Learning to Control Rapidly Changing Synaptic Connections: An Alternative Type of Memory in Sequence Processing Artificial Neural Networks

Kazuki Irie[1] **Jürgen Schmidhuber**[1,2]
[1]The Swiss AI Lab, IDSIA, USI & SUPSI, Lugano, Switzerland
[2]AI Initiative, KAUST, Thuwal, Saudi Arabia
{kazuki, juergen}@idsia.ch

Abstract

Short-term memory in standard, general-purpose, sequence-processing recurrent neural networks (RNNs) is stored as activations of nodes or "neurons." Generalising feedforward NNs (FNNs) to such RNNs is mathematically straightforward and natural, and even historical: already in 1943, McCulloch and Pitts proposed this as a surrogate to "synaptic modifications" (in effect, generalising the Lenz-Ising model, the first non-sequence processing RNN architecture of the 1920s). A lesser known alternative approach to storing short-term memory in "synaptic connections"—by parameterising and controlling the dynamics of a context-sensitive time-varying weight matrix through another NN—yields another "natural" type of short-term memory in sequence processing NNs: the Fast Weight Programmers (FWPs) of the early 1990s. FWPs have seen a recent revival as generic sequence processors, achieving competitive performance across various tasks. They are formally closely related to the now popular Transformers. Here we present them in the context of artificial NNs as an abstraction of biological NNs—a perspective that has not been stressed enough in previous FWP work. We first review aspects of FWPs for pedagogical purposes, then discuss connections to related works motivated by insights from neuroscience.

1 Introduction

Memory is essential for problem solving in the natural world in which presently available information may become important later. In the context of artificial neural networks (NNs), we may distinguish two major categories of memory: long-term (LTM) and short-term memory (STM). For example, a supervised feedforward NN (FNN) learning to solve some prediction task first sees a sequence of (batches of) training examples, then test examples. During the training phase, its weight matrix (WM) is iteratively modified by some learning algorithm to reduce the NN's errors. In the conventional setting, the WM is frozen once training ends, and becomes a permanent form of memory (LTM) reflecting its training experience, repeatedly used to make predictions about unseen test examples. This LTM stored in the WM is the NN's program, where the NN architecture is considered a computer [1].

When the world is only partially observable, problem-solving NNs generally need additional *short-term memory* (STM), to temporally store information about previously observed inputs. STM is typically specific to the current task, to be erased when the next task starts. A general and standard sequence processing NN is the recurrent NN (RNN; [2, 3], see also older works reviewed in Sec. 3). While the transition from an FNN to an RNN is straightforward (reviewed in Sec. 2.1), it is worth noting the underlying decision of storing STM as activations of "neurons" while keeping the WM, i.e., the "synaptic connection weight strengths" fixed. Here our main goal is to shed light on an alternative

Workshop on Memory in Artificial and Real Intelligence (MemARI), NeurIPS 2022, New Orleans, LA, USA.

way of introducing STM into the FNN: the Fast Weight Programmers (FWPs; [4, 5]) which learn to store STM in synaptic connection weights.

2 Short-Term Memory in Sequence Processing Nets

2.1 Conventional Approach: Storing Short-Term Memory in Neurons

Before presenting the alternative memory type in Sec. 2.2, we first review "conventional" STM in sequence processing NNs. In what follows, let d_{in}, d_{out}, and t denote positive integers. We define a layer of an NN as a function with a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ which transforms an input vector $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{in}}}$ to an output vector $\boldsymbol{y}_t \in \mathbb{R}^{d_{\text{out}}}$ as

$$\boldsymbol{y}_t = \sigma(\boldsymbol{W}\boldsymbol{x}_t) \tag{1}$$

where σ is an element-wise activation function. We omit the additive bias term without loss of generality. This is indeed an abstraction for an NN where $\boldsymbol{x}$ and $\boldsymbol{y}$ represent the activities of input and output "neurons" respectively, and the weight matrix $\boldsymbol{W}$ stores strengths of "synaptic connections" between them.[1] This system has no STM: when we feed another input $\boldsymbol{x}_{t+1}$ at the next time step $t+1$, the previous input $\boldsymbol{x}_t$ has no influence on the computation of output $\boldsymbol{y}_{t+1}$. This *feedforward* layer can be straightforwardly extended through an STM by making the output $\boldsymbol{y}_t$ at step t dependent on its own output $\boldsymbol{y}_{t-1}$ from the previous time step $t-1$ by introducing another weight matrix $\boldsymbol{R} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$

$$\boldsymbol{y}_t = \sigma(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{R}\boldsymbol{y}_{t-1}) \tag{2}$$

This is a trivial and natural way of introducing STM to ANNs for sequence processing[2]. Essentially we model the dynamics of the output neurons $\boldsymbol{y}_t$ over time by another fictive NN with synaptic connections $\boldsymbol{R}$ between "neurons at time t" and "neurons at time $t-1$". Since no such synaptic connections over time exist physically, we may be implicitly giving up the analogy to biological NNs here. It is also worth noting that, while Jordan [2] and Elman [3] propose this kind of recurrence purely for the purpose of introducing temporal dependency, work by McCulloch and Pitts [6] in 1943 already proposed it as a replacement of modifiable synaptic connections—see *"Theorem 7. Alterable synapses can be replaced by circles."* Now what if we modelled the "*alterable synapses*" directly? The alternative memory type presented in the next section represents an answer to this question.

2.2 Alternative Approach: Storing Short-Term Memory in Synaptic Connections

Restarting from Eq. 1, we now show how to introduce STM to FNNs by explicitly and rapidly modifying the synaptic weights as a function of the inputs. The core idea is to control the mechanism of synaptic weight changes by another learning NN. Similarly to the recurrent connections $\boldsymbol{R}$ introduced in the standard RNN (Sec. 2.1), this NN may be "fictive" from the perspective of the biological NN analogy. We present a bottom-up step-by-step construction as follows. Starting from the goal, we want $\boldsymbol{W}$ to be parameterised as a function of the inputs, i.e., we want $\boldsymbol{W}_{t-1}$ to become $\boldsymbol{W}_t$ at each step t. This requires an update rule. For now, let us take a very simple update equation:

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} + \boldsymbol{v}_t \otimes \boldsymbol{k}_t \tag{3}$$

where $\otimes$ denotes outer product $\boldsymbol{v}_t \otimes \boldsymbol{k}_t \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ between two vectors $\boldsymbol{v}_t \in \mathbb{R}^{d_{\text{out}}}$ and $\boldsymbol{k}_t \in \mathbb{R}^{d_{\text{in}}}$. This is a generic equation akin to Hebb's informal learning rule [7] ($\boldsymbol{k}_t$ is the input and $\boldsymbol{v}_t$ the output of this layer) as well as to the gradient descent update rule of the WM of a linear layer ($\boldsymbol{k}_t$ as the input and $\boldsymbol{v}_t$ the gradient of the loss w.r.t. the linear layer's output scaled by a negative learning rate). Now, these newly introduced variables $\boldsymbol{k}_t$ and $\boldsymbol{v}_t$ have to be generated somehow. We can simply generate them from the actual external input $\boldsymbol{x}_t$, i.e.,

$$[\boldsymbol{k}_t, \boldsymbol{v}_t] = \boldsymbol{A}\boldsymbol{x}_t \tag{4}$$

where $\boldsymbol{A} \in \mathbb{R}^{(d_{\text{in}}+d_{\text{out}}) \times d_{\text{in}}}$ is a weight matrix, and the square brackets denote vector concatenation. Putting these operations in the right order yields a sequence processor that, just like the standard

[1] Generally speaking, the analogy never goes beyond this, and has limited (if any) practical implications or benefits.

[2] See, e.g., Elman [3]'s simplification of Jordan [2]'s version featuring one more layer before the recurrence.

RNN, transforms an input $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{in}}}$ into an output $\boldsymbol{y}_t \in \mathbb{R}^{d_{\text{out}}}$ at each time step t as follows

$$[\boldsymbol{k}_t, \boldsymbol{v}_t] = \boldsymbol{A}\boldsymbol{x}_t \tag{5}$$

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} + \boldsymbol{v}_t \otimes \boldsymbol{k}_t \tag{6}$$

$$\boldsymbol{y}_t = \sigma(\boldsymbol{W}_t \boldsymbol{x}_t) \tag{7}$$

Essentially, an NN with parameters $\boldsymbol{A} \in \mathbb{R}^{(d_{\text{in}}+d_{\text{out}}) \times d_{\text{in}}}$ translates the input observations into weight changes through Eqs. 5-6. This effectively augments Eq. 1 of an FNN with STM that is stored in the context-sensitive weight matrix $\boldsymbol{W}_t$ representing the dynamic synaptic weights. This idea is what was introduced by Schmidhuber [4, 5] as *"an alternative to recurrent nets"*[3]. The original work presents this model as a system of two networks: the network in Eq. 7 is the *fast net*, whose *fast weights* $\boldsymbol{W}_t$ are modified at every time step by the *slow net* of Eq. 5 whose *slow weights* $\boldsymbol{A}$ are trained e.g, by backpropagation through time. In the original paper [4, 5], the slow net is said to "control" the fast weights (and thus, it is a "fast weight controller"), while recent works refer to it as a "Fast Weight Programmer" (FWP) [8]; considering the WM as the program of an NN [1]. Modifying the WM in a goal-oriented manner means programming the NN via fast weight changes.

Practical Enhancements. Two technical enhancements have improved FWPs in practice. Firstly, Transformers with linearised self-attention [9, 10] have a "dual form" that is an outer product-based FWP: exactly the same mathematical relation [11] maps the perceptron to its dual form, the kernel machine [12]. A practical implication is that FWPs can directly adopt advancements of architectural designs originally developed for Transformers [13], as FWPs implement their (linearised) self-attention layers. These designs include an additional *query* projection that generates the input to the fast net, as well as the now standard two-layer feedforward blocks and layer normalisation. In fact, it is now also common to use the associative memory terminology "key/value" to describe FWPs (as reflected in the notations $\boldsymbol{k}_t$ and $\boldsymbol{v}_t$ above), rather than the original "FROM/TO" terminology of the 1990s.

The second kind of enhancement is an improved update rule. While the "purely additive" update rule of Eq. 3 is the one used in the standard Linear Transformer [9], Schlag et al. [8] identify its memory capacity problem and proposes to replace it by the delta learning rule [14, 15], which has been shown to consistently outperform the purely additive one on various tasks including language modelling [8], algorithmic tasks [16], time series prediction [17], image generation [18], and video game playing in reinforcement learning [16].

Properties. The incremental/additive nature of STM updates in FWPs (Eq. 6) yields interesting properties that the standard RNN does not have. Firstly, it provides good gradient flow [19] and alleviates the fundamental vanishing gradient problem [20] of sequence learning through gradient descent. Second, unlike in standard RNNs, the weight update equation (e.g., Eq. 6) can be directly seen as an Euler discretisation of its continuous-time counterpart. This can be exploited to derive a powerful Neural-ODE/CDE based sequence processors for the continuous-time domain [17].

Another important property not shared by standard RNNs is the number of temporal variables, one of the original motivations of FWPs [21]. Assuming $d = d_{\text{in}} = d_{\text{out}}$, FWPs store $O(d^2)$ memory instead of $O(d)$ in the case of RNNs. While this may increase the memory storage, it also increases the space complexity of FWPs trained by backpropagation through time. A naive implementation has a space complexity of $O(Td^2)$ where T is the number of backpropagation steps. Fortunately, the incremental/additive nature of memory allows for deriving memory-efficient backpropagation that reduces this complexity to $O(d^2)$ (see e.g., [9, 8, 22]), allowing for scaling up practical applications.

Other Extensions. There is no reason to restrict the STM storage to either "neurons" or "synaptic weights." We can use both: one can parameterise weight matrices in the standard RNN (Eq. 2) as a function of inputs using the FWP principle of Sec. 2.2. Alternatively, one can also introduce recurrent connections to the FWP that connect activations of the fast net from the previous time step to the input of the slow net. These hybrid approaches [16] have been shown to improve over other FWPs without recurrence on various tasks.

Another natural extension is to make all weights context-dependent, including the slow weights in the FWP themselves. Such a model can be obtained with minor modifications to Eqs. 5-7 [22]. These

[3] Up to slight changes introduced for pedagogical purposes: we placed an activation function in Eq. 7 instead of Eq. 6 to facilitate direct comparison to the original FNN of Eq. 1.

self-referential variants [23, 24, 25] allow an explicit formulation of recursive self-modifications in artificial NNs, another interesting perspective directly derived from the FWP principle.

3 Relating to Works Motivated by Neuroscience

The importance of rapidly changing NN weights was first explicitly emphasised by von der Malsburg [26], then also by other authors Feldman [27], Hinton and Plaut [28]. The parameterisation and control of synapses through another NN, however, was introduced only in the early 1990s by Schmidhuber [4, 5]. There are also several rather recent works motivated by insights from neuroscience to improve "synapses" in artificial NNs. For example, Lahiri and Ganguli [29] and Zenke et al. [30] question the single-scalar parameterisation of synapses in artificial NNs in light of the complexity of biological synapses. The core idea of FWPs directly relates to this spirit, as FWPs parameterise synapses by a general purpose NN that can potentially control any complex dynamics. The models of Miconi et al. [31, 32] motivated by synaptic plasticity in the brain (see also [33]) are FWP variants where plasticity is introduced as augmentation to some base RNN (in the spirit of the "hybrid approaches" mentioned above in Sec. 2.2 "Other Extensions"). Here we stress that such synaptic weight modifications can be a stand-alone mechanism for STM. In machine learning, dynamic synapses are also motivated by applications in meta-learning (e.g., [34, 35, 36, 37, 38]) and continual learning (e.g., [30, 39, 40]).

The work of Whittington et al. [41] implicitly relates a model of hippocampal formation (Tolman-Eichenbaum Machine; TEM [42]) to Transformers without softmax in self-attention, which relate to the so-called Linear Transformers [9], which are FWPs [8]. Specifically, TEM (simplified with some reasonable assumptions [41]) is an auto-regressive sequence processor that, at each time step $t > 0$, estimates the next sensory input $\boldsymbol{y}_t \in \mathbb{R}^{d_{\text{in}}}$ as a function of the current sensory input $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{in}}}$, the action represented by an integer a_t, and the positional representation $\boldsymbol{g}_t \in \mathbb{R}^{d_{\text{pos}}}$ (where d_{pos} is a positive integer) corresponding to activities of "grid cells" (or "medial entorhical cells"). $\boldsymbol{g}_t$ is parameterised by an RNN without external inputs, with an action-dependent weight matrix $\mathbf{W}_{a_t} = f(a_t) \in \mathbb{R}^{d_{\text{pos}} \times d_{\text{pos}}}$ where f is a parameterised feedforward NN as follows:[4]

$$\boldsymbol{g}_t = \sigma(\mathbf{W}_{a_t} \boldsymbol{g}_{t-1}) \tag{8}$$

$$\boldsymbol{y}_t = \alpha \boldsymbol{X}_t \boldsymbol{G}_t^\intercal \boldsymbol{g}_t \tag{9}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{d_{\text{in}} \times t}$ and $\boldsymbol{G}_t \in \mathbb{R}^{d_{\text{pos}} \times t}$ are matrices constructed by concatenating the corresponding vectors from the previous steps, i.e., $\boldsymbol{X}_t = [\boldsymbol{x}_1, ..., \boldsymbol{x}_t]$ and $\boldsymbol{G}_t = [\boldsymbol{g}_1, ..., \boldsymbol{g}_t]$, and $\alpha \in \mathbb{R}$. Clearly, this computation can be expressed as an NN (linear layer) with context-dependent synaptic connections $\boldsymbol{W}_t \in \mathbb{R}^{d_{\text{in}} \times d_{\text{pos}}}$ (with $\boldsymbol{W}_0 = 0$) that evolve over time in FWP fashion as follows:

$$\boldsymbol{W}_t = \alpha \boldsymbol{X}_t \boldsymbol{G}_t^\intercal = \boldsymbol{W}_{t-1} + \alpha \boldsymbol{x}_t \otimes \boldsymbol{g}_t \tag{10}$$

$$\boldsymbol{y}_t = \boldsymbol{W}_t \boldsymbol{g}_t \tag{11}$$

Unlike in the original formulation of TEM, the activities of "place cells" do not directly appear in this FWP formulation (they are hidden in the computation of dynamic synaptic connection weights). We also note that this connection is not surprising, since the motivation of TEM [42] ("structural generalisation") is a form of systematic generalisation [43], and FWPs relate to tensor product representations that are popular in the context of systematic generalisation [44, 45].

The general principle of storing patterns in LTM based on changing synaptic connections is a much older concept already found in non-learning (and non-sequence processing) RNNs. In the 1970s Amari [46] extended the Ising or Lenz-Ising model introduced in the 1920s [47, 48, 49, 50, 51] (where a binary neuron is analogous to a spin), by making it adaptive and thus capable of learning associations of input/output patterns by changing the connection weights. This precedes the work of Little [52] and Hopfield [53] (see also [54]) on what has been called the "Hopfield Network" or Amari-Hopfield Network [55] that has also been recently revisited [56, 57, 58]. Amari [46] even discusses the sequence processing scenario.

As mentioned above, the principles of standard RNNs were also discussed by neuroscientists McCulloch and Pitts [6] in 1943, and analysed by Kleene et al. [59] in the 1950s. Turing [60]'s unpublished "unorganized machines" of 1948 also relate to RNNs.

[4] We introduce two notational changes compared to Whittington et al. [41]. First, for consistency with the rest of this paper, we use column-major vectors. Second, for clarity, we introduce time indices to all variables. We also omit many projection layers including extra layer after $\boldsymbol{y}_t$ as well as some additional residual connections that are in the official implementation.

4 Conclusion

We discussed Fast Weight Programmers (FWPs) as an alternative to standard RNNs or to McCulloch and Pitts' model with "cycles" that replace "alterable synapses" (see their informal Theorem 7). This interesting perspective has not been stressed much in previous work on FWPs. While the exact role of synaptic plasticity on memory in the brain remains unclear (see ongoing discussions of this topic in neuroscience, e.g., [61, 62]), we saw how compactly and elegantly FWPs can implement the learnable dynamics of short-term memory in time-varying synaptic connections of artificial NNs.

Acknowledgements

This research was partially funded by ERC Advanced grant no: 742870, project AlgoRNN, and by Swiss National Science Foundation grant no: 200021_192356, project NEUSYM.

References

[1] Jürgen Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. *Institut für Informatik, Technische Universität München. Technical Report FKI-126*, 90, 1990.

[2] Michael I Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. Conf. of the Cognitive Science Society*, pages 531–546. Amherst, MA, USA, August 1986.

[3] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[4] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. Technical Report FKI-147-91, Institut für Informatik, Technische Universität München, March 1991.

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

[6] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[7] Donald Olding Hebb. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

[8] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers are secretly fast weight programmers. In *Proc. Int. Conf. on Machine Learning (ICML)*, Virtual only, July 2021.

[9] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. Int. Conf. on Machine Learning (ICML)*, Virtual only, July 2020.

[10] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, 2021.

[11] Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *Proc. Int. Conf. on Machine Learning (ICML)*, Baltimore, MD, USA, July 2022.

[12] Mark A. Aizerman, Emmanuil M. Braverman, and Lev I. Rozonoer. Theoretical foundations of potential function method in pattern recognition. *Automation and Remote Control*, 25(6): 917–936, 1964.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, CA, USA, December 2017.

[14] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. In *Proc. IRE WESCON Convention Record*, pages 96–104, Los Angeles, CA, USA, August 1960.

[15] Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, May 2021.

[16] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Virtual only, December 2021.

[17] Kazuki Irie, Francesco Faccio, and Jürgen Schmidhuber. Neural differential equations for learning to program neural nets through continuous learning rules. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, USA, December 2022.

[18] Kazuki Irie and Jürgen Schmidhuber. Images as weight matrices: Sequential image generation through synaptic learning rules. *Preprint arXiv:2210.06184*, 2022.

[19] Juergen Schmidhuber. 26 March 1991: Neural nets learn to program neural nets with fast weights—like today's Transformer variants. 2021: New stuff!, AI Blog, 2021. URL `https://people.idsia.ch/~juergen/fast-weight-programmer-1991-transformer.html`.

[20] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma thesis, Technische Universität München*, 1991.

[21] Jürgen Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *International Conference on Artificial Neural Networks (ICANN)*, pages 460–463, Amsterdam, Netherlands, September 1993.

[22] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 9660–9677, Baltimore, MA, USA, July 2022.

[23] Jürgen Schmidhuber. Steps towards "self-referential" learning. Technical Report CU-CS-627-92, Dept. of Comp. Sci., University of Colorado at Boulder, November 1992.

[24] Jürgen Schmidhuber. A self-referential weight matrix. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, pages 446–451, Amsterdam, Netherlands, September 1993.

[25] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[26] Christoph von der Malsburg. The correlation theory of brain function. Internal Report 81-2, Goettingen: Department of Neurobiology, Max Planck Intitute for Biophysical Chemistry, 1981.

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

[28] Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proc. Conf. of Cognitive Science Society*, pages 177–186, Seatle, WA, USA, July 1987.

[29] Subhaneil Lahiri and Surya Ganguli. A memory frontier for complex synapses. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 1034–1042, Lake Tahoe, NV, USA, December 2013.

[30] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 3987–3995, Sydney, Australia, August 2017.

[31] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 3559–3568, Stockholm, Sweden, July 2018.

[32] Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[33] Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Short-term plasticity neurons learning to learn and forget. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 18704–18722, Baltimore, MA, USA, July 2022.

[34] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 2554–2563, Sydney, Australia, August 2017.

[35] Tsendsuren Munkhdalai and Adam Trischler. Metalearning with hebbian fast weights. *Preprint arXiv:1807.05076*, 2018.

[36] Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned neural memory. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, December 2019.

[37] Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Virtual only, December 2020.

[38] Louis Kirsch and Jürgen Schmidhuber. Meta-learning backpropagation and improving it. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Virtual only, December 2021.

[39] Tsendsuren Munkhdalai. Sparse meta networks for sequential adaptation and its application to adaptive language modelling. *Preprint arXiv:2009.01803*, 2020.

[40] Vithursan Thangarasa, Thomas Miconi, and Graham W. Taylor. Enabling continual learning with differentiable hebbian plasticity. In *Proc. Int. Joint Conf. on Neural Networks*, pages 1–8, Glasgow, United Kingdom, July 2020.

[41] James C. R. Whittington, Joseph Warren, and Tim E. J. Behrens. Relating transformers to models and neural representations of the hippocampal formation. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, April 2022.

[42] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[43] Jerry A Fodor, Zenon W Pylyshyn, et al. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

[44] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

[45] Imanol Schlag and Jürgen Schmidhuber. Learning to reason with third order tensor products. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 9981–9993, Montréal, Canada, December 2018.

[46] S-I Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, 100(11):1197–1206, 1972.

[47] Wilhelm Lenz. Beitrag zum Verständnis der magnetischen Erscheinungen in festen Körpern. *Physikalische Zeitschrift*, 21:613–615, 1920.

[48] Ernst Ising. *Beitrag zur Theorie des Ferro- und Paramagnetismus*. PhD thesis, University of Hamburg, 1924.

[49] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Physikalische Zeitschrift*, 31:253–258, 1925.

[50] Hendrik A Kramers and Gregory H Wannier. Statistics of the two-dimensional ferromagnet. *Physical Review*, 60(3):252, 1941.

[51] Gregory H Wannier. The statistical problem in cooperative phenomena. *Reviews of Modern Physics*, 17(1):50, 1945.

[52] William A Little. The existence of persistent states in the brain. *Mathematical biosciences*, 19 (1-2):101–120, 1974.

[53] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[54] Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Spin-glass models of neural networks. *Physical Review A*, 32(2):1007, 1985.

[55] Ana P Millán, Joaquín J Torres, and Joaquín Marro. How memory conforms to brain development. *Frontiers in Computational Neuroscience*, 13:22, 2019.

[56] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, May 2021.

[57] Dmitry Krotov and John J. Hopfield. Large associative memory problem in neurobiology and machine learning. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, May 2021.

[58] Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal Hopfield networks: A general framework for single-shot associative memory models. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 15561–15583, Baltimore, MA, USA, July 2022.

[59] Stephen C Kleene et al. Representation of events in nerve nets and finite automata. *Automata studies*, 34:3–41, 1956.

[60] Alan M Turing. Intelligent machinery. Unpublished report for National Physical Laboratory. Published later in Meltzer and Michie "Machine Intelligence" vol 5 in 1969, 1948.

[61] Patrick C Trettenbrein. The demise of the synapse as the locus of memory: A looming paradigm shift? *Frontiers in systems neuroscience*, page 88, 2016.

[62] Samuel J Gershman. The molecular memory code and synaptic plasticity: a synthesis. *Available Online*, 2021.