Biological key-value memory networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

In neuroscience, Hopfield networks are the classical biologically plausible model of long-term memory, relying on Hebbian plasticity for storage and attractor dynamics for recall. In contrast, memory-augmented neural networks in machine learning commonly use a key-value mechanism to store and read out memories in a single step. Such networks can achieve impressive feats compared to traditional variants, yet their biological relevance is unclear. Here, we propose a biological implementation of basic key-value memory that stores inputs using a combination of Hebbian and non-Hebbian plasticity rules. Similar plasticity rules are recovered when network parameters are meta-learned. Our network performs similarly to Hopfield networks on autoassociative memory tasks and can be naturally extended to continual recall, heteroassociative memory, and sequence learning. Our results suggest a compelling alternative mechanism to Hopfield networks for biological long-term memory.

1 Introduction

Long-term memory is an essential aspect of our everyday lives. It is the ability to rapidly memorize an experience or item, and to retain that memory in a retrievable form over a prolonged duration (days to years in humans). Neural networks capable of long-term memory have been studied in both neuroscience and machine learning, yet a wide gap remains between the mechanisms and interpretations of the two traditions.

In neuroscience, long-term associative memory is typically modeled by variants of Hopfield networks [Hopfield, 1982, Amit and Amit, 1992, Willshaw et al., 1969]. Rooted in statistical physics, Hopfield networks represent one of the earliest and best known class of neural network models in neuroscience. A classical Hopfield network stores an activation pattern by strengthening the recurrent connections between co-active neurons using the biologically-plausible Hebbian plasticity rule. Hopfield networks allow retrieval of a memory from a corrupted version through recurrent attractor dynamics, thereby providing a content-addressable and pattern-completing autoassociative memory. In general, a network with N neurons can store $O(N)$ arbitrary N-dimensional patterns [Gardner and Derrida, 1988]. Variants of Hopfield networks were developed, for example, to increase the capacity for sparse patterns [Tsodyks and Feigel'man, 1988], and to avoid "blackout catastrophe" where all memories are lost when the number of patterns stored exceed capacity [Parisi, 1986].

In a parallel and more recent thread of development in machine learning, many different kinds of memory networks have been devised to augment traditional neural networks [Sukhbaatar et al., 2015, Graves et al., 2014, 2016, Munkhdalai et al., 2019, Le et al., 2019, Bartunov et al., 2019]. Inspired by computer memory, memory-augmented neural networks utilize a more stable external memory system than more volatile neural networks, such as recurrent neural networks. Many memory networks from this tradition can be viewed as consisting of memory slots where each slot can be addressed with a key and returns a memory value.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

A clear gap exists between these two lines of research on neural networks for long-term memory. In this work, we focus on studying whether key-value memory networks used in machine learning can provide a strong alternative to Hopfield networks as biological models for long-term memory. We introduce *KEVIN* (KEy-Value network, Implemented with Neurons), a biologically plausible neural circuit model that approximates key-value memory networks. Our network uses both Hebbian and non-Hebbian plasticity rules to store memories. We add increasing biological realism to our model and find that meta-learning of the plasticity mechanism still recovers the same plasticity rules as the original model. We further show how the feedforward, slot-based structure of *KEVIN* allows it be naturally applied to more biologically-motivated memory tasks.

2 Related work: Key-value memory networks

Key-value memory networks date back to at least the 1980s [Kanerva, 1988, 1992]. Inspired by random-access memory in computers, it is at the core of many memory networks recently developed in machine learning [Graves et al., 2014, 2016, Sukhbaatar et al., 2015, Banino et al., 2020, Le et al., 2019]. A basic key-value memory network contains a key matrix $\boldsymbol{K}$ and a value matrix $\boldsymbol{V}$. Given a query vector $\boldsymbol{q}$, a memory read operation will retrieve an output $\boldsymbol{y}$ as

$$\begin{aligned} \boldsymbol{h} &= f(\boldsymbol{K}\boldsymbol{q}) \\ \boldsymbol{y} &= \boldsymbol{V}^\intercal \boldsymbol{h} \end{aligned} \tag{1}$$

$f(\cdot)$ is an activation function that sparsifies the hidden response $\boldsymbol{h}$. It is commonly the softmax function [Sukhbaatar et al., 2015], making memory retrieval equivalent to the "key-value attention" [Bahdanau et al., 2015] used in many recent natural language processing models [Vaswani et al., 2017].

Key-value memory networks have also been derived from a different tradition. A number of recent works [Krotov and Hopfield, 2016, Ramsauer et al., 2020, Krotov and Hopfield, 2020] have described a family of modern Hopfield networks. Similar to Hopfield networks, an energy function can be defined such that autoassociative memory retrieval corresponds to an update in the input space that decreases the energy. For certain energy functions, the memory retrieval process is the key-value mechanism described above [Ramsauer et al., 2020, Krotov and Hopfield, 2020].

A clear advantage of key-value memory over classical Hopfield networks is the decoupling of memory capacity from input dimension [Krotov and Hopfield, 2020]. In Hopfield networks, the input dimension determines the number of recurrent connections, and thus upper bounds the capacity. In key-value memory networks, by increasing the size of the hidden layer, the capacity can be much larger when measured against the input dimension (the capacity per connection is similar).

Variations exist in the reading mechanisms of key-value memory networks. For example, $f(\cdot)$ has been set as the step function [Kanerva, 1988] and hard-max function [Weston et al., 2015]. Since we are interested in exploring whether key-value memory networks can serve as models for biological long-term memory, we use the version in Equation 1, as this basic memory read mechanism can be implemented in biological neural circuits [Kanerva, 1992].

There is an even greater variation across writing mechanisms of memory networks. Some works rely on highly flexible mechanisms where an external controller learns which slots to write and overwrite [Graves et al., 2016, 2014], although appropriate memory write strategies can be difficult to learn. Other works have used simpler mechanisms where new memories can be appended sequentially to the existing set of memories [Sukhbaatar et al., 2015, Hung et al., 2019, Ramsauer et al., 2020] or written through gradient descent [Bartunov et al., 2019, Munkhdalai et al., 2019, Le et al., 2019].

It remains unclear whether a biological mechanism can implement the memory write. Kanerva [1992] updates the value matrix through Hebbian plasticity, but fixes the key matrix. Munkhdalai et al. [2019] turns memory write into a key-target value regression problem, and updates an arbitrary feedforward memory network using metalearned local regression targets. We will show that biologically-plausible plasticity rules exist when both key and value are updated as in Equation 1.

3 Simplified *KEVIN* mechanism

We begin by describing a simplified version of *KEVIN*, a biologically plausible implementation of the slot-based key-value memory models. This version will serve as a benchmark to compare against

existing memory networks and will be further modified to increase neural plausibility and improve performance in biologically-relevant tasks.

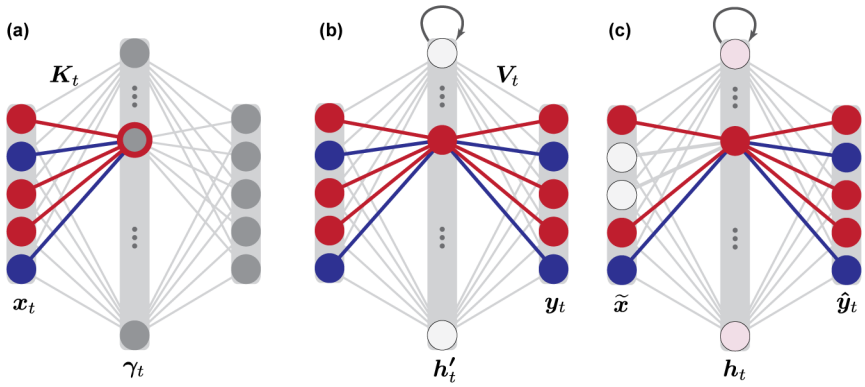

Figure 1: Network architecture and read/write mechanism. (a) A hidden neuron is selected via local third factor and the input is written into the corresponding slot of the key matrix with a "pre-only" plasticity rule. (b) The same hidden neuron is selected through hidden unit activation and the target is written to the value matrix with a Hebbian update. (c) The network is given a query (corrupted version of stored key) and returns the corresponding value.

KEVIN consists of three fully-connected layers of neurons (Figure 1): a d-dimensional input with activity at time t given by the vector $\boldsymbol{x}_t$, an N-dimensional hidden layer $\boldsymbol{h}_t$, and an m-dimensional output layer $\boldsymbol{y}_t$. Given an input $\boldsymbol{x}_t$, the hidden layer activity is given by

$$\boldsymbol{h}_t = \text{softmax}(\boldsymbol{K}_t \boldsymbol{x}_t) \tag{2}$$

where the input-to-hidden synaptic weights $\boldsymbol{K}_t$ forms the key matrix. The softmax function normalizes the activity, and can be approximated biologically with inhibitory recurrent connections. The output layer activity is a simple linear readout,

$$\hat{\boldsymbol{y}}_t = \boldsymbol{V}_t \boldsymbol{h}_t \tag{3}$$

When target values are binary, $\boldsymbol{y}_t \in \{+1, -1\}^m$ (neurons are either active or silent), we de-noise this output by taking $\text{sign}(\hat{\boldsymbol{y}}_t)$ when evaluating performance.

The network stores a key-value mapping such that if a query $\widetilde{\boldsymbol{x}}$ is presented to the network, it returns the corresponding value $\boldsymbol{y}$. For this, the synaptic weights $\boldsymbol{K}_t$ and $\boldsymbol{V}_t$ are updated according to biologically plausible plasticity rules, involving only the pre- and post-synaptic neuronal activity as well as a third modulating factor [Gerstner et al., 2018].

3.1 Writing keys

The inputs are stored in the the key matrix $\boldsymbol{K}_t$, where a memory slot corresponds to the connection weights from the input to a single hidden neuron (Figure 1a). Given an input $\boldsymbol{x}_t$, we write it into one of the slot of the key matrix using a non-Hebbian plasticity rule, where the presynaptic neuronal activity dictates the synaptic update $\Delta w_{ij} \sim x_j$ (rather than pre- and postsynaptic co-activation $\Delta w_{ij} \sim h_i x_j$ as in a traditional Hebbian rule). This plasticity rule resembles the recently characterized behavioral time scale plasticity (BTSP) [Bittner et al., 2017] discovered in hippocampus, the brain structure critical for formation of long-term memory.

We furthermore introduce a local third factor $[\boldsymbol{\gamma}_t]_i \in \{0, 1\}$ (Figure 1a, red circle) that gates the plasticity of input connections to hidden unit i, enabling selection of a single neuron for writing. Biologically, this may correspond to a dentritic spike [Stuart and Spruston, 2015, Gambino et al., 2014]. Importantly, the plateaus occur independently of the somatic activity $\boldsymbol{h}_t$ which performs the feedforward computation. In this simplified model we approximate them as occurring in the least-recently-written-to neuron by cycling through the hidden units sequentially,

$$[\boldsymbol{\gamma}_t]_i = \begin{cases} 1 \text{ if } t = i \bmod N \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

Alternatively, we consider an even simpler mechanism for local third factor selection, which does not require any coordination among the hidden layer neurons: each neuron independently has some probability p of generating a dendritic spike:

$$[\boldsymbol{\gamma}_t]_i = \text{Bernoulli}(p) \tag{5}$$

To gate whether a stimulus should be stored at all, we include a scalar global third factor $q_t \in \{0, 1\}$. Biologically, this may correspond to a neuromodulator such as acetylcholine [Rasmusson, 2000] that affects the entire population of neurons, controlled by novelty or other global signals. Thus, the final learning rate of the synapse between input unit i and hidden unit j is the product of the local and global third factors:

$$[\boldsymbol{\eta}_t^k]_{ij} = q_t[\boldsymbol{\gamma}_t]_i \tag{6}$$

Finally, to allow reuse of memory slots, we introduce a forgetting mechanism, corresponding to a rapid synaptic decay mediated by the local third factor. Whenever a synapse gets updated we have $[\boldsymbol{\eta}_t^k]_{ij} = 1$, so we can zero out its value by multiplying it by $1 - [\boldsymbol{\eta}_t^k]_{ij}$. If there is no update (either third factor is zero), the weight is not affected. The synaptic update is therefore:

$$\boldsymbol{K}_{t+1} = (1 - \boldsymbol{\eta}_t^k) \odot \boldsymbol{K}_t + \boldsymbol{\eta}_t^k \odot [\mathbf{1}\boldsymbol{x}_t^T] \tag{7}$$

where $\odot$ indicates the Hadamard (element-wise) product, and $\mathbf{1} \equiv (1, 1, .., 1)$.

3.2 Writing values

Values are stored in the hidden-to-output weights, the "value matrix" $\boldsymbol{V}_t$. Since a hidden neuron (more precisely, its local third factor) is indexed to store a key, we use same index to store the value. The hidden layer activity at this intermediate stage is given by

$$\boldsymbol{h}'_t = \text{softmax}(\boldsymbol{K}_{t+1}\boldsymbol{x}_t) \tag{8}$$

In the idealized case of random uncorrelated binary random memories $\boldsymbol{x}_t \in \{+1, -1\}^d$, assuming the current input $\boldsymbol{x}_t$ has been successfully stored in the ith row of the key matrix, the ith entry of $\boldsymbol{K}_{t+1}\boldsymbol{x}_t$ will be equal to $\boldsymbol{x}_t \cdot \boldsymbol{x}_t = d$. All other entries will be much closer to 0, since they are uncorrelated with $\boldsymbol{x}_t$. Thus, after normalization via the softmax, $\boldsymbol{h}'_t$ will be approximately one-hot, with $[\boldsymbol{h}'_t]_i \approx 1$.

To store the value, the output layer activity is clamped to the target $\boldsymbol{y}_t$ (Figure 1b). Biologically, this can be achieved either by strong one-to-one residual connections from the input to the output layer, or by having a common circuit which drives activity in both the input and output layers. Since the only hidden unit active is the one indexing the ith column of the value matrix, we can update it by a Hebbian rule, with learning and rapid decay rates[1] analogous to those for the key matrix:

$$\boldsymbol{V}_{t+1} = (1 - \boldsymbol{\eta}_t^v) \odot \boldsymbol{V}_t + \boldsymbol{\eta}_t^v \odot \boldsymbol{y}_t(\boldsymbol{h}'_t)^T \tag{9}$$

3.3 Reading

Having stored a set of key-value pairs $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}$ in the network, given a query $\tilde{\boldsymbol{x}}$, the hidden layer activation (Equation 2) computes its similarity to each stored key (row of $\boldsymbol{K}_t$). The output layer then uses these similarity scores to compute the estimated value as the weighted sum of the stored values (columns of $\boldsymbol{V}_t$). Assuming uncorrelated inputs, the dot product of the query $\tilde{\boldsymbol{x}}$ will be maximal with its corresponding stored key $\boldsymbol{x}^*$, and near zero for all other stored keys, so $\boldsymbol{h}$ will be approximately one-hot. Thus, Equation 3 reduces to $\hat{\boldsymbol{y}} \approx \boldsymbol{y}^*$ as desired (Figure 1c).

4 Results

4.1 Benchmark: autoassociative recall

As a simple evaluation of *KEVIN*'s performance and comparison to other memory networks, we consider the classical autoassociative memory recall task where the key is equal to the value, ($d = m$,

[1] If local third factors are plateau potentials in the *dendrites* of hidden layer neurons, it may not be realistic for the *axons* of the same neurons to be affected by these plateau potentials. We consider this case for simplicity, as an upper-bound on performance. We will later lift this assumption without significantly hurting performance.

Figure 1). The network stores a set of T stimuli $\{\boldsymbol{x}_t\}$ and the query $\tilde{\boldsymbol{x}}$ is a corrupted version of a stored key (60% of the entries are randomly set to zero). The network returns the corresponding uncorrupted version (Appendix A, Figure A1).

We compute the accuracy as a function of the number of stored stimuli (Figure 2a). By design, *KEVIN* with sequential local third factors performs identically to a simple version of a key-value memory network (TVT, [Hung et al., 2019], Appendix B). It has perfect accuracy for $T \leq N$ since every pattern is stored in a unique slot. For $T > N$, previously stored patterns are overwritten one-by-one and accuracy smoothly degrades. With random local third factors, accuracy begins to degrade sooner due to random overwriting. This holds for arbitrary network sizes, unlike the classical Hopfield network, which experiences a "blackout catastrophe" if the network size is large and the number of inputs exceeds its capacity [McCloskey and Cohen, 1989, Robins and McCallum, 1998], where all stored patterns are erased.

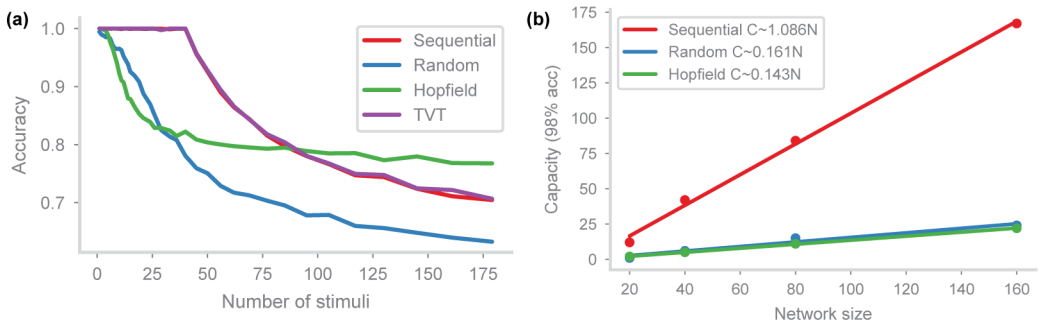

Figure 2: *KEVIN* ($d = N = 40$) with sequential and random ($p = 0.1$) local third factors, Hopfield network, and TVT [Hung et al., 2019] performance on the autoassociative memory task. (a) Accuracy as a function of stored stimuli. (b) Capacity (maximum number of stored stimuli at $\geq 98\%$ accuracy) as a function of network size.

Next, by measuring the number of patterns that the network can store before its recall accuracy drops below a threshold ($\theta = 0.98$), we can estimate this network's memory capacity C. This is an imperfect measure because some networks may have accuracy that stays above threshold longer but drops sharply after that. Nevertheless, this metric allows us to investigate empirically how the network's performance scales with its size (Figure 2b). We note that with the same number of hidden neurons, *KEVIN* has twice as many connections as Hopfield networks.

The empirical scaling of the Hopfield network is $C \sim 0.14N$ [Amit et al., 1985], consistent with theoretical calculations, and the sequential *KEVIN* capacity scales approximately as $C \sim 1.0N$, as expected. Importantly, the random *KEVIN* capacity also scales linearly, $C \sim 0.16N$, with a slope similar to the Hopfield network.

4.2 Meta-learning of plasticity rules

We now introduce parameters that can be meta-learned to optimize performance on a particular dataset without sacrificing biological plausibility. First, we enable varying the scale of the synaptic plasticity rates – each one is multiplied by a learnable parameter $\widetilde{\boldsymbol{\eta}}^k, \widetilde{\boldsymbol{\eta}}^v$:

$$[\boldsymbol{\eta}_t^k]_{ij} = q_t[\boldsymbol{\gamma}_t]_i[\widetilde{\boldsymbol{\eta}}^k]_{ij} \text{ and } [\boldsymbol{\eta}_t^v]_{ij} = q_t[\widetilde{\boldsymbol{\eta}}^v]_{ij} \tag{10}$$

Next, as a further improvement on biological plausibility, we remove the assumption that the hidden-to-output synapses (hidden layer axons) have access to the local third factors (hidden layer dendritic plateau potentials). Instead, as a forgetting mechanism, the hidden-to-output synapses decay by a fixed factor $\widetilde{\boldsymbol{\lambda}}$ each time a stimulus is stored:

$$\boldsymbol{\lambda}_t = 1 - (1 - \widetilde{\boldsymbol{\lambda}})q_t \tag{11}$$

Although in general $\widetilde{\boldsymbol{\eta}}^k, \widetilde{\boldsymbol{\eta}}^v$, and $\widetilde{\boldsymbol{\lambda}}$ can each be a matrix which sets a learning/decay rate for each synapse or neuron, we consider the simpler case where each is a scalar, shared across all synapses.

Most importantly, we parameterize the update rule itself. Each of the pre- and postsynaptic firing rates is linearly transformed before the synaptic weight is updated according to the typical "pre-times-post" rule. The ith input neuron's transformed firing rate is $[f^k(\boldsymbol{x})]_i = \widetilde{a}^{f^k} x_i + \widetilde{b}^{f^k}$, and others are analogous. Although these functions can take arbitrary forms in general, this simple parameterization enables interpolating between traditional Hebbian, anti-Hebbian, and non-Hebbian (pre- or post-only) rules. The update rules can be summarized as follows:

$$\boldsymbol{K}_{t+1} = (1 - \boldsymbol{\eta}_t^k) \odot \boldsymbol{K}_t + \boldsymbol{\eta}_t^k \odot [g^k(\boldsymbol{h}_t) f^k(\boldsymbol{x}_t)^T] \tag{12}$$

$$\boldsymbol{V}_{t+1} = \boldsymbol{\lambda}_t \odot \boldsymbol{V}_t + q_t \widetilde{\boldsymbol{\eta}}^y \odot [g^v(\boldsymbol{y}_t) f^v(\boldsymbol{h}'_t)^T] \tag{13}$$

We begin by training this network on the benchmark task from subsection 4.1, optimizing these parameters using stochastic gradient descent (Adam, [Kingma and Ba, 2014]). Figure 3a shows the performance of the trained *KEVIN* in comparison to the corresponding simple networks. Importantly, removing the local third factor in the hidden-to-output synapses and replacing it with a passive decay does not significantly impact performance, as long as the decay rate is appropriately set, and may even improve performance for longer sequences. In the case of a random local third factor, we also compare values of p. Empirically, we find that $p \approx 4/N$ produces desirable performance: it ensures that the probability of no local third factor occurring (and therefore no storage) is small ($< 2\%$) while minimizing overwriting. Computing the capacity (Figure 3b), we see that there is an advantage for the probability to scale with the network size rather being a fixed value.

We next investigate the learning trajectories. Figure 3c shows the loss and accuracy curves over the course of training for networks with sequential and random ($p = 4/N$) local third factors. Training data consists of sequence lengths between $T = N/2$ and $T = 2N$ (above capacity for both versions of the network), so accuracy does not reach 1.0. Nevertheless, the network successfully generalizes to lengths outside of this range, indicating a robust mechanism for memory storage.

Most importantly, we examine the plasticity rule discovered by optimization. Figure 3d shows the slope $\widetilde{a}$ and offset $\widetilde{b}$ parameters for the firing rate transfer functions g^k, f^k, g^v, f^v in the sequential *KEVIN* over the course of training. The plasticity rule for the input-to-hidden connections (Figure 3d, left) becomes pre-dependent ($\widetilde{a}^{f^k} \approx 0.5$, $\widetilde{b}^{f^k} \approx 0$) but not post-dependent ($\widetilde{a}^{g^k} \approx 0$, $\widetilde{b}^{g^k} \approx 0.5$), analogous to the idealized plasticity rule described in section 3. Similarly, the plasticity rule for the hidden-to-output connections becomes Hebbian (both pre-dependent and post-dependent, $\widetilde{b}^{g^v} \approx \widetilde{b}^{f^v} \approx 0$, but $\widetilde{a}^{g^v} \approx \widetilde{a}^{f^v} \approx 1$). The random *KEVIN* shows qualitatively similar trained behavior (Figure A5).

Since the performance and parameterization of the trained *KEVIN* is almost identical to the simple case, we only train the version with random local third factors in subsequent sections.

4.3 Continual, flashbulb, and correlated memory tasks

We next test *KEVIN* on more ecologically relevant memory tasks. These tasks reflect the complexity of biological stimuli and the functional need for versatile memory storage.

In realistic scenarios, memories are stored and recalled in a continual manner – the subject sees an ongoing stream of inputs and is required to recall a stimulus that was shown some time ago (Appendix A). Figure A2a shows such a sample dataset with a delay interval of 2. *KEVIN* is naturally suited for this task due to its decay mechanisms – recent stimuli are stored, but distal ones are forgotten. Its accuracy decreases in steps of width N since all memories with delay kN are equally likely to be recalled. Performance with random third factors decays smoothly since writing is stochastic, but both networks show better overall performance than the Hopfield network (Figure 4a).[2]

Next, we consider "flashbulb" memories, a phenomenon where highly salient experiences are remembered with extreme efficacy, often for life [Brown and Kulik, 1977]. Functionally, these memories may be used to avoid adverse experiences or seek out rewarding ones. We modify the continual recall task such that a small number of stimuli are deemed salient and accompanied by a stronger global third factor, akin to a boosted neuromodulatory influence on learning. We add a simple

[2]For fair comparison, we introduce an empirically chosen synaptic decay parameter $\lambda = 0.95$ to the Hopfield network. We also consider a version of the Hopfield network with bounded weights [Parisi, 1986], designed for continual learning, but we find that synaptic weight decay has better overall performance.

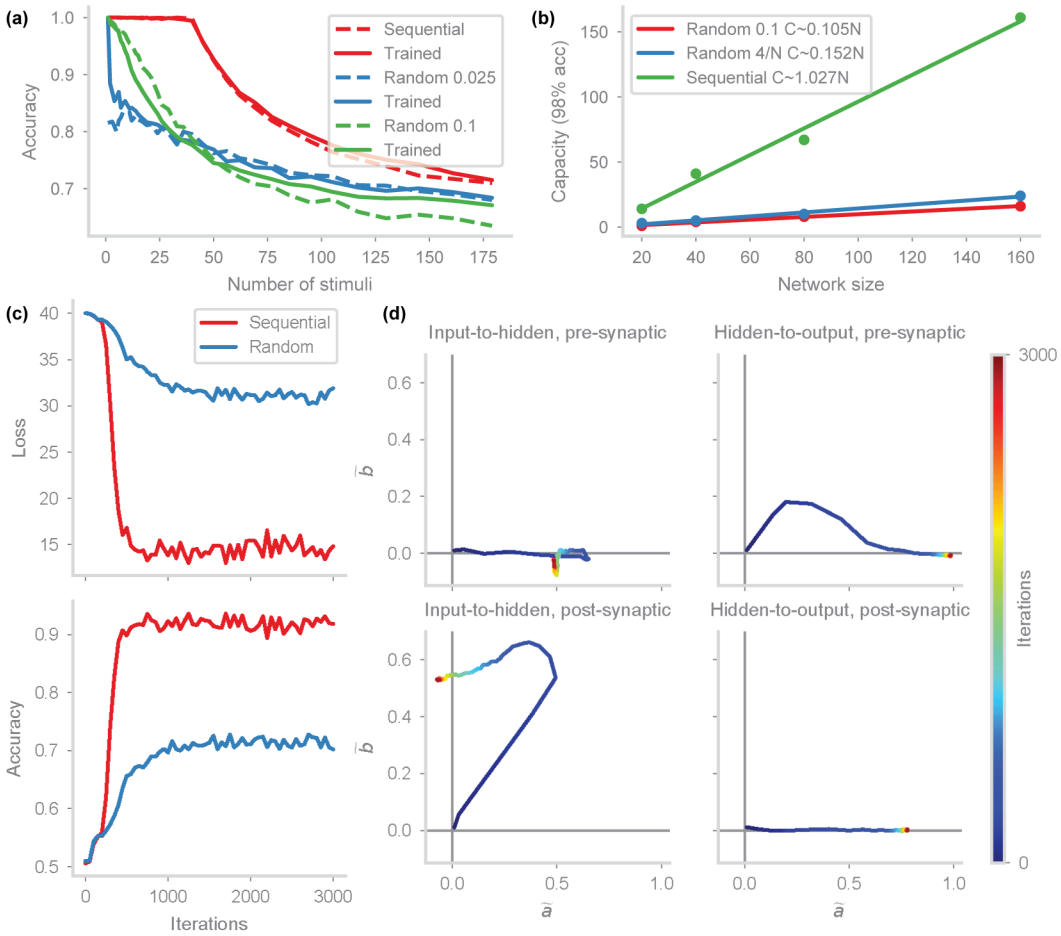

Figure 3: (a) Performance comparison of simple and parameterized (trained) *KEVIN* ($d = N = 40$, p varied, shown in legend.) (b) Capacity of trained *KEVIN* with sequential or random (fixed p or scaled with N) local third factor. (c) Training the sequential and random ($p = 0.1$) network from (a,b) Although accuracy is not 1.0 during training, the network successfully generalizes to unseen sequence lengths. (d) Learning trajectory of the plasticity parameters in the sequential network converges to qualitatively similar solutions as the original simplified network (see also Figure A5).

meta-plasticity mechanism to *KEVIN*, a stability parameter $[\boldsymbol{S}_t]_{ij}$ for each synapse that suppresses subsequent plasticity events if the learning rate is above a threshold i.e. $[\boldsymbol{S}_{t+k}]_{ij} = 1$ for all $k \geq 0$ if $[\boldsymbol{\eta}_{t-1}]_{ij} > 1$. Thus, the learning rate for the input-to-hidden layer is as follows (learning rate $\boldsymbol{\eta}_t^v$ is analogous):

$$[\boldsymbol{\eta}_t^k]_{ij} = (1 - [\boldsymbol{S}_t^h]_{ij}) q_t [\boldsymbol{\gamma}_t]_i \tag{14}$$

KEVIN retains flashbulb memories with minimal effect on its recall performance on regular memory items (Figure 4b, left, middle). To perform this task in the Hopfield network, introducing synaptic stability is significantly detrimental to performance, so we simply store flashbulb memories as large-magnitude updates to the weight matrix. As a result, it exhibits a tradeoff between storage of regular items and efficacy of flashbulb memories (Figure 4b, right). A benefit of the slot-based storage scheme is that flexible treatment of individual memories can be naturally implemented.

Finally, real-world stimuli are often spatially and temporally correlated – for instance, two adjacent video frames are almost identical. Storing such patterns in a Hopfield network causes interference between attractors in the energy function, thus decreasing its capacity. In contrast, by storing stimuli in distinct slots of the key and value matrices, *KEVIN* can more easily differentiate between correlated but distinct memories. To verify this, we generate a correlated dataset by starting with a "template" vector and generating stimuli by randomly flipping some fraction of its entries Figure A2. The

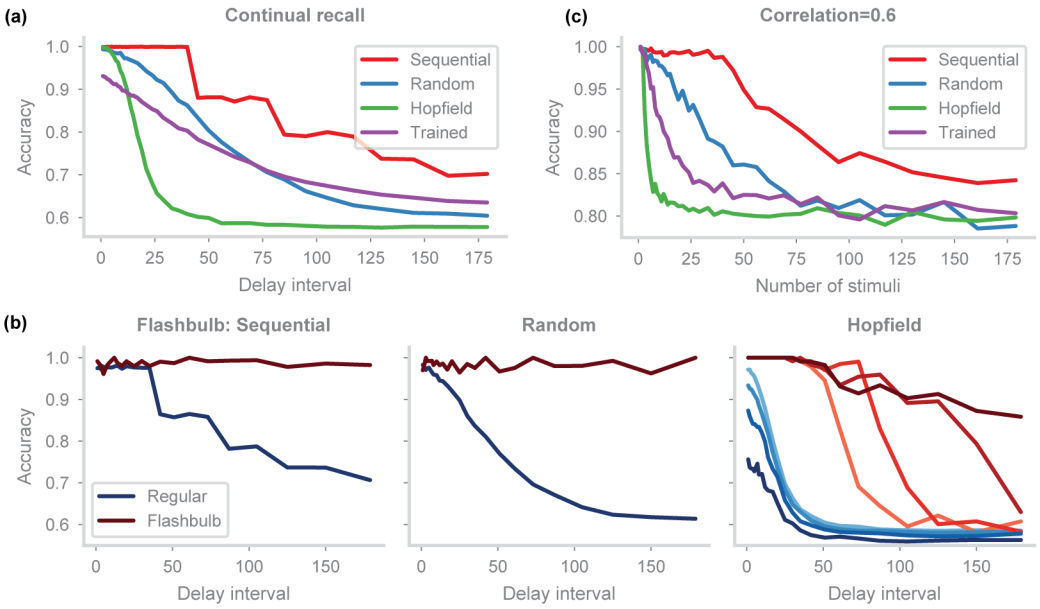

Figure 4: (a) Performance on continual recall task, accuracy measured as a function of the number of timesteps between the storage and recall of a memory. (b) Same as (a), but including 5 "flasbulb" memories. Performance of simplified *KEVIN* with sequential (left) or random (middle) local third factors and stable synapses. For Hopfield network (right), the darkness of each line corresponds to the write strength for flashbulb memories. Lighter lines indicate lower write strengths. (c) Same as Figure 2a, but with memories having correlation of 0.6.

performance of *KEVIN* is similar to that for uncorrelated data (Figure 2a), with minor degradation due to spurious recall of similar stored memories (Figure 4c). Figure A3 shows similar results for different correlation strengths.

4.4 Heteroassociative and sequence memory

The learning mechanism of *KEVIN* is agnostic to the relationship between the input and target patterns, and so naturally generalizes to heteroassociative memory. To draw comparisons with Hopfield-type networks, we consider the Bidirectional Associative Memory (BAM) [Kosko, 1988], a generalization of the Hopfield network suited for heteroassociative recall. We evaluate the networks on a modified version of the recall task from subsection 4.1 where target patterns are distinct from the keys $\boldsymbol{y}_t \in \{+1, -1\}^m$ $(d \neq m)$ (Figure 5a, top, Figure A4). Compared to the autoassociative task (Figure 2), the Hopfield-type network's performance on the heteroassociative task suffers (Figure A4). On the other hand, the performance of *KEVIN* remains unaffected, as the factorized key-value structure of *KEVIN* may allow more flexibility in the types of memory associations learned.

A more biologically relevant version of heteroassociative memory is sequence learning. Experimental and theoretical evidence suggests that hippocampal memory can serve as a substrate for planning and decision making through the replay of sequential experiences [Foster and Wilson, 2006, Pfeiffer and Foster, 2013, Mattar and Daw, 2018]. As a simple probe for a similar functionality, we use a sequence recall task where a network is presented with a sequence of patterns to memorize. Afterwards, the network is prompted with any patterns in the sequence and tasked with recalling the remaining patterns in the sequence. By adding a recurrent loop (Figure 5b, top), *KEVIN* with sequential third factors can perform sequence learning without error until its capacity limit (Figure 5b, bottom). BAM does not perform as well, likely due to propagating errors from incorrect recall of earlier patterns in the sequence.

Finally, we consider a more complex task that requires our memory module to be integrated within a larger system, demonstrating the modular nature of our memory network. In the "copy-paste" task, the system must store a variable-length sequence of patterns and output this sequence when the

end-of-sequence marker is seen (Figure A4). We train a feedforward network as a "controller" to generate $\boldsymbol{x}_t$, $\boldsymbol{y}_t$, and $\boldsymbol{q}_t$ (Figure 5c, top) (for BAM, $\boldsymbol{q}_t$ scales the magnitude of the Hebbian update). With this controller, *KEVIN* successfully learns the task (Figure 5c, bottom). Interestingly, BAM performs better on this task, despite having a theoretically lower capacity. This suggests *KEVIN* may be more sensitive to the design of the controllers than BAM and is a topic for future study.

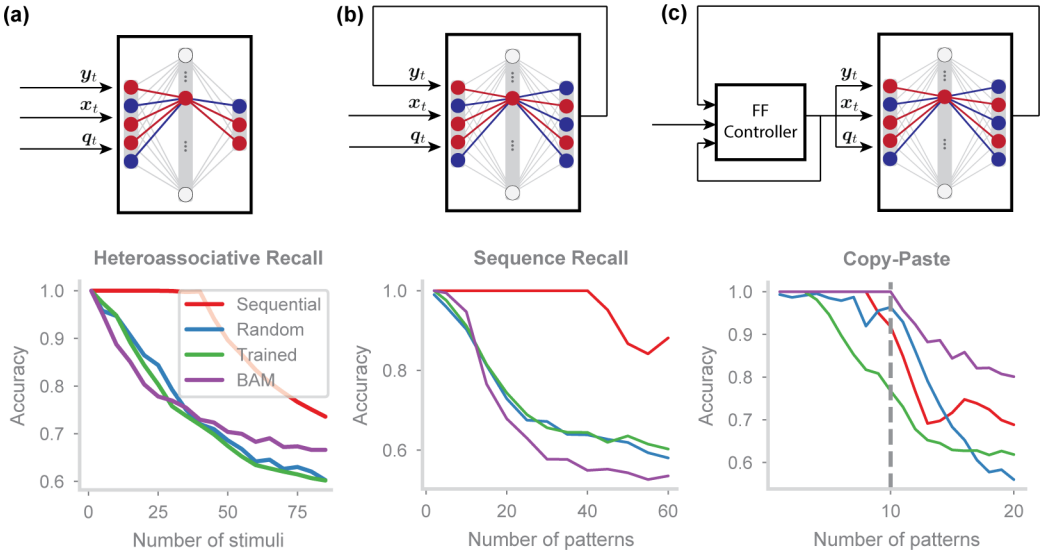

Figure 5: (a) Architecture ($d = N = 40, m = 20$) and performance on heteroassociative version of the benchmark task (Figure 2). (b) Recurrent architecture modification for sequence recall, $\boldsymbol{y}_t = \boldsymbol{x}_{t-1}$. Accuracy is plotted as a function of the length of the entire sequence presented. (c) Network is embedded in a larger system with a feedforward network to perform the copy-paste task. Dashed line is the maximum number of patterns shown during training.

5 Discussion

We proposed models of biological memory by taking inspiration from key-value memory networks used in machine learning. These biologically plausible models use a combination of Hebbian and non-Hebbian plasticity rules to approximate key-value memory networks. It can naturally be adapted for biologically relevant tasks due to the flexibility of its structure.

Our results suggest an alternative framework of biological long-term memory that focuses on feedforward computation, as opposed to recurrent attractor dynamics. Intriguingly, both our hand-designed and machine evolved results propose a role for recently discovered non-classical plasticity rules [Bittner et al., 2017] that are advantageous for this type of feedforward computation. Furthermore, we propose an architecture where individual memories are stored more independently of each other than in Hopfield networks. Such a factorized design creates opportunities for more versatile use and control of memory.

Our study is limited in a few ways. We focused on artificial stimuli with simple statistical structures. It is unclear how our model will perform with more complex and naturalistic datasets. A more principled approach may be needed to determine how memory writes should occur with structured datasets [Wu et al., 2018]. We also have not made direct comparisons between *KEVIN* and experimental findings in memory-related behavior or memory-related neural activity. Finally, long-term memory describes many categories of memory supported by many brain regions. We have not disambiguated between these differences and their implications on testing and interpreting our model.

Taken together, our results take a neuroscience-minded approach to connect two lines of work in memory networks that have been largely disparate.

References

D. J. Amit and D. J. Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1992.

D. J. Amit, H. Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015.

A. Banino, A. P. Badilea, R. Köster, M. J. Chadwick, V. Zambaldi, D. Hassabis, C. Barry, M. Botvinick, D. Kumaran, and C. Blundell. Memo: A deep network for flexible combination of episodic memories. *arXiv preprint arXiv:2001.10913*, 2020.

S. Bartunov, J. W. Rae, S. Osindero, and T. P. Lillicrap. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*, 2019.

K. C. Bittner, A. D. Milstein, C. Grienberger, S. Romani, and J. C. Magee. Behavioral time scale synaptic plasticity underlies ca1 place fields. *Science*, 357(6355):1033–1036, 2017.

R. Brown and J. Kulik. Flashbulb memories. *Cognition*, 5(1):73–99, 1977.

D. J. Foster and M. A. Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440:680–683, 2006.

F. Gambino, S. Pagès, V. Kehayas, D. Baptista, R. Tatti, A. Carleton, and A. Holtmaat. Sensory-evoked ltp driven by dendritic plateau potentials in vivo. *Nature*, 515(7525):116–119, 2014.

E. Gardner and B. Derrida. Optimal storage properties of neural network models. *Journal of Physics A: Mathematical and general*, 21(1):271, 1988.

W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea. Eligibility traces and plasticity on behavioral time scales: Experimental support of neohebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018. ISSN 1662-5110. doi: 10.3389/fncir.2018.00053. URL `https://www.frontiersin.org/article/10.3389/fncir.2018.00053`.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

C.-C. Hung, T. Lillicrap, J. Abramson, Y. Wu, M. Mirza, F. Carnevale, A. Ahuja, and G. Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):1–12, 2019.

P. Kanerva. *Sparse distributed memory*. MIT press, 1988.

P. Kanerva. *Sparse distributed memory and related models*, volume 92. NASA Ames Research Center, Research Institute for Advanced Computer Science, 1992.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, man, and Cybernetics*, 18(1):49–60, 1988.

D. Krotov and J. Hopfield. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.

D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. *arXiv preprint arXiv:1606.01164*, 2016.

H. Le, T. Tran, and S. Venkatesh. Neural stored-program memory. *arXiv preprint arXiv:1906.08862*, 2019.

M. G. Mattar and N. D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21:1609–1617, 2018.

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

T. Munkhdalai, A. Sordoni, T. Wang, and A. Trischler. Metalearned neural memory. *arXiv preprint arXiv:1907.09720*, 2019.

G. Parisi. A memory which forgets. *Journal of Physics A: Mathematical and General*, 19(10):L617, 1986.

B. E. Pfeiffer and D. J. Foster. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497:74–79, 2013.

H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

D. D. Rasmusson. The role of acetylcholine in cortical synaptic plasticity. *Behav Brain Res.*, 115(2): 205–218, 2000. doi: 10.1016/s0166-4328(00)00259-x.

A. Robins and S. McCallum. Catastrophic forgetting and the pseudorehearsal solution in hopfield-type networks. *Connection Science*, 10(2):121–135, 1998. doi: 10.1080/095400998116530. URL `https://doi.org/10.1080/095400998116530`.

G. J. Stuart and N. Spruston. Dendritic integration: 60 years of progress. *Nature neuroscience*, 18 (12):1713–1721, 2015.

S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

M. V. Tsodyks and M. V. Feigel'man. The enhanced storage capacity in neural networks with low activity level. *EPL (Europhysics Letters)*, 6(2):101, 1988.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969.

Y. Wu, G. Wayne, A. Graves, and T. Lillicrap. The kanerva machine: A generative distributed memory. *arXiv preprint arXiv:1804.01756*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See final paragraph of discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We believe our work is relevant to basic science research in the neuroscience field. These are small networks designed with a focus on biological plausibility over optimal performance.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Submitted in an attached zip file.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The key dataset parameters, hyperparameters, and training algorithm are noted in the main text or figure captions. Additional settings are clarified in the appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] These were small networks that we ran on standard laptops.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the Temporal Value Transport memory network in the appendix and modify their code for our purposes. This is cited in the appendix and described in more detail.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]