 2019; Hospedales et al., 2020; Sinitsin et al., 2020). It is plausible that learning, not only to learn, but also to explicitly forget would mitigate catastrophic forgetting. Here indeed we present a procedure of learning to learn and forget, realized as learning of STP parameters.

1.5. Contribution to the field

We present STPN, a new recurrent type of unit that expands the family of RNNs with the possibility of a recurrent state within each input synapse. It extends other fast-weight models by adding STP to inputs, and by making STP trainable per synapse. It builds on other models of differentiable plasticity by including a short-term aspect. It complements learning to learn with learning to forget. We will show that it is a better RNN choice than LSTM, surpasses the most recent fast-weight models, and outperforms other differentiable plasticity mechanisms, in a variety of tasks, with supervised and reinforcement learning, including examples of meta-learning. STPN's benefits comprise both improved task proficiency and energy efficiency.

2. The STPN model

2.1. The model

We begin to construct the STPN model by defining initially a simple feed-forward hidden unit with activation h_j, that receives an input vector $\boldsymbol{x}$ from a preceding layer, and weights it by a synaptic efficacy vector $\boldsymbol{g}_j$ (Fig. 1B). Two realizations (Fig. 1C) serve as intuition for casting such a model as a recurrent unit when STP is added. Firstly, the postsynaptic activation h_j may feed back to the input synapses as one of the factors of the plasticity rule, e.g if STP is Hebbian. Secondly, the short-term memory and decay of each efficacy g_{ij} implies that the synapse contains a state variable that is partly propagated to itself forward in time. Intuitively, these are two self-recurrent loops.

The specific type of STP that we choose to base the STPN model on is the one that was motivated theoretically in Moraitis et al. (2020). According to this, a hidden neuron j receiving an input vector $\boldsymbol{x}$ from a preceding layer, weights it by a synaptic efficacy vector $\boldsymbol{g}_j$ that consists of two additive components: $\boldsymbol{g}_j = \boldsymbol{w}_j + \boldsymbol{f}_j$. STP acts on $\boldsymbol{f}_j$. Meanwhile, component $\boldsymbol{w}$ may be fixed, or updated by a different learning rule, either concurrently with STP, or preceding it. In this work, $\boldsymbol{w}$ is updated via backpropagation in the outer loop between sequences. This STP rule dictates that f_{ij} in a synapse j from neuron i, firstly decays exponentially with time t, and secondly is subject to Hebbian plasticity proportional to the pre- and post-synaptic variables x_i and h_j.

Based on the intuition for the two self-recurrent loops we construct STPN as a recurrent unit that aims to have analogous functionality to the latter Hebbian STP rule. We introduce upper-case symbols $\boldsymbol{G} = \boldsymbol{W} + \boldsymbol{F}$ to indicate the matrices that connect the input vector to the hidden unit vector, and two equally-sized matrices $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}$ that contain elements λ_{ij} and γ_{ij}, which parameterize the STP. We will symbolize the element-wise and outer products by $\odot$ and $\otimes$ respectively.

One time-step's pass through a layer of STPN units in this basic version, and assuming fixed weights $\boldsymbol{W}$ and a non-linearity $\sigma(\cdot)$, is described by the following set of equations, where $\mathbb{1}$ indicates a matrix of ones:

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)} \tag{1}$$

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)}) \tag{2}$$

$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\Gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)}) + (\mathbb{1} - \boldsymbol{\Lambda}) \odot \boldsymbol{F}^{(t)} \tag{3}$$

The activation $\boldsymbol{h}$ depends on the short-term component $\boldsymbol{F}$ through $\boldsymbol{G}$ (Eqs. (1) and (2)). $\boldsymbol{F}$, in turn, depends on the activation and on itself (Eq. (3)), wherein lies the recurrency of this model. Unlike in standard RNNs, this recurrency does not necessitate recurrent connections between hidden states h_j, i.e. there are no synapses connecting units of the same layer. The recurrency is mediated in this case through the parameter matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ that connect the synaptic state to itself and to its postsynaptic neuron. Therefore, this model realizes an uncommon type of recurrent connectivity, characterized by meta-weights, i.e. connections (self-loop from f_{ij} through λ_{ij} in Fig. 1D) within each synaptic connection (blue square g_{ij} in Fig. 1D).

In meta-learning terms, the Hebbian STP of Eqs. (1)-(3) describes one iteration of an inner learning (and forgetting) loop that is unsupervised (see Section 2.3).

2.2. Equivalence to STP

It can be shown rather simply that the STPN construct is not merely analogous to a neuron with the Hebbian STP rule that we focus on, but it is exactly this - however in the discrete rather than the continuous time domain. The original equations of the rule dictate firstly an exponential decay over time with a rate $0 < \lambda_{ij} < 1$:

$$\frac{df_{ij}}{dt}^{(decay)} = -\lambda_{ij} f_{ij}. \tag{4}$$

Secondly, at any discrete time point that the synapse receives an input $X_i^{(t)}$, Hebbian plasticity with a learning rate γ_{ij} increments f_{ij} as well, by a Δf_{ij} that also depends on the post-synaptic output $h_j^{(t)}$:

$$\frac{df_{ij}}{dt}^{(Hebb)} = \Delta f_{ij}^{(t)(Hebb)} = \gamma_{ij} x_i^{(t)} h_j^{(t)}. \tag{5}$$

The combined effect of Eqs. (4) and (5) describes the full original STP rule. The continuous-time Eq. (4) can be approximated arbitrarily closely by small discrete time-steps

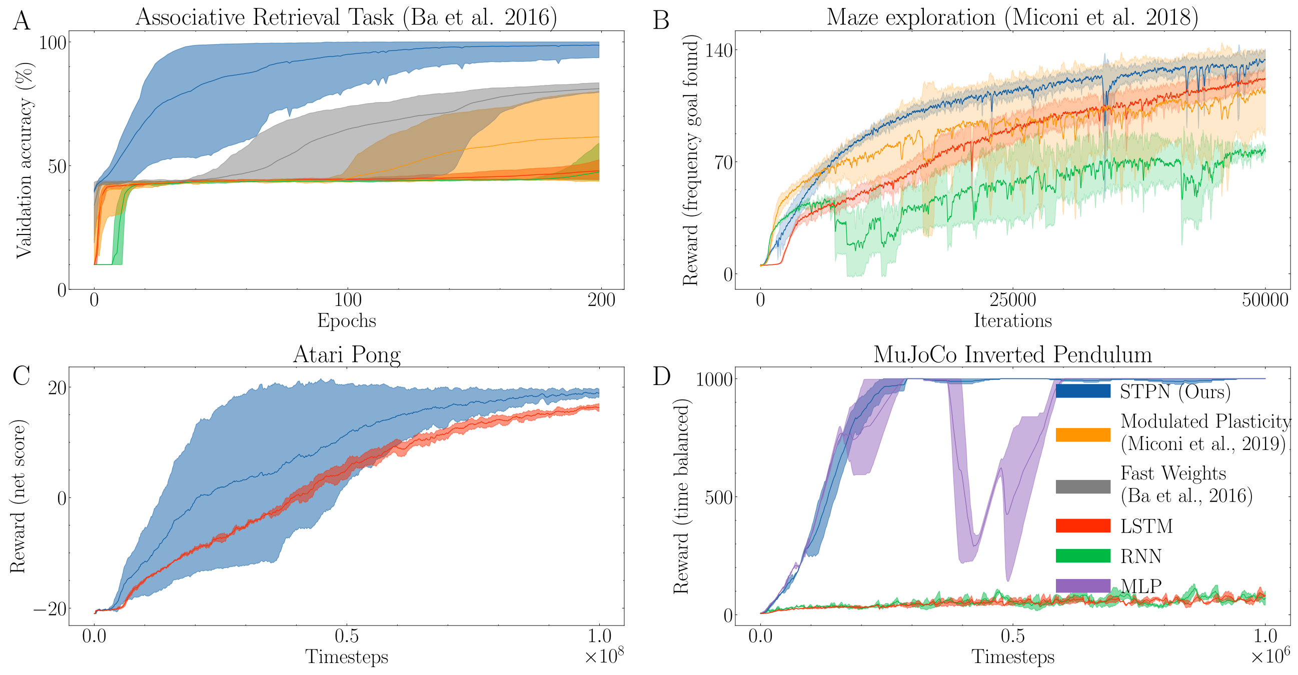

Figure 2. Proficiency during training of the STPN and other models across 4 tasks. Proficiency refers to: (A) Validation accuracy in ART (Ba et al., 2016); (B) Accumulated reward in an episode of the Maze Exploration task (Miconi et al., 2018); (C) Net score in the Atari game Pong (Bellemare et al., 2013) when one of the players reaches 21. (D) Number of timesteps in balance within an episode, capped at 1000, in Mujoco Inverted Pendulum (Todorov et al., 2012).

through the Euler method (Euler, 1794; Kendall et al., 1989):

$$\delta \boldsymbol{F}^{(t+1)(decay)} = -\boldsymbol{\Lambda} \odot \boldsymbol{F}^{(t)}. \quad (6)$$

This method has long been used to simulate the continuous-time evolution of neuromorphic models, such as spiking neurons, including recently (Woźniak et al., 2020).

On the other hand, Eq. (5) is already only dependent on discrete-time events, so in discrete time it remains the equivalent: $\Delta \boldsymbol{F}^{(t+1)(Hebb)} = \boldsymbol{\Gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})$. STPN's Eq. (3) can therefore be written as

$$\boldsymbol{F}^{(t+1)} = \boldsymbol{F}^{(t)} + \Delta \boldsymbol{F}^{(t+1)(Hebb)} + \delta \boldsymbol{F}^{(t+1)(decay)}, \quad (7)$$

which shows that the model indeed is equivalent to discrete-time Hebbian STP.

2.3. Learning to learn and forget with STPN

STPN's parameters γ_{ij} and λ_{ij} are interpretable quite concretely in two ways. Firstly, as parameters of recurrent connections, they have a clear role of short-term memory. Secondly, γ is also the Hebbian learning rate of the synapse, whereas λ is its forget rate. Further, by framing the model as a network that is based on standard recurrent weighting operations, STPN's parameters are trainable. That is, not only the standard between-neuron connections $\boldsymbol{W}$, but also the characteristic connections-within-synapses can be trained. Notably, as these latter learned connections act as STP's rates of learning and forgetting, then training these parameters realizes a learning-to-learn scheme that also learns to forget. In this realization of meta-learning, the inner learning loop is the online unsupervised adaptation of the network through STP (Eq. (3)) to a given input sequence, whereas the outer learning loop consists in the optimization of the network's parameters, e.g. via backpropagation through time (BPTT), for the inner online learning task, over multiple examples of this inner task (see Algorithm 1 in the Appendix).

2.4. STPN variants

Even though thus far we have provided a description of STPN based on a feed-forward base structure (Eq. (2)), the same connection-within-a-synapse type of recurrency and plasticity can also be added to models that have recurrency between hidden units, like simple RNNs do. This simply changes the unit to also receive inputs from the same hidden layer, i.e. changes Eq. (2) into

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}[\boldsymbol{x}^{(t)}; \boldsymbol{h}^{(t-1)}]), \quad (8)$$

accompanied by the corresponding change in size of the parameter matrices $\boldsymbol{F}$ and $\boldsymbol{W}$. We refer to the networks of Eqs. (2) and (8) as the STPNf and STPNr variants, respectively, of STPN connectivity (for their **F**eed-forward and **F**ully connected, or **R**NN non-plastic skeleton). This formulation is extensible to further variants, such as STPNl for an STPN with the addition of LSTM's gating, although those are not explored in this work. The results in Section 4 are achieved using the STPNr in most tasks, except for a fully-observable robotics environment where we tested the STPNf.

3. Methods

3.1. Training methods

We found that, given the complexity of our model's per-synapse STP parameters, and especially for the more complex STPN models that include recurrency both in the synapse and between neurons such as the STPNr, training stability and robustness are not possible automatically. We explored approaches for making the STPN's training robust, and arrived in a strategy that consists in an initialization method and a weight-normalization approach that we introduced. See Appendix Algorithm 1 and Appendix A.4 for details.

3.2. Energy consumption measurement

Our approach focuses on the synaptic weighting mechanism, and as such it could provide guidelines for future neuromorphic hardware. Its energy consumption is therefore a key concern. Given that STP is a highly neuromorphic mechanism inspired by biophysics, it is probable that it is also highly efficient. In principle, STP has the potential to depress synaptic currents flexibly and independently through Hebbian decrements and short-term decay, which is a reason to intuitively expect high energy-efficiency from STPN. With this in mind, we measured the power consumption that each model would incur through weighting operations in hypothetical analog neuromorphic hardware. In such hardware, presynaptic input $\boldsymbol{x}$ is provided as a voltage V, whereas the matrix $\boldsymbol{G}$ of synaptic efficacies g could be represented in an array of resistive devices with conductance g, such as memristors (Sarwat et al., 2022a). This enables weighting and summation through Ohm's and Kirchhoff's laws, and incurs a power consumption by each synapse, which is proportional to the conductance g and the square of the applied input voltage V^2, as shown in Eq. (9)

$$P = VI = V\left(\frac{V}{R}\right) = V^2\left(\frac{1}{R}\right) = V^2 g, \qquad (9)$$

where I is the electrical current, and R is the resistance. This method allowed us to measure the hypothetical power consumption of the various models using $P = x^2|g|$. Notably, analog signals underlie synaptic transmission also in biological synapses, through ions. Fewer ions and smaller currents are transmitted when the biological synapse is depressed. Therefore, to the extent that STP is biologically plausible, our measurements of the STPN's power consumption provide some insight also into STP's role in the brain's energy budget.

3.3. Tested baseline models

In our experiments, we compare the STPN with other widely used networks with memory, like standard RNN and LSTM, and others with different kinds of synaptic memories, like RNN with Fast Weights (Ba et al., 2016) and Modulated plasticity RNN (Miconi et al., 2019). Given the qualitative architectural differences across the tested models, we make them comparable by choosing hidden sizes that correspond to equal number of parameters between models in each experiment. Ba et al. (2016) augmented RNNs by adding **Fast Weights** to some connections, specifically between recurrent neurons, i.e. not in feedforward synapses such as from the input layer. Also, the hyperparameters of plasticity are uniform throughout all fast-weight synapses: Hebbian update factor and decay of current memories. Additionally, they describe the use of an inner loop where fast weights can repeatedly act on an intermediate hidden state and be updated, while the effect of slow weights acts as a sustained boundary condition in each iteration. However, they found no significant empirical benefit in performing multiple such iterations; and neither do we when tuning this baseline. Additionally, note that this same mechanism can also be applied to any other network with synaptic memories, and is conceptually closer to work on recurrent processing of the same input as a form of adding computational resources (Schwarzschild et al., 2021a;b; Banino et al., 2021). Fast Weights RNN is a conceptually simpler version of STPN, primarily due to its use of plasticity only on recurrent connections, the plasticity rule being uniform across neurons and synapses, and the fact that parameters modulating fast learning and forgetting are not trained (but tuned using the validation set). Other mechanisms implemented by the Fast Weights network, like layer-normalization of the hidden activations, and a repeated application of the plasticity through multiple recurrent iterations on each time-step of the input sequence. Our control experiments (not shown) showed that this iterative aspect offers only a small improvement, and could be equally applied to STPN. Therefore, the Fast Weights model serves as a good baseline to the training of a synapse-specific STP mechanism. Miconi et al. (2018) exploit the idea of optimizing plasticity by training parameters that control each synapse's memory via backpropagation, hence **Differentiable Plasticity**. Miconi et al. (2019) further add a modulating term to the plasticity, akin to a third factor of three-factor plasticity rules, which is also trainable. Unlike STPN, in that prior work no variant could evolve (e.g. decay) its efficacies spontaneously with time, like STPN

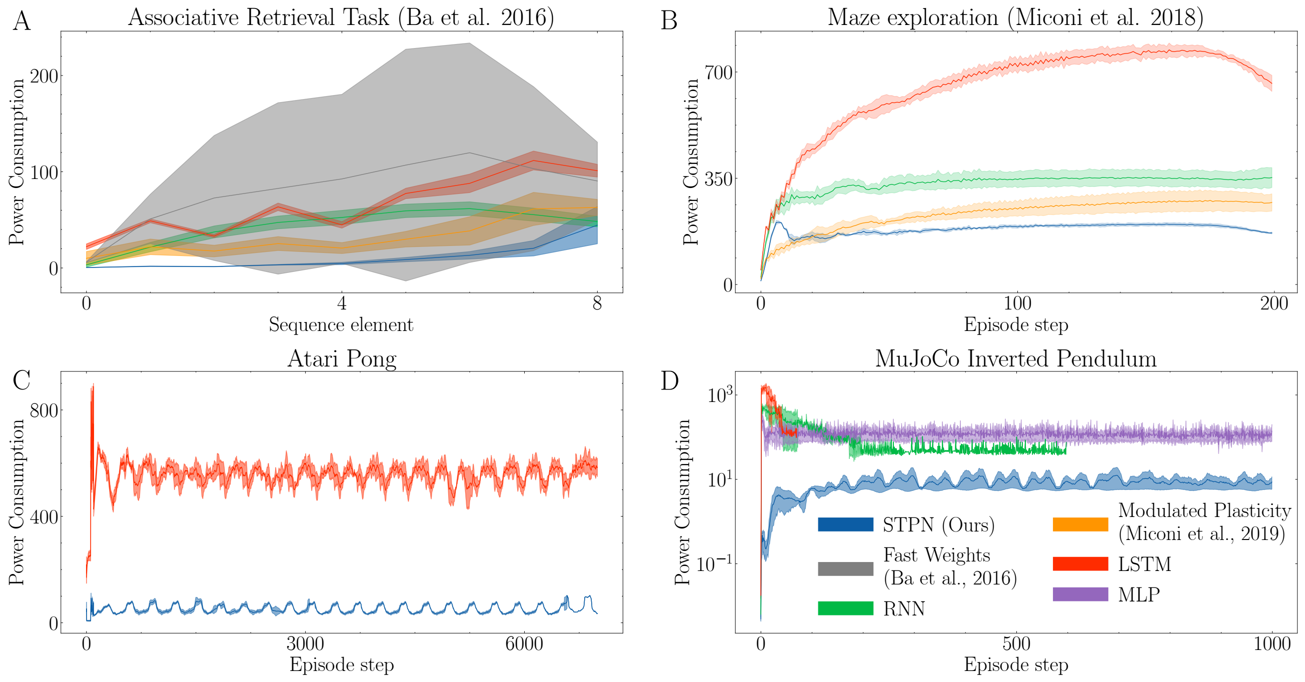

Figure 3. Power consumption of trained models during inference, evaluated at each timestep. Mean and standard deviation over multiple random seeds are shown, averaged over multiple inference runs. Lower is better.

does through λ. We select the modulated plasticity variant ("Modplast") as a baseline due to its superior performance among the variants tested in the Maze Exploration task in (Miconi et al., 2019). The other two major differences in this specific model with respect to STPN are: a) plasticity only in recurrent synapses, b) a meta-trained parameter per synapse that clips the plastic part of the efficacy. This model being the best out of multiple variants of trainable plastic RNNs serves as a good baseline for the STPN. **HebbFF** (Tyulmankov et al., 2022) is architecturally equivalent to STPNf with uniform plasticity. However, it offers no option for per-synapse plasticity parameters, recurrent between-neuron connections, or mechanisms to stabilize plastic updates such as the ones that we introduced Section 3. These become necessary in tasks with higher complexity than those tested in (Tyulmankov et al., 2022), which we tested and confirmed. Specifically, given that HebbFF could be described superficially, as a highly-simplified version of the STPN, we include comparisons to HebbFF alongside several other simpler STPN versions in Appendix A.2. We also carry out experiments using networks with non-synaptic memories, but which are general purpose ANNs with memory, and shown to perform meta-learning through the encoding of their neuronal memories (Wang et al., 2016). We compare the performance of **RNN** (a direct comparison to STPNr but without synaptic STP, and a different type of memory to STPNf) and **LSTM** (as a more complex non-plastic RNN with gating and an additional memory mechanism).

3.4. Tested tasks

Associative Retrieval Task (ART) (Ba et al., 2016) tests the capabilities of a network to successfully store associations between pairs of elements seen in a sequence, and retrieve one of the elements of the pair (the value) when queried with the other (the key) at the end of the sequence. This task and other variants (Schlag & Schmidhuber, 2017; Le et al., 2020) are therefore commonly used to compare the abilities of networks with memories of different nature. For the setup of this task, we mostly follow the experimental details provided in Ba et al. (2016), besides the modifications described in Appendix A.5.1. **Maze Exporation:** Maze or grid-like tasks have been commonly used in RL as a data efficient and interpretable task to test RL algorithms. (Miconi et al., 2018) instantiates a specific form of such maze, in which an agent has an egocentric view of its surroundings and whose goal is to navigate towards a reward, which it cannot see and whose position is randomized across episodes. Furthermore, the agent is randomly relocated to another position in the grid upon reward-finding and at the start of an episode. Given this environment design, we can think of the Maze Exploration task as a test to the meta-learning capabilities of the agents. Firstly, each maze instantiation (with a fixed reward position) represents the inner loop of learning, as the agent needs to learn how to effectively navi-

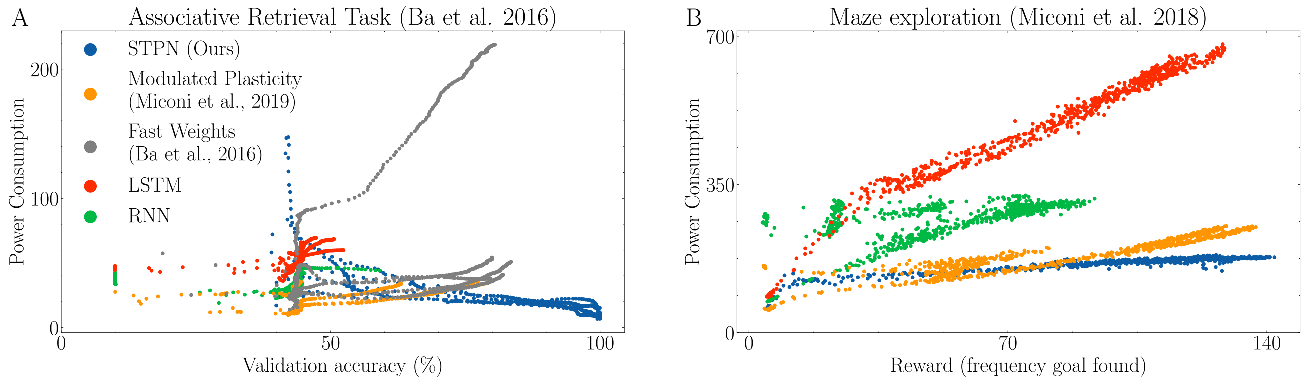

Figure 4. Energy efficiency as a function of task-specific proficiency, measured at regular intervals during training. Each scattered point is one instance of the model at a specific point of learning. All seeds for a model shown with the same color.

gate towards this new reward position, and do so from each initial random position after relocation when the reward is found. Secondly, across different episodes, parameters need to be optimized as to help within the inner loop and through different instantiations of the maze, hence they are learned to learn. Additionally, even within an episode, we can distinguish two qualitatively different phases: in the first steps, the agent needs to explore this instantiation of the maze in order to find the reward location. Upon learning of the positioning of the goal, it changes to an exploitation phase in which the key is to quickly reach the already seen reward. These two task distributions (the distribution of all possible mazes and the binary exploration-exploitation phases) positions this task as a good benchmark for the meta-learning and sequence adaptiveness capabilities of non-plastic and plastic networks with memory. **Atari games and MuJoco simulated robotics:** The previous two experimental setups, namely ART and Maze Exploration, were presented in plasticity related articles and have some characteristics that might favor models with plastic connections. RL is an area of research where the use of RNNs and its variants is still widespread in SOTA algorithms, more so in comparison to supervised learning domains regarding language, video or audio. In order to go beyond simple tasks to which the plastic networks literature has been limited, in the interest of exploring a domain in which RNNs are a component of SOTA approaches, and with the additional intent to examine the use of STPN as a general-purpose network with memory, we present some preliminary experiments with two tasks from two common Deep RL benchmarks: Atari Pong and MuJoCo Inverted Pendulum. **Pong** is an Atari 2600 game implemented in the Arcade Learning Environment (ALE) (Bellemare et al., 2013) in which the player faces an opponent, each player controls a bar that they can move up and down in order to hit a ball and therefore score a goal or stop the opponent from doing so. The game stops when a player reaches a score of 21. The reward obtained by the agent is the net score of the finished game, and the observations are the game frames. For our experiments in Pong, we use A2C, the synchronous version of A3C (Mnih et al., 2016) as the learning algorithm. The agent's network uses convolutional layers such as those in Mnih et al. (2016), a recurrent policy (shared by value and action branches) and non-plastic, feed-forward value and action branches. A simple algorithm such as A2C allows to better explore the real impact of the recurrent unit, and its previous use in the Maze task provides us with some assurances of its compatibility with plastic networks. Beyond the shift of domain from toy, plastic-network favoring tasks, Pong offers the first situation in which STPN is embedded in a larger network, and whose input is not sparse and directly from the environment, but dense and the result of the processing of multiple layers. Note because we do not perform frame stacking, the environment is a partial observable Markov decision process (POMDP), given that velocity of the ball cannot be inferred from a single frame (Hausknecht & Stone, 2015). This need for agents with memory justifies the use of recurrent policies and lack of comparison with a feed forward policy. MuJoCo (Todorov et al., 2012) is a physics engine widely used for research in robotics and reinforcement learning. **Inverted Pendulum** is one of the simplest tasks within MuJoCo, where the agent aims to balance an inverted pendulum which sits on a cart, by moving the cart forward and backward in one dimension with continuous valued controls. Our experiment with MuJoCo's inverted pendulum represents a preliminary step in the direction of using STPN in more complex problems in robotics and continuous control like those in the MuJoCo environment, and in combination more advanced RL algorithms, like Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) in this case. Unlike Pong without frame stacking, InvertedPendulum is a fully observable Markov decision process (MDP), so memory of the past is not needed to assess the present or predict the future. Therefore, training the parameters that process hidden memories could represent an unnecessary effort in terms of credit assignment, aggravated by hidden states being of much greater dimensional that environmental observations. Therefore, we use the STPNf in this case, and MLP as a baseline; in addition to RNN and LSTM. For the **terminology** we use for different timescales (e.g. episodes, epochs, timesteps), see Appendix A.6.

4. Results

4.1. Accuracy & reward

Fig. 2 shows that the STPN is more proficient, i.e. obtains larger validation accuracy and reward, than all other baselines. These plots report the mean and standard deviation over multiple seeds, except in Fig. 2.C , where we follow recommendations given in Agarwal et al. (2021) and report within 0.25 inter-quantile ranges, due the known propensity to certain catastrophic and non-representative runs of brittle algorithms like A2C in ALE. Furthermore, we find the unsupervised adaptability of STPN's plastic weights offers greater performance at inference time, as detailed in Table 3. Appendix A.2 presents the empirical advantages of including recurrent connections and learning per-synapse plasticity for STPN, with respect to non-recurrent or uniform plasticity versions of STPN. Additionally, we present benefits over HebbFF (Tyulmankov et al., 2022), a simpler feed-forward STPN with uniform plasticity.

Across all tasks, we observe STPN show noticeable stability in the learning process while achieving higher proficiency earlier in training. This is particularly clear in the Inverted Pendulum task, where the mid-training instability of MLP, also reported by Schulman et al. (2017), is counteracted by the addition of STP to the synapses. We hypothesize that compared to other plastic models, explicitly learning decaying dynamics of synaptic memories helps adding stability to processing of the same inputs across time, as less relevant synapses will be closer to their long-term value w_{ij}.

STPN seems to be more robust to hyperparameter choice than other tested baselines. Three examples are: a) the fact that it outperforms modulated plasticity, a similarly complex plastic model, with the same parameters that are tuned for that specific network; b) in Pong, STPN can tolerate a larger learning rate than LSTM, as the latter was unable to learn with high learning rate for most random seeds, such that it needed a lower learning rate for its best mean final reward; and c) like in Maze Exploration, STPN outperforms the model for which the experiment hyperparameters are tuned in Inverted Pendulum.

4.2. Energy consumption

The initial hypothesis regarding the trainable dynamical suppression of synapses, specially to input synapses, improving the energy efficiency of backpropagated-trained models is unequivocally confirmed in our experiments, as shown in Figs. 3 and 4. See also Appendix Table 4. Unlike for proficiency metrics like accuracy or reward, the networks are not given any explicit signal or instruction to be more energetically efficient, making these results even more remarkable. However, in some cases a common goal can improve both proficiency and efficiency in the STPN: suppress unimportant synapses. We believe these results show such a goal is met. This is particularly clear in Fig. 4. (A) As the network optimizes proficiency, its energy consumption decays. Similarly, in Fig. 4. (B) The growth of energy with respect to reward is minimal relative to other models, and largely attributable to past reward being part of the agent's observation in this experiment (hence a higher reward leading to a higher norm in the input and higher energy consumption).

4.3. Learning to learn and forget

Standard RNNs are seen as being able to meta-learn the encoding of a learning algorithm within its neuronal memories (Wang et al., 2016). STPN adds a synaptic state, which is updated by three meta-learned parameters (the long term synaptic efficacy, and the two STP parameters λ, γ), and has the potential of a higher influence in the output of the unit as compared to neuronal memories. The higher proficiency achieved by STPN on meta-learning tasks shows a learning-to-learn process is improving the predictive performance of the network. Further evidence for learning to learn and forget in the STPN is the following. Firstly, high proficiency is reached early in training, without incurring in too many instabilities or preventing further optimization at later stages of training, as can be seen in Fig. 2. Secondly, the negative slope of the efficiency vs proficiency curve of the STPN scatter points (Fig. 4) indicates a learning-to-forget mechanism, that improves both aspects.

5. Discussion

STPN is founded on prior theoretical optimality proofs and biological evidence. As a result, STPN is more performant, efficient and learns to better learn and forget than a variety of other models, and in a broad range of difficult tasks, by introducing individually trainable short-term plasticity to all synapses. The model also offers a new method to increase the efficiency of neuromorphic platforms. In combination with the prior experimental evidence that supports the biological plausibility of our plasticity, our results also raise STP's significance for animal behaviour and for the brain's energy budget. This outcome paves new and interdisciplinary research avenues, in RNNs, meta-learning, neuromorphic computing, and neuroscience, which we hope to see explored.

References

Abraham, W. C. and Bear, M. F. Metaplasticity: the plasticity of synaptic plasticity. *Trends in neurosciences*, 19(4): 126–130, 1996.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *CoRR*, abs/2108.13264, 2021. URL https://arxiv.org/abs/2108.13264.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. Using fast weights to attend to the recent past. *Advances in Neural Information Processing Systems*, 29: 4331–4339, 2016.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

Bellec & Salaj, Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/c203d8a151612acf12457e4d67635a95-Paper.pdf.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47: 253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*. Citeseer, 1990.

Chamberlain, S. E., Yang, J., and Jones, R. S. The role of nmda receptor subtypes in short-term plasticity in the rat entorhinal cortex. *Neural plasticity*, 2008, 2008.

Euler, L. *Institutiones calculi integralis*. Academia Imperialis Scientiarum, 1794.

Feldman, J. A. Dynamic connections in neural networks. *Biological cybernetics*, 46(1):27–39, 1982.

Fiebig, F. and Lansner, A. A spiking working memory model based on hebbian short-term potentiation. *Journal of Neuroscience*, 37(1):83–96, 2017.

Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019.

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits*, 12:53, 2018.

Hausknecht, M. J. and Stone, P. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*, pp. 29–37. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673.

Hebb, D. O. *The organisation of behaviour: a neuropsychological theory*. John Wiley & Sons, Inc., New York, 1949.

Hinton, G. Using fast weights to store temporary memories, 2017. URL https://www.youtube.com/watch?v=GLmptInTNSw.

Hinton, G. E. and Plaut, D. C. Using fast weights to deblur old memories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, pp. 177–186, 1987.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hochreiter, S., Younger, A. S., and Conwell, P. R. Learning to learn using gradient descent. In *IN LECTURE NOTES ON COMP. SCI. 2130, PROC. INTL. CONF. ON ARTI NEURAL NETWORKS (ICANN-2001*, pp. 87–94. Springer, 2001.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

Illing, B., Ventura, J., Bellec, G., and Gerstner, W. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. *Advances in Neural Information Processing Systems*, 34, 2021.

Indiveri, G. Introducing 'neuromorphic computing and engineering'. *Neuromorphic Computing and Engineering*, 1(1):010401, 2021.

Indiveri, G. and Horiuchi, T. K. Frontiers in neuromorphic engineering, 2011.

Javed, K. and White, M. Meta-learning representations for continual learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and

Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f4dd765c12f2ef67f98f3558c282a9cd-Paper.pdf.

Kendall, E. A. et al. An introduction to numerical analysis. *John Wiley and Sons Inc., New York, USA*, pp. 37–45, 1989.

Le, H., Tran, T., and Venkatesh, S. Self-attentive associative memory. In *International Conference on Machine Learning*, pp. 5682–5691. PMLR, 2020.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Leng, L., Martel, R., Breitwieser, O., Bytschok, I., Senn, W., Schemmel, J., Meier, K., and Petrovici, M. A. Spiking neurons with short-term synaptic plasticity form superior generative networks. *Scientific reports*, 8(1):1–11, 2018.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., and Stoica, I. RLlib: Abstractions for distributed reinforcement learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3053–3062. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/liang18b.html.

Limbacher, T. and Legenstein, R. H-mem: Harnessing synaptic plasticity with hebbian memory networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f6876a9f998f6472cc26708e27444456-Abstract.html.

Löwe, S., O'Connor, P., and Veeling, B. S. Putting an end to end-to-end: Gradient-isolated learning of representations. *arXiv preprint arXiv:1905.11786*, 2019.

Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9): 1659–1671, 1997.

Miconi, T. Learning to acquire novel cognitive tasks with evolution, plasticity and meta-meta-learning. *arXiv preprint arXiv:2112.08588*, 2021.

Miconi, T., Stanley, K., and Clune, J. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pp. 3559–3568. PMLR, 2018.

Miconi, T., Rawal, A., Clune, J., and Stanley, K. O. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lrAiA5Ym.

Miconi, T., Rawal, A., Clune, J., and Stanley, K. O. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv preprint arXiv:2002.10585*, 2020.

Millidge, B., Tschantz, A., and Buckley, C. L. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*, 2020.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

Mongillo, G., Barak, O., and Tsodyks, M. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, 2008.

Moraitis, T., Sebastian, A., and Eleftheriou, E. The role of short-term plasticity in neuromorphic learning: learning from the timing of rate-varying events with fatiguing spike-timing-dependent plasticity. *IEEE Nanotechnology Magazine*, 12(3):45–53, 2018a.

Moraitis, T., Sebastian, A., and Eleftheriou, E. Spiking neural networks enable two-dimensional neurons and unsupervised multi-timescale learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018b.

Moraitis, T., Sebastian, A., and Eleftheriou, E. Optimality of short-term synaptic plasticity in modelling certain dynamic environments. *arXiv preprint arXiv:2009.06808*, 2020.

Moraitis, T., Toichkin, D., Chua, Y., and Guo, Q. Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks, 2021.

Nessler, B., Pfeiffer, M., and Maass, W. Stdp enables spiking neurons to detect hidden causes of their inputs. *Advances in neural information processing systems*, 22: 1357–1365, 2009.

Pogodin, R. and Latham, P. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7296–7307. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/517f24c02e620d5a4dac1db388664a63-Paper.pdf.

Ponulak, F. and Kasinski, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, 71(4): 409–433, 2011.

Ren, M., Liao, R., Fetaya, E., and Zemel, R. S. Incremental few-shot learning with attention attractor networks. *arXiv preprint arXiv:1810.07218*, 2018.

Rosenbaum, R., Rubin, J., and Doiron, B. Short term synaptic depression imposes a frequency dependent filter on synaptic information transfer. *PLoS computational biology*, 8(6):e1002557, 2012.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Sarwat, S. G., Kersting, B., Moraitis, T., Jonnalagadda, V. P., and Sebastian, A. Phase-change memtransistive synapses for mixed-plasticity neural computations. *Nature Nanotechnology*, 17(5):507–513, 2022a.

Sarwat, S. G., Moraitis, T., Wright, C. D., and Bhaskaran, H. Chalcogenide optomemristors for multi-factor neuromorphic computation. *Nature communications*, 13(1): 1–9, 2022b.

Scellier, B. and Bengio, Y. Equivalence of equilibrium propagation and recurrent backpropagation. *Neural computation*, 31(2):312–329, 2019.

Schlag, I. and Schmidhuber, J. Gated fast weights for on-the-fly neural program generation. In *NIPS Metalearning Workshop*, 2017.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Schmidhuber, J. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Schmidhuber, J. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *International Conference on Artificial Neural Networks*, pp. 460–463. Springer, 1993.

Schmidhuber, J., Zhao, J., and Wiering, M. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34, 2021a.

Schwarzschild, A., Gupta, A., Ghiasi, A., Goldblum, M., and Goldstein, T. The uncanny similarity of recurrence and depth. *arXiv preprint arXiv:2102.11011*, 2021b.

Sinitsin, A., Plokhotnyuk, V., Pyrkin, D., Popov, S., and Babenko, A. Editable neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJedXaEtvS.

Soltoggio, A., Stanley, K. O., and Risi, S. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

Szatmáry, B. and Izhikevich, E. M. Spike-timing theory of working memory. *PLoS computational biology*, 6(8): e1000879, 2010.

Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998.

Tieleman, T. and Hinton, G. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1033–1040, 2009.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Tsodyks, M. V. and Markram, H. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the national academy of sciences*, 94(2):719–723, 1997.

Tyulmankov, D., Yang, G. R., and Abbott, L. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557, 2022.

Vuorio, R., Cho, D.-Y., Kim, D., and Kim, J. Meta continual learning. *arXiv preprint arXiv:1806.06928*, 2018.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Woźniak, S., Pantazi, A., Bohnstingl, T., and Eleftheriou, E. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6):325–336, 2020.

Zucker, R. S. Short-term synaptic plasticity. *Annual review of neuroscience*, 12(1):13–31, 1989.

A. Appendix

A.1. The meta-learning algorithm

Algorithm 1 STPN learning to learn and forget in a supervised meta-learning setting

```
Initialize weights and meta-parameters $\boldsymbol{\Theta} = [\boldsymbol{W}; \boldsymbol{\lambda}; \boldsymbol{\gamma}]$
▷ Outer-loop learning. E.g. BPTT.
while training do
    ▷ Sample input sequence.
    $\boldsymbol{\mathcal{S}} = [\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}] \sim \mathcal{D}$
    ▷ Initialize short-term component.
    $\boldsymbol{F}^{(0)} = \mathbf{0}$
    ▷ Inner-loop learning via Hebbian STP ≡ Recurrent model receiving an input sequence.
    for $t = 1$ to $T$ do
        ▷ Obtain total efficacies.
        $\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$
        ▷ Normalize total efficacies and their short-term components (optional but practically impactful).
        $\hat{\boldsymbol{G}}^{(t)} = \boldsymbol{G}^{(t)} / \|\boldsymbol{G}^{(t)}\|$
        $\hat{\boldsymbol{F}}^{(t)} = \boldsymbol{F}^{(t)} / \|\boldsymbol{G}^{(t)}\|$
        ▷ Forward pass.
        $\boldsymbol{h}^{(t)} = \sigma(\hat{\boldsymbol{G}}^{(t)} \boldsymbol{x}^{(t)})$
        ▷ Hebbian STP update ≡ Synaptic-recurrent-state update.
        $\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \hat{\boldsymbol{F}}^{(t)}$
    end for
    ▷ Meta-learn via gradient update, e.g. BPTT.
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \nabla \mathcal{L}$
end while
```

A.2. Importance of learning per-synapse STP parameters

Table 1 shows that learning per-synapse STP parameters achieves greater proficiency and efficiency. This happens for both STPNf and STPNr, respectively without and with recurrent connections. Note energy efficiency being lower for purely feed-forward models, as happens for ART, can be expected. This is because recurrent activations are dense, whereas input for these problems is more sparse and lower magnitude on average, which we do not correct for in these measurements. All models had their hidden size chosen to have similar number of parameters.

Table 1. Inference-time proficiency and energy efficiency of per-synapse and uniform STP models on ART and Maze Exploration

	ART		Maze	
Model	Test accuracy	Power consumption	Reward	Power consumption
STPNr per-synapse STP (Ours)	**99.99** $\pm$ **0.01**	3.4 $\pm$ 0.4	**115.7** $\pm$ **1.6**	**80.2** $\pm$ **2.4**
STPNf per-synapse STP (Ours)	89.40 $\pm$ 5.14	**1.2** $\pm$ **0.1**	112.9 $\pm$ 1.5	154.8 $\pm$ 6.3
STPNr uniform STP (Ours)	96.89 $\pm$ 6.21	4.0 $\pm$ 0.3	74.0 $\pm$ 1.4	150.2 $\pm$ 5.5
STPNf uniform STP (Ours)	60.21 $\pm$ 0.52	2.6 $\pm$ 0.1	85.2 $\pm$ 1.6	355.6 $\pm$ 12.9
HebbFF (Tyulmankov et al., 2022)	10.62 $\pm$ 0.02	7.9 $\pm$ 1.8	39.9 $\pm$ 1.3	385.9 $\pm$ 8.9

ART and Maze Exploration tasks were presented in other works presenting different networks with plasticity, so they serve as solid baselines for STPN without an explicit task bias introduced by our work. Nonetheless, HebbFF's performance on these two tasks is far below the structurally equivalent STPNf. Therefore, we decide to test the importance of recurrence and per-synapse STP in the Continual Familiarity Detection Task presented in (Tyulmankov et al., 2022). In this task, at each timestep the network is presented with a binary vector whose elements are 1 or -1. With probability $1 - p$, this vector is randomly sampled. Otherwise, with probability p, the vector provided to the network R timesteps ago is used as input again. One exception is that, if an already repeated input was sampled to be repeated again, a new vector is generated instead. The network outputs a single bit at every timestep, representing whether it predicts the vector presented is repeated or not.

We report results on the same three variations of this task presented in Tyulmankov et al. (2022), running 5 seeds for each variation of the task. In the first variation, a single dataset of random vectors is used for all iterations of training ('dataset' mode). In the following two, a new set of vectors is generated at each training iteration ('infinite' mode). The distance between repeated vectors (R) is also varied from 3 to 6 in the last two experiments. We use T=5000 vectors per iteration (as the experiment logs in the public repository for Tyulmankov et al. (2022)), and train for the same number of epochs as the released models (Tyulmankov et al., 2022) (although it is not clear whether these were originally trained until convergence). All other training configurations are exactly the same as found in their publicly available code. We choose the hidden size of all other variants as to have similar number of parameters to HebbFF. Given HebbFF has a much higher memory consumption for the same number of trainable parameters than per-synapse STPNr, the STPNr has nearly half the number of hidden units in the Maze exploration for these comparison results as compared to those shown in Figs. 2 and 3, which explains the lower performance.

Table 2 shows the generalization accuracy for the three tasks we described. For the 'dataset' mode, this means accuracy on a validation set. For the 'infinite' mode, accuracy is measured on a newly generated dataset at each iteration, which we label as 'train accuracy' although the network hasn't seen these specific examples before (hence generalization). The results confirm that STPN, especially in its per-synapse variants, performs competitively with HebbFF. Note that we did not tune any of the STPN variants for this task at all, whereas we can HebbFF to be well tuned. STPN is clearly stronger in the infinite mode with R=3, which happens to be the one with the shortest training iterations (1600, vs 3000 for 'dataset' mode and 3500 for the other 'infinite' mode task). Possibly training STPN until convergence would have led to more competitive results, in addition to tuning. Another reason for which HebbFF is better suited for this task is the domain restriction of STP parameters for HebbFF (e.g. only anti-Hebbian and short-term behavior) can be specially suited to this task but not generalize to more complex problems. Furthermore, the plastic update stabilization and initialization we use for STPN might be better suited for more complex problems. This results confirm the use of recurrence and per-synapse STP as a better general approach to configuring STPN.

Table 2. Generalization proficiency of per-synapse and uniform STP models and LSTM in three variants of the Continual Familiarity Detection Task (Tyulmankov et al., 2022)

Model	Dataset mode (R=3)	Infinite mode (R=3)	Infinite mode (R=6)
STPNr per-synapse STP (Ours)	**97.53 ± 0.31**	**99.99 ± 0.02**	98.98 ± 0.28
STPNf per-synapse STP (Ours)	**98.16 ± 0.91**	99.92 ± 0.04	98.72 ± 0.19
STPNr uniform STP (Ours)	78.52 ± 12.82	94.13 ± 7.59	90.88 ± 3.00
STPNf uniform STP (Ours)	81.08 ± 14.27	91.95 ± 9.45	91.50 ± 11.67
HebbFF (Tyulmankov et al., 2022)	**98.47 ± 0.87**	99.68 ± 0.09	**99.71 ± 0.12**
LSTM	66.75 ± 0.39	78.52 ± 2.07	74.37 ± 2.90

A.3. Further results

For completeness, we present the inference-time equivalents of Figs. 2 and 3 in tabular form, which can be found in Tables 3 and 4. STPN has the added benefit of being able to adapt its synapses after training and hence having a better adaptation during inference, which clearly provides an added extra performance at test time.

A.4. Training methods

Stabilization through a type of weight normalization. Plastic weights can significantly impact the predictions of a network based on recent experience. Although significant changes to the effective weights can increase predictive performance, fast changes of connectionist parameters have the potential to cause a variety of issues during training and evaluation. For instance, Ba et al. (2016) found that large or small norms in hidden vectors can cause the fast weights derived from them to grow or shrink excessively, leading to exploding or vanishing gradients. In their case, using layer normalization of hidden states, and performing a decay of fast weights, successfully controls such instabilities. Miconi et al.

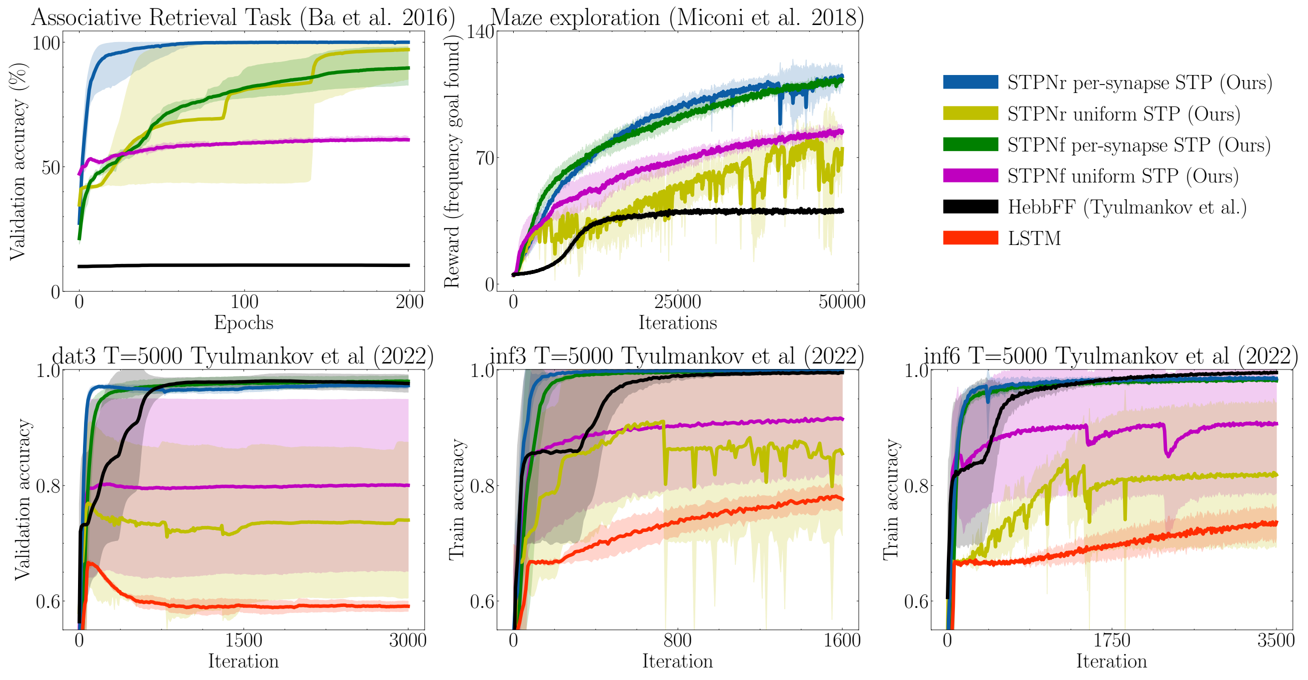

Figure 5. Generalization proficiency curves during training of per-synapse and uniform STP models and LSTM, in ART, Maze Exploration and three variants of the Continual Familiarity Detection Task (Tyulmankov et al., 2022)

Table 3. Accuracy and reward of trained models during inference. Averaged over multiple evaluating runs, reported mean and standard deviation of different random seeds.

Model	Associative Retrieval	Maze Exploration	Pong	Inverted Pendulum
STPN (Ours)	**98.55** $\pm$ **2.76**	**132.7** $\pm$ **1.4**	**20.7** $\pm$ **0.5**	**985.0** $\pm$ **15.0**
LSTM	47.28 $\pm$ 3.16	119.8 $\pm$ 2.1	18.0 $\pm$ 4.6	23.2 $\pm$ 10.1
RNN	46.83 $\pm$ 5.56	97.9 $\pm$ 1.9	-	143.0 $\pm$ 22.4
Modulated Plasticity	61.49 $\pm$ 15.79	112.2 $\pm$ 2.0	-	-
Fast Weights	80.87 $\pm$ 1.67	-	-	-
MLP	-	-	-	701.8 $\pm$ 98.9

Table 4. Power consumption of trained models during inference. Average over multiple inference runs and timesteps within a run, reported mean and standard deviation over seeds.

Model	Associative Retrieval	Maze Exploration	Pong	Inverted Pendulum
STPN (Ours)	**10.9** $\pm$ **3.8**	**181.1** $\pm$ **2.9**	**52.7** $\pm$ **6.4**	**7.9** $\pm$ **2.5**
LSTM	65.6 $\pm$ 3.4	649.1 $\pm$ 15.3	576.8 $\pm$ 33.5	758.8 $\pm$ 81.7
RNN	43.0 $\pm$ 4.9	330.6 $\pm$ 24.3	-	143.0 $\pm$ 22.4
Modulated plasticity	32.0 $\pm$ 7.4	231.6 $\pm$ 23.9	-	-
Fast Weights	80.6 $\pm$ 69.4	-	-	-
MLP	-	-	-	115.8 $\pm$ 33.9

(2019) cites the inherent instability of Hebbian updates as the reason for inclusion of a clipping mechanism for synaptic memories. Alternatively, it also experiments with an implicit normalization (by using Oja's rule instead of Hebb's rule for the plastic weight updates), or a decay term (not-trained, a function of the 'fast learning rate' which is uniform for all synapses); ultimately choosing the clipping due to superior empirical results.

In our case, layer normalization of hidden inputs alone is not enough to counteract hidden states of varying norm, as input also forms part of the Hebbian update. Additionally, the gradients of STP parameters λ and γ, which are trainable in the STPN, suffer from further instability as these parameters are connected to the computational graph only through the updating of $\boldsymbol{F}$. Relying on decaying $\boldsymbol{F}$ is also not an option, given we do not to restrict λ to cause a decay of synaptic memories (as we found restricting it to hinder proficiency). In fact, in some cases, some synapses learn to potentiate their current memories. This adds to the issue of growing memories, especially if the updates to synaptic memories happen to also be purely additive. We experiment with multiple of these techniques, on top of explicit normalization of weights; which ultimately turns out to be the best option in most cases. This weight normalization is performed on the effective weights for each batch element at each timestep, which also normalizes the synaptic memories. Note however we store the un-normalized long-term weights, as these are common to all batch elements and an average norm across the batch defeats the purpose of sequence specific weight adaptation. Additionally, this allows to implicitly learn the relative norm between long-term and short-term weights, given no scaling of the normalized weights is learned as in standard weight normalization (Salimans & Kingma, 2016). We performed normalization per neuron, although similar performance was observed for global normalization and could be further explored. Additional learnable norm scaling is left for future work. To confirm weight normalization did not provide an inherent training improvement for all models (but only for STPN due to the previously cited reasons), we trained all baselines in ART and Maze with equivalent non-parametric weight normalization schemes, and find neither proficiency or efficiency are improved for other baselines.

Initialization strategy of STP parameters. Initialization of STP parameters λ and γ can also have a significant impact on convergence speed. We tested different random initialization distributions in ART, where we find the best scheme that we use in all other experiments. We find initializing $\gamma_{ij} \sim \mathcal{U}(-\frac{0.001}{\sqrt{h}}, \frac{0.001}{\sqrt{h}})$ is better over purely positive, centered away from zero, or more largely spread values. We hypothesize small initial values of γ allow long-term weights to have a greater impact on very early training as short-term weights are very weakly updated. Surprisingly, a much wider and purely positive initialization $\lambda_{ij} \sim \mathcal{U}(0, 1)$ is empirically better. Presumably, decay doesn't play a major role at very early stages of training when γ is very close to zero, and in later stages of learning when plastic weights start playing a role, having large variety of dynamics across synapses and neurons allows for richer evolution of synaptic memories.

A.5. Experimental details

We run a different number of seeds across experiments, running more when either or both experiment was not highly computationally expensive or more instances were necessary to make clearer conclusions. Specifically, we used 5 (ART and Pong), 3 (Maze) and 2 (Pendulum) seeds respectively.

A.5.1. Associative Retrieval

Compared to the described setup in Ba et al. (2016), a modification in the experiments carried out is we do not include an embedding or (post-RNN,) pre-SoftMax fully connected layer, of size 100 each. The motivation behind such a choice is three fold: a) Reducing the network to only a RNN and a SoftMax layer makes the predictive capacity of the Network rely maximally on the RNN, which is the true subject of study b) the second author admits to the lack of need for such embedding (Hinton, 2017), and attributes its inclusion to reasons beyond the scope of such paper, and c) when tuning the learning rate we find all models (including non-plastic baselines with small hidden units) easily solve the task if an embedding layer is included, which we hypothesize is caused by such layer taking a major role in prediction.

We tune the learning rate (to 0.001), which is not provided in Ba et al. (2016). The number of training epochs for the results reported in (Ba et al., 2016) is not provided either. We experiment with some values and report results on networks trained for 200 epochs, after which we find networks generally do not improve. Hidden sizes of networks are chosen so that a similar number of parameters are trained, with a focus on low-parameter regime as larger networks all solve the problem (also reported in Ba et al. (2016)). Therefore we choose a hidden size of 20 (the smallest tested size in Ba et al. (2016)) for the Fast weights RNN, and calculate the hidden size of other models as to obtain a similar number of parameters of the entire network. For RNN with fast weights, we use layer normalization, only one iteration of the inner loop, and λ and η, all as indicated in the Appendix A.1 in Ba et al. (2016). Although the network structure is different, we don't change the STP parameters (λ and η) as they the same values are used throughout the different tasks in the paper due to alleged robustness provided by layer normalization. Our experiments tuning fast weights RNN confirms this, as they show little difference when exploring other values. Additionally, we use hyperbolic tangent as activation function, for fairer comparison to other networks and as our experiments show a better performance than with ReLU (which only provides purely additive Hebbian factors). Initialization of recurrent (slow) weights as an identity matrix is know to favor longer (orthogonal) storing of hidden states in RNNs (Hinton, 2017) and hence don't offer a specific advantage to RNN with fast weights vs other RNNs, so the results shown in this paper don't follow such specific initialization for recurrent connections.

A.5.2. Maze Exploration

We use the same parameters provided in (Miconi et al., 2019), and we run the experiments by adapting the open source repository (hence also using the parameters not explicitly mentioned in the article but introduced in the code). We only increase the batch size (number of agents 'acting in parallel') from 16 (in the code, not mentioned in the article) to 512 to maximize computational efficiency of gradient updates. Besides faster learning in a shorter amount of iterations due to more experience per gradient update, we find it can reduce variance of the update as in some episodes initial exploration for the reward fails or comes too late in the episode, hence the agent not being able to exploit learning the location of the reward. Regarding the hidden sizes of the RNNs, we choose a hidden size of 100 for the standard RNN (as provided in the standard Miconi et al. (2019) parameters), and calculate the hidden size of other models as to obtain a similar number of parameters for the entire network. Additionally, following Miconi et al. (2019), value and action branches are non plastic for all models, only fixed and feed forward.

A.5.3. Atari Pong

We use RLLib (Liang et al., 2018) to train and evaluate agents in PongNoFrameskip-v4. Besides the experimental choices described in Section 3.4, it should be noted that we use STPNr with 64 hidden units, and adjust the size of the LSTM to 48 hidden units in order to have a similar number of trainable parameters. We also follow Mnih et al. (2016) in the preprocessing (dimensionality and color scale) for the game frames, besides frame stacking. We tune rollout length (50), gradient clipping (40), discount factor (0.99) in shorter runs (which both models share in the displayed results); and additionally tune initial learning rate for the final longer runs (0.0007 and 0.0001 respectively), using a linear decay learning rate schedule finishing at 10^{-11} at 200 million iterations. Models are trained from the experienced collected by 64 parallel agents.

A.5.4. MuJoCo Inverted Pendulum

We use RLLib to train and evaluate agents in InvertedPendulum-v2. We mostly do so with the same parameters reported in (Schulman et al., 2017), and for any other parameters not specified there we use the default options provided by RLLib. The only differences are: a) the use of a linearly decaying learning rate schedule, starting from the same original learning rate and finishing at 4 million timesteps at 10^{-11}; b) since it's not specified in (Schulman et al., 2017), we employ a single parallel worker collecting experience in one environment c) the choice of a single layer policy network (besides de action and value branch) of size 64 (this is the only case where models' hidden sizes were not adjusted to obtain a similar number of parameters, as it didn't prove relevant in our experiments).

A.6. Terminology on timescales

An explicit note about the different timescales across experiments might ease the interpretation of the reported results by the reader, and hence we now provide it. In the supervised learning task, namely ART, an epoch represents a training iteration in which the entire training dataset is seen, and sequence elements describe each of the characters presented to the network in a sequential manner. In reinforcement learning, where an agent interacts with an environment via actions, and receives rewards and observations; we refer to each of these interactions as an episode step or timestep. An episode encompasses a variable (Pong, Inverted Pendulum) or fixed (Maze) number of episode steps or timesteps. Finally, although iteration could also refer to a single timestep, we use such term to indicate one period between gradient updates, which can enclose multiple episodes experienced by multiple agents, depending on the learning algorithm.

A.7. STPN mechanics

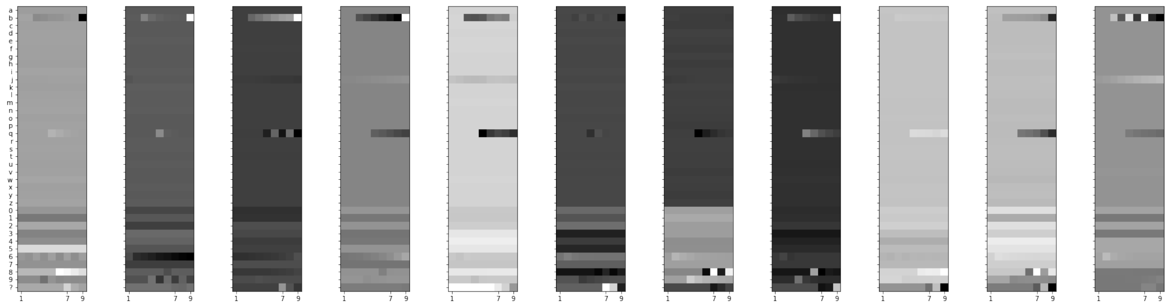

Figure 6. Evolution of effective weights through processing of a sequence in ART. Each separate plot represents a neuron, where the y axis are the input synapses for such neuron, and the x axis represents time, i.e. sequence elements. Color represents the magnitude of G_{ij} for each synapse. We can observe the effect of training λ_{ij} as different synapses increase or decrease in value at different rates

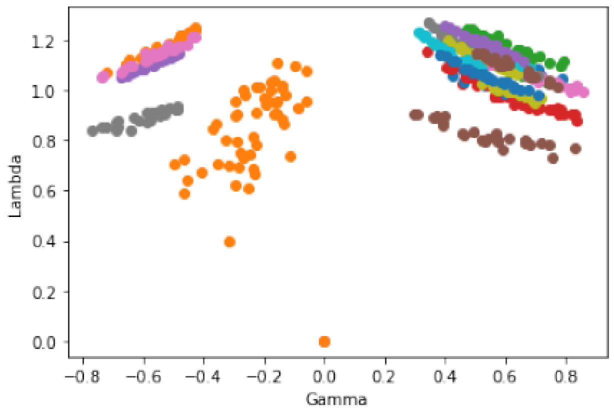

Figure 7. Learnt STP parameters for STPN in ART. Each dot represents one synapse, each color represents the synapses of a single neuron. There is clearly some clustering of the parameters for each neuron at a specific sign for γ, but with different learning rates for different synapses.