Neural Networks 186 (2025) 107216

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Daydreaming Hopfield Networks and their surprising effectiveness on correlated data

Ludovica Serricchio [a,b], Dario Bocchi [a], Claudio Chilin [c], Raffaele Marino [d], Matteo Negri [a,e,*], Chiara Cammarota [a,f], Federico Ricci-Tersenghi [a,e,f]

[a] Dipartimento di Fisica, Sapienza Università di Roma, Piazzale Aldo Moro 5, Rome, 00185, Italy
[b] Center for Life Nano & Neuro-Science, Istituto Italiano di Tecnologia, Viale Regina Elena 291, Rome, 00161, Italy
[c] Departamento de Física Teórica, Universidad Complutense, Pl. de las Ciencias 1, Madrid, 28040, Spain
[d] Physics Department, Università degli Studi di Firenze, Via Sansone 1, Firenze, 50019, Italy
[e] CNR-Nanotec Rome unit, Piazzale Aldo Moro 5, Rome, 00185, Italy
[f] INFN Sezione di Roma1, Piazzale Aldo Moro 5, Rome, 00185, Italy

A R T I C L E I N F O

Keywords:
Unlearning
Hopfield networks
Structured data
Statistical mechanics

A B S T R A C T

To improve the storage capacity of the Hopfield model, we develop a version of the dreaming algorithm that *perpetually* reinforces the patterns to be stored (as in the Hebb rule), and erases the spurious memories (as in dreaming algorithms). For this reason, we called it *Daydreaming*. Daydreaming is not destructive and it converges asymptotically to stationary retrieval maps. When trained on random uncorrelated examples, the model shows optimal performance in terms of the size of the basins of attraction of stored examples and the quality of reconstruction. We also train the Daydreaming algorithm on correlated data obtained via the random-features model and argue that it spontaneously exploits the correlations thus increasing even further the storage capacity and the size of the basins of attraction. Moreover, the Daydreaming algorithm is also able to stabilize the features hidden in the data. Finally, we test Daydreaming on the MNIST dataset and show that it still works surprisingly well, producing attractors that are close to unseen examples and class prototypes.

1. Introduction

Hopfield Networks (Hopfield, 1982) are among the most well-known models for storing and retrieving memories in a neural network. The original Hopfield model, based on the Hebb rule (Hebb, 1949), is both analytically tractable (Amit, Gutfreund, & Sompolinsky, 1985, 1987b) and biologically plausible in its dynamics (Amit, 1989). Indeed, the retrieval dynamics in the original Hopfield model is defined by the following updating rule

$$s_i^{(t+1)} = \text{sign}\left(\sum_{j=1}^{N} J_{ij} s_j^{(t)}\right) , \qquad J_{ij} = \frac{1}{N}\sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu , \tag{1}$$

where the N neurons take discrete values ($s_i = \pm 1$), and the P discrete patterns $\{\xi^\mu\}_{\mu=1,\dots,P}$ are the memories stored in the network. The retrieval of a memory is successful if the spin dynamics converge to one of the stored patterns.

Although the number of patterns P that can be stored and efficiently retrieved in the Hopfield model is extensive, i.e. linear in the number of neurons N, the storage capacity (or maximum load) α_c is rather limited. Indeed, memories can be efficiently retrieved only if $\alpha = P/N < \alpha_c \simeq 0.138$ (Amit et al., 1987b), that is the number of stored memories is just a small fraction of the number of neurons. Given this limitation of the Hebb rule, it is crucial to find other rules that can improve the storage capacity of the Hopfield model.

To overcome such limitations, researchers have introduced new strategies. One such strategy involves changing the nature of interactions by considering many-body terms (i.e. multi-spin interactions). This approach significantly enhances the capacity, making it more than linear (Agliari, Albanese et al., 2023; Gardner, 1987; Krotov & Hopfield, 2016), and in some cases even exponential (Demircigil, Heusel, Löwe, Upgang, & Vermet, 2017; Lucibello & Mézard, 2023; Ramsauer et al., 2020). While these strategies have demonstrated powerful applications, we will not discuss them in this work, as we are primarily interested in maintaining the Hebbian-like pairwise interactions, that lead to the simple and biologically plausible dynamics shown above.

In the original Hopfield model, the coupling matrix J is given by the Hebb rule and does not need to be learned.[1] Analogously there exist

* Corresponding author at: Dipartimento di Fisica, Sapienza Università di Roma, Piazzale Aldo Moro 5, Rome, 00185, Italy.
E-mail address: matteo.negri@uniroma1.it (M. Negri).

[1] Alternatively one can say that couplings are learned on the very first step as soon as the P patterns have been seen once.

https://doi.org/10.1016/j.neunet.2025.107216
Received 14 June 2024; Received in revised form 24 January 2025; Accepted 26 January 2025
Available online 13 February 2025

other and more efficient rules that produce a different coupling matrix J again without the need to learn it through a dynamical learning process (we review these rules in Section 1.1.1). In our approach, instead, we assume no a priori information and we learn the coupling matrix from the data (i.e. the patterns) through a learning dynamical process. In this learning context, the patterns are also called examples.

Our work elaborates on a learning strategy of the Hopfield model called *dreaming* (or *unlearning*) (Hopfield, Feinstein, & Palmer, 1983). The terminology is inspired by the hypothesis that the human brain, during the REM sleep phase, selectively erases useless memories while reinforcing useful ones (Crick & Mitchison, 1983). In this context, the Hebb rule can be interpreted as the "day" phase where the memories to be stored are encoded in the synapses' strengths (i.e. in the couplings), while the dreaming procedure can be interpreted as the "night" phase where spurious memories are erased.

However, most versions of the dreaming iterative procedure studied so far in the literature encounter significant challenges. Indeed, repeating the unlearning steps excessively often leads to a point where, depending on the load α, even desired memories start deteriorating. This phenomenon results in catastrophic forgetting, where eventually no memory can be retrieved anymore unless the number of dreaming iterations is fine-tuned (Van Hemmen, Ioffe, Kühn, & Vaas, 1990; Van Hemmen & Klemmer, 1992). Additionally, the known dreaming procedures struggle to handle efficiently correlated examples (Van Hemmen, 1997).

Inspired by the concept of reinforcing the memories studied in Fachechi, Agliari, and Barra (2019), in this work we design a new iterative learning procedure that solves the major problems that limited unlearning algorithms so far. First, it can be iterated indefinitely, thus avoiding fine-tuning the early stopping. Moreover, it operates independently of any assumptions regarding the structure and correlation in the examples to be stored (at variance with methods like Der, Dotsenko, & Tirozzi, 1992; Hayashi, Hashimoto, Kimoto, & Uezu, 2018). Finally, the procedure only depends on a single parameter and keeps the conceptual simplicity of the original dreaming procedure. We call it *Daydreaming*, as it drops any distinction between night and day cycles. The key features of our learning procedure are:

1. We sample states to be unlearned as in the original procedure (Hopfield et al., 1983), by initializing randomly the model and converging to a spurious state;
2. We reinforce *multiple times* each example that we want to store. This is at variance with many examples from the literature (see for example Christos, 1996; Robins & McCallum, 1998; Van Hemmen, 1997), where they reinforce *new* memories and forget the old ones, in the same spirit as Marinari (2019) and Parisi (1986);
3. We find the right balance between these two operations, which consist in reinforcing a randomly chosen example each time we sample a spurious state to be unlearned.

It is worth stressing that the Daydreaming procedure is fully local, that is the learning of the coupling J_{ij} only depends on the values of neurons s_i and s_j (as well as the values of the patterns in i and j). This is an important observation, given that any biologically plausible learning procedure must be local, while methods based on the inversion of the pattern correlation matrix are not local (Kanter & Sompolinsky, 1987; Personnaz, Guyon, & Dreyfus, 1985).

We test Daydreaming on both uncorrelated and correlated data, the latter being generated via the so-called *random-features* (or *hidden-manifold*) model (Goldt, Mézard, Krzakala, & Zdeborová, 2020). This class of models has been proposed to generate more realistic datasets, closer to those typically used in machine learning applications (Baldassi et al., 2022; Gerace, Saglietti, Mannelli, Saxe, & Zdeborová, 2022); the random-features scheme has been used to improve linear models (see for example Louart, Liao, & Couillet, 2018; Mei & Montanari, 2022; Rahimi & Recht, 2007). Moreover, it has been shown in Negri, Lauditi, Perugini, Lucibello, and Malatesta (2023) that the Hopfield model storing such structured data has a richer phase diagram, with a new *learning phase* where the model learns to store features instead of examples (see Section 2.3 for a brief review). Therefore, random-features examples offer a good preliminary playground to test the retrieval performances of our algorithm (see Section 3). Surprisingly, we find that Daydreaming produces bigger basins of attraction for correlated data, signaling that the algorithm can *spontaneously* exploit correlations in the data. Coherently with these findings, we obtain good results also on the MNIST dataset (better than any Hopfield network that we know of, see Section 4).

The rest of the paper is organized as follows. In Section 1.1 we list all the Hopfield network learning rules that Daydreaming is related to, organizing them by which problem they address. In Section 2 we detail the Daydreaming algorithm and the model to generate synthetic correlated data. In Section 3 we report the results obtained on synthetic data (both correlated and uncorrelated). In Section 4 we describe the results for the MNIST dataset in different cases: we analyze the storage at low load, at high load, and the possible use of the model in a supervised setting. In Section 5 we discuss the results obtained, also reporting limitations of Daydreaming. Finally, in Section 6 we discuss future perspectives.

1.1. Related works

Since we stressed that Daydreaming can be iterated indefinitely, here we review the literature on alternative learning rules, arguing that rules that avoided the pitfall of "dreaming too much" showed other undesirable properties (the most common ones being the addition of non-local terms and the vanishing of the basins of attraction).

Note that the way in which we combine learning and unlearning is reminiscent of Hebbian and anti-Hebbian terms in moment-matching algorithms, with important differences in the sampling. We will mention these after we introduce our rule in Section 2.2 and discuss them in Section 5.4.

1.1.1. Learning rules with analytic expressions

Several strategies to increase the storage capacity avoid completely the use of the iterative learning procedure and rely on analytical expressions for the coupling matrix J, in the same spirit as the original Hebb rule. Many of the rules that we discuss in the following paragraph have also been found to be the fixed points of iterative procedures (although such an iterative procedure does not need to be run in practice).

The pseudo-inverse rule (Kanter & Sompolinsky, 1987; Personnaz et al., 1985) was introduced to deal with correlated data: it introduces a non-local term that makes it possible to store all the examples without error, regardless of their correlation. It is non-local in the sense that it requires the inversion of an $N \times N$ matrix, a task that cannot be done locally on a single neuron, and requires the intervention of a *deus ex machina* that knows the status of the entire network. Another problem is that this rule requires looking at all the training examples at the same time, meaning that if we want to store an additional example we should repeat the procedure from scratch. This rule reaches the maximal theoretical capacity of symmetric networks, namely $\alpha_c = 1$ (Gardner, 1988), but it presents a severe drawback: as $\alpha \to 1$ the radius of the basins of attraction goes to zero, making retrieval possible only in the trivial case of a noiseless initial condition.

In Dotsenko and Tirozzi (1991) and Dotsenko, Yarunin, and Dorotheyev (1991), the authors define a learning rule that is an interpolation between the Hebb and the pseudo-inverse rule. This rule can also be seen as the continuous limit of a certain dreaming procedure: the idea is to stop before the limit where basins of attraction vanish, in the same spirit as stopping the dreaming iterations. This rule also has a local-update version that is inspired by the learning with noise strategy suggested by Gardner, Wallace, and Stroud (1989). Note that this rule needs to be modified in case of correlated examples (Der et al., 1992).

Another interesting case of correlated examples is Hayashi et al. (2018), where the authors consider examples correlated within groups. They show that, for this specific setting, unlearning mixtures of correlated examples is an effective strategy to make the examples retrievable.

In Fachechi et al. (2019), the authors deal with the problem of dreaming indefinitely long by adding a reinforcement term in the energy. They have a parameter that interpolates between the Hebb rule and a pseudo-inverse-like rule, in the same spirit as Dotsenko et al. (1991) and Dotsenko and Tirozzi (1991), but it is not known if it is possible to converge to this rule with a local iterative update of the coupling matrix. The answer to this question was our initial motivation to the present work.

1.1.2. *Iterative learning rules*

The locality of the learning rule is a key property to make the rule biologically plausible and efficient to implement. Several modifications of the original dreaming procedure have been developed over the years to impose such a locality property. Unfortunately, they maintain the locality at the cost of other problems. For example, in Plakhov (1994), Plakhov and Semenov (1992) and Plakhov, Semenov, and Shuvalova (1995) the authors define a dreaming procedure that uses local fields instead of spin configurations. This local rule converges to the pseudo inverse rule for a large number of iterations (Kanter & Sompolinsky, 1987), which means that it has the same problem of vanishing basins of attraction. Moreover, in Nokura (1996a, 1996b) it was shown that even by modifying the rule by subtracting states sampled from the Gibbs measure at high temperature, the learning dynamics converge to a coupling matrix of the pseudo-inverse family. An iterative and local rule that can be seen as the expansion of the pseudo-inverse rule is the Storkey rule (Storkey, 1997), that was shown (Storkey & Valabregue, 1999) to realize a trade-off between the large basins of the Hebb rule and the high capacity of the pseudo-inverse rule. Finally, in Christos (1996), Marinari (2019), Parisi (1986) and Robins and McCallum (1998) the authors describe a local dreaming procedure that can be iterated indefinitely long, constantly learning new examples at the cost of forgetting the earlier ones, which means that this method does not improve the capacity of the Hebb rule. Note that also in Van Hemmen (1997) a combination of repeated learning and unlearning phases is described, but is suffers the same problem of forgetting older memories since the learning phase is applied only to new examples and it never reinforces old ones.

An example of an iterative but non-local rule can be found in Benedetti, Carillo, Marinari, and Mèzard (2023), where the authors subtract the largest eigenvector of the coupling matrix at each step of dreaming. They find that this procedure has the same performance as the classical unlearning, but has the great advantage of being much more transparent and analytically accessible.

Recently, more advanced dreaming variations, inspired by the perceptron rule, have been studied in Benedetti, Ventura, Marinari, Ruocco, and Zamponi (2022) and Benedetti and Ventura (2023). They seem to achieve optimal results but still require the fine-tuning of some parameters (similarly to the fine-tuning of the number of dreaming iterations). Furthermore, in Agliari, Aquaro, Alemanno and Fachechi (2023), the authors explicitly linked the number of dreaming iterations to regularization hyperparameters, interpreting the whole model through the lens of machine learning. Notably, they also discuss ways to fix these hyperparameters a priori.

2. Model, algorithm, and data

2.1. *Hopfield networks*

A Hopfield network is a recurrent neural network made of N neurons $\mathbf{s} = \{s_i\}_{i=1}^N$ that can be in the states ± 1 depending on the signal that they receive from all the other neurons. At each time step t, every neuron is updated according to the rule

$$s_i^{(t+1)} = \text{sign}\left(\sum_{j=1}^{N} J_{ij} s_j^{(t)}\right) , \tag{2}$$

where J_{ij} are the elements of the matrix of synaptic weights (or coupling matrix), that needs to be learned. The dynamics defined by Eq. (2) minimizes the following energy function (Hopfield, 1982)

$$H(\mathbf{s}) = \frac{1}{N} \sum_{i,j} J_{ij} s_i s_j \ . \tag{3}$$

The use of this Hamiltonian made it possible to solve the Hopfield model in the context of statistical physics of disordered systems (Amit et al., 1985; Amit, Gutfreund, & Sompolinsky, 1987a; Amit et al., 1987b): memories are interpreted as local minima of this energy landscape. When P is proportional to N, this landscape is populated by many spurious minima that can trap the dynamics defined by Eq. (2) and thus stop the retrieval of a memory if the dynamics is initialized too far from it. We consider a symmetric matrix ($J_{ij} = J_{ji}$) with zeros on the diagonal ($J_{ii} = 0$), which means we do not consider the so-called self-energies. We perform asynchronous updates, meaning that, at each time step, we update one neuron at a time in an arbitrary order. The dynamics stop when spin values reach a fixed point (or, equivalently, a local minimum of the energy, that can be either a memory or a spurious one).

This model works as an associative memory if, when initialized on a noisy version of one of the P examples $\{\xi^\mu\}_{\mu=1}^P$, the model converges to the clean version of such an example. We are interested in finding, in an efficient way, the synaptic weights that maximize the number of retrievable examples and the mean initial distance that still allows for correct memory retrieval.

2.2. *Daydreaming algorithm*

Our algorithm aims to enhance the storage capacity of the Hopfield network by simultaneously reinforcing the memories to be stored and "dreaming away" the spurious memories (i.e. local minima of the Hamiltonian not corresponding to memories to be stored). The removal part operates as in the original dreaming procedure (Hopfield et al., 1983).

At each step u in the learning process, we initialize the network to a random configuration, and we run the update rule in Eq. (2) until we reach a fixed point $\boldsymbol{\sigma}^{(u)}$. Then we increase the energy $H(\boldsymbol{\sigma}^{(u)})$ by adding $-\sigma_i^{(u)}\sigma_j^{(u)}$ to the coupling matrix. Simultaneously, we reinforce one of the memories, i.e. we choose an index $\mu(u)$ uniformly at random and we decrease the energy $H(\xi^{\mu(u)})$ by adding $\xi_i^{\mu(u)}\xi_j^{\mu(u)}$. In formulas, the Daydreaming update rule reads

$$J_{ij}^{(u+1)} = J_{ij}^{(u)} + \frac{1}{\tau N}\left(\xi_i^{\mu(u)}\xi_j^{\mu(u)} - \sigma_i^{(u)}\sigma_j^{(u)}\right) , \tag{4}$$

where τ is a timescale parameter that acts as an inverse learning rate and the factor $1/N$ ensures that the rule works well in the limit of very large networks. For the same reason, we define N steps of Daydreaming as one epoch of training, so that we have a measure of training time that is fair at any N (the time t that appears in all figures measures the number of epochs). Finally, although the update rule in Eq. (2) is invariant under a global rescaling of the coupling matrix J, we prefer to keep it well bounded, and so we normalize it at the end of every epoch.

The learning rule in Eq. (4) needs to be equipped with an initial condition. We have tried several uninformed choices — e.g. $J_{ij} = 0$ or $J_{ij} \sim \mathcal{N}(0,1)$ being $\mathcal{N}(0,1)$ the normal distribution — and all converge to a set of coupling matrices with the same retrieval properties. Just to make the learning process faster, we initialize J_{ij} according to the Hebb rule. The complete pseudo-code is reported in Alg. 1.

We designed this procedure as an iterable version of the rule described in Agliari, Alemanno, Barra, and Fachechi (2019). The reason for learning and dreaming at the same time is that we found heuristically that this is the key factor that allows indefinite iteration. The

Algorithm 1 Daydreaming learning algorithm

```
Input examples $\{\xi^\mu\}_{\mu=1}^P$
Output coupling matrix $J$
  $J_{ij} \leftarrow \frac{1}{N}\sum_\mu \xi_i^\mu \xi_j^\mu$        ▷ Initialization with the Hebb rule
  $J_{ii} \leftarrow 0$
  for $t = 1, \dots, E$ do                                   ▷ Do $E$ epochs
      for $u = 1, \dots, N$ do                               ▷ Do $N$ steps in each epoch
          $\mu \leftarrow \mathrm{Unif}(\{1, \dots, P\})$              ▷ Pick an example at random
          $s_i \leftarrow \mathrm{Unif}(\{+1, -1\})$                   ▷ Initialize spins at random
          while not converged to some fixed point $s_i = \sigma_i^{(u)}$ do
              $s_i \leftarrow \mathrm{sign}(\sum_j J_{ij} s_j)$        ▷ Run the spin update dynamics
          end while
          $J_{ij} \leftarrow J_{ij} + \frac{1}{\tau N}(\xi_i^\mu \xi_j^\mu - \sigma_i^{(u)}\sigma_i^{(u)})$   ▷ Update the coupling matrix
          $J_{ii} \leftarrow 0$
      end for
  end for
```

reason for subtracting one state at a time is that, for uncorrelated data, we know (from Amit et al., 1987b) that random initializations output with high probability a spin-glass states, which are uncorrelated with the memories and with themselves. Therefore, subtracting the contribution of multiple spurious states (like in Christos, 1996; Robins & McCallum, 1998) is less effective, as they tend to average out.

Interestingly, we obtained an update rule that resembles moment-matching algorithms, which can be derived from a maximum-likelihood principle. We will discuss this point in more detail in Section 5.4, but we note here that the key difference from moment-matching algorithms is that we sample states to be unlearned using a zero-temperature dynamics, which finds spin-glass states with high probability and therefore is intrinsically out of thermodynamic equilibrium. Moment-matching algorithms, instead, require equilibrium sampling, which is much more computationally expensive.

2.3. Datasets

We will test the Daydreaming algorithm on different datasets to measure the capacity and the mean size of basins of attraction in the Hopfield model with coupling matrix J trained with the learning rule in Eq. (4).

Apart from the random patterns dataset, where $\xi_i^\mu = \pm 1$ with equal probability, and the MNIST dataset (LeCun, Bottou, Bengio, & Haffner, 1998), we are going to use also the random features model (Goldt et al., 2020), which is defined as follows: P examples ξ^μ are generated as superpositions of D random features $\mathbf{f}^k \in \{-1, +1\}^N$, namely

$$\xi_i^\mu = \mathrm{sign}\left(\sum_{k=1}^{D} c_k^\mu f_{ki}\right), \tag{5}$$

where $c_k^\mu \sim \mathcal{N}(0, 1)$ and $f_{ki} \sim \mathrm{Unif}(\{+1, -1\})$.

This model, in addition to the usual load parameter $\alpha = P/N$, is also controlled by the parameter $\alpha_D = D/N$, which describes how strongly correlated the examples are: if $\alpha_D \gg 1$ the distribution of the examples converges to $\mathrm{Unif}(\{+1, -1\})$ and we get back a dataset of uncorrelated examples; while if $\alpha_D \lesssim 1$ the examples are correlated, and the task of storing them becomes more complicated, even if in principle the correlation among example could be exploited to increase the capacity. For instance, the storage capacity of the Hebb rule decreases with α_D. In addition to the usual *storage phase*, the Hopfield model with the Hebb rule for correlated examples also exhibits a *learning phase*: if α is larger than a critical threshold $\alpha^*(\alpha_D)$, examples are unstable and features become locally stable fixed points of the dynamics (Negri et al., 2023). Understanding whether this rich behavior of the Hopfield model is present also with coupling matrices different from the Hebbian one is an open question.

3. Results on synthetic data

Given a generic Hopfield network with coupling matrix J and assuming that the retrieval of the stored patterns is performed following the parallel update rule in Eq. (2), we need to quantify the capacity of the network and the mean size of the basins of attraction. We do this by computing the so-called *retrieval map* as follows.

First, we define the *magnetization* m^μ (or overlap) of a configuration s with a given example ξ^μ as

$$m^\mu = \frac{1}{N}\sum_{i=1}^{N} \xi_i^\mu s_i \,. \tag{6}$$

This quantity describes the correlation between the current state of the network and a memory: if $m^\mu = 1$, the state corresponds exactly to the example ξ^μ, while $m^\mu = 0$ means that the state contains no information about ξ^μ. Then, we initialize the network on a configuration that has initial magnetization m_I^μ with an example, we run the dynamics in Eq. (2) until convergence, and then we measure the final magnetization m_F^μ with the chosen example. We repeat the dynamics with several initial configurations (at given m_I^μ) and for all the examples. Finally, we average the magnetization m_F over all the examples and initial conditions. The resulting averaged curves of m_F vs. m_I are called *retrieval maps* and are shown in Figs. 2 and 4.

From the retrieval maps, one can read out a lot of useful information. For example, when the network reaches its capacity, the stored patterns (at least some of them) become locally unstable, i.e. they are no longer fixed points of the spin update dynamics. Looking at the retrieval map, one realizes that the learned examples are *locally stable* if $m_\mathrm{I} = 1$ corresponds to a $m_\mathrm{F} \simeq 1$. Moreover, the mean size of the basins of attraction can be estimated from the length of the plateau with $m_\mathrm{F} \simeq 1$. Indeed one is interested in understanding how distant from a typical pattern the starting configuration can be placed such that the spin update dynamics converge to the chosen pattern. Given that a perfect retrieval ($m_\mathrm{F} = 1$) is not required and also $m_\mathrm{F} \simeq 1$ is fine for the definition of the basin of attraction, we do not provide a precise and quantitative definition. Nevertheless, looking at the retrieval maps in Figs. 2 and 4, the presence of the plateau with $m_\mathrm{F} \simeq 1$ is very clear, and also its length can be estimated reasonably (especially if we just need to compare two different networks to classify them according to the plateau length).

3.1. Convergence

We start the analysis of the Daydreaming algorithm by studying its convergence toward the stationary asymptotic state. In Fig. 1 we show the training of the coupling/synaptic matrix J on uncorrelated data for $\alpha = 0.2$ (a value larger than the capacity of the original Hopfield model with Hebbian matrix). It is worth stressing that the Daydreaming algorithm has just one parameter, τ, which regulates the speed of training, and we are going to show that its precise value is not relevant as long as it is large enough. So, in practice, Daydreaming does not need to be fine-tuned, and it is almost a parameter-free algorithm.

The training of the coupling matrix J is a dynamical stochastic process taking place in a very high-dimensional space[2] and so we need to project it on simpler observables to study it. In Fig. 1 we plot two of such interesting observables: in the left panel we show the scaled norm of the matrix increment $\tau\|\Delta J\|_2$, where $\tau\Delta J_{ij}^{(u)} = (\xi_i^{\mu(u)}\xi_j^{\mu(u)} - \sigma_i^{(u)}\sigma_j^{(u)})/N$, and in the right panel the distance of the coupling matrix J from the Hebbian initial condition J_0. The two observables provide complementary information: the latter measures the typical distance covered by the learning dynamics, i.e. how far it is from the starting point, while the former measures the typical size of the displacement

[2] The number of independent elements in a $N \times N$ symmetric matrix with null diagonal elements is equal to $N(N-1)/2$.

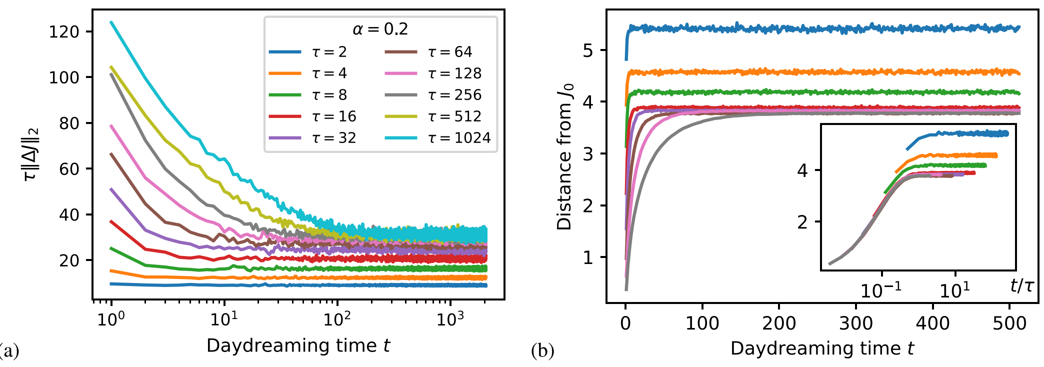

Fig. 1. The Daydreaming algorithm has a good asymptotic behavior. The training process becomes stationary after a time $t \simeq \tau$ and, when $\tau \geq 64$, the asymptotic regime does not depend on the value of τ. *Left panel:* we plot, as a function of training time t, the rescaled norm of the increment $\tau\|\Delta J\|_2$. *Right panel:* we show the distance of the coupling matrix J from the Hebbian initial condition J_0. We show curves only for $\tau \leq 256$ to be more clear. *Inset*: we plot the same curves of the large panel, now as function of t/τ to make the collapse more visible. We used $N = 500$ and $P = 100$ for these figures.

per step.

We plot the scaled displacement because it does not depend on τ, for τ large enough. In this limit both the scaled displacement and the distance from J_0 converge to a constant asymptotic value on timescales comparable with the rate τ. We are not claiming that the matrix J converges to a fixed point: the matrix J keeps changing slowly due to the stochastic nature of the learning rule, as the non-zero scaled displacement witnesses. However the learning dynamics reaches an asymptotic state, which is an ensemble concentrated on coupling matrices with similar statistical properties, that eventually produce the same retrieval maps. In any case, the experiments shown in Fig. 1 strongly suggest it is not useful to run the Daydreaming algorithm for a time much larger than τ.

More importantly, the asymptotic state reached by Daydreaming slightly depends on τ only for very small values of τ. As soon as τ is large enough (and a number of the order of a few tens of steps is enough) the results are τ-independent, making the Daydreaming algorithm free from parameters to be tuned. The practical rule of choosing a reasonably large τ (e.g. $\tau \sim 100$) and running the algorithm for a time slightly larger than τ steps seems to work well and saves the time usually dedicated to the fine-tuning of algorithm parameters.

The great advantage of the Daydreaming algorithm is that it can be iterated indefinitely because the coupling matrix J does not degrade over time and the corresponding Hopfield network maintains its retrieval capabilities, as shown from the convergence of the retrieval maps in the panel (a) of Fig. 2 and in all panels of Fig. 4.

For the above reason, the best signature for the convergence of the Daydreaming algorithm is the retrieval map becoming stationary in time. We are going to use this stronger criterion for all the different datasets studied in the present work (uncorrelated data, random features data, and MNIST data).

3.2. Retrieval of uncorrelated data

In Fig. 2 we report the results obtained by running Daydreaming on uncorrelated data. In the left panel, we show the retrieval maps for $\alpha = 0.4$ ($N = 10^3$ is used in both panels). Different colors correspond to different training times, from lighter to darker. At $t = 0$ we start from the Hebbian matrix and indeed, for short times, the network fails in retrieving the random patterns stored, as can be deduced from the fact the curve in $m_\mathrm{I} = 1$ reaches a value $m_\mathrm{F} < 1$. This is expected given that $\alpha = 0.4$ is larger than the capacity of the Hebbian coupling matrix ($\alpha_c \simeq 0.138$). During the training, the retrieval map improves, especially for large values of m_I, i.e. close to the stored patterns. Already for $t = 64$ the stored patterns become locally stable and for $t = 128 \lesssim \tau$ the retrieval map has reached its asymptotic shape, which would remain unchanged even keeping running the Daydreaming algorithm. The asymptotic shape is interesting: first of all, it is much better than the one for Hebbian couplings, with a plateau extending down to $m_\mathrm{I} \simeq 0.7$; moreover, it slightly improves the one found in Benedetti et al. (2022), which is the current state-of-the-art and was also considered optimal (see the black dashed line in Fig. 2). In other words, the Daydreaming algorithm can spontaneously modify the coupling matrix to accommodate a large number of patterns, maximizing the basin of attraction of these attractive fixed points without any fine-tuning (while the algorithms described in Benedetti et al. (2022) need a precise fine-tuning, either in the dreaming time or the perceptron threshold).

The right panel of Fig. 2 shows the converged retrieval maps obtained at several values of α, from very small values up to $\alpha = 1$. The stored patterns are stable up to $\alpha = 1$ proving that the Daydreaming algorithm can reach the maximum capacity, $\alpha_c = 1$ for uncorrelated patterns (Gardner, 1988). As expected, the size of the basins of attraction shrinks when α grows and becomes vanishing at $\alpha_c = 1$. Such a maximum capacity (with vanishing basins of attraction) has been already obtained with other algorithms implementing the dreaming procedure (Fachechi et al., 2019). However, the Daydreaming algorithm achieves these optimal performances by running a local (i.e. biologically plausible) learning rule, without the need for a rule involving the entire network (as when matrix inversions are required). Moreover, approaching the maximum capacity there are often numerical instabilities (e.g. in the inversion of the patterns correlation matrix for the pseudo-inverse rule) but we do not observe any numerical issue in the Daydreaming algorithm.

To understand how fast the Daydreaming algorithm learns the optimal retrieval maps shown in the right panel of Fig. 2, the reader can look at the data plotted with red and reddish colors in panels (a), (b), and (c) of Fig. 4. The convergence to the asymptotic state is faster when α is smaller and becomes slower approaching the maximum load. It is worth noticing — see panel (a) of Fig. 4 — that, even for low loads, when the Hebbian rule can store the pattern, the Daydreaming algorithm slightly improves over the Hebbian rule, by enlarging the basins of attraction.

Overall, we find that the Daydreaming algorithm matches the storage capacity for uncorrelated examples of state-of-the-art results (Agliari et al., 2019; Benedetti et al., 2022) (which is the highest that is theoretically possible), while it has larger basins of attraction than those achieved in Benedetti et al. (2022).

3.2.1. Comparison with the pseudo-inverse rule and the Storkey rule

In Fig. 3 we compare the coupling matrix obtained with the Daydreaming procedure to other famous learning rules, namely the pseudo-inverse rule (Personnaz et al., 1985) and the Storkey rule (Storkey, 1997). As Daydreaming and Storkey are iterative rules, we first need to converge to the stationary state and then measure some statistical properties of the asymptotic coupling matrices J. In the left panel of

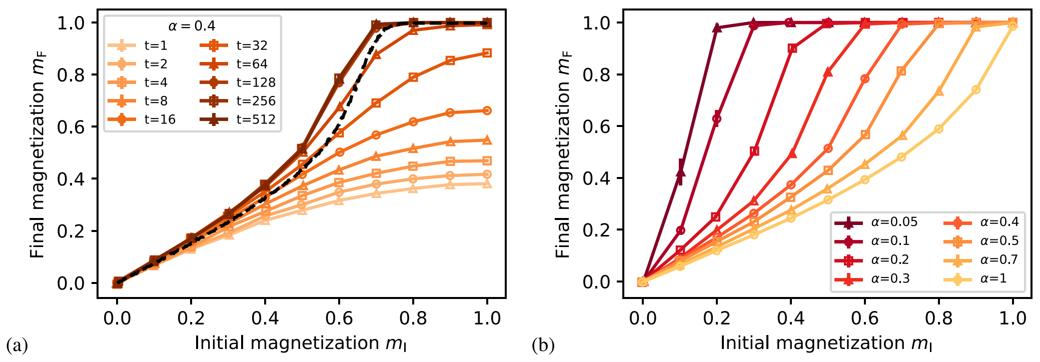

Fig. 2. The Daydreaming algorithm improves state-of-the-art performances in retrieving random data. We report the retrieval maps for uncorrelated examples. *Left panel:* we show the evolution of the retrieval map during the training with $\tau = 256$ (from lighter to darker shades) at $\alpha = 0.4$. The training converges around $t = 128 \lesssim \tau$ and the asymptotic network has better performances than the state-of-the-art results of Benedetti et al. (2022) (dashed black line). *Right panel:* we show the asymptotic retrieval maps for several values of the load α. A plateau with $m_F \simeq 1$ exists for any $\alpha < \alpha_c = 1$, i.e. the network trained with Daydreaming has a maximum capacity. We used $N = 10^3$ for these figures. Note that error bars are not always visible, since most of them are of order 10^{-2}.

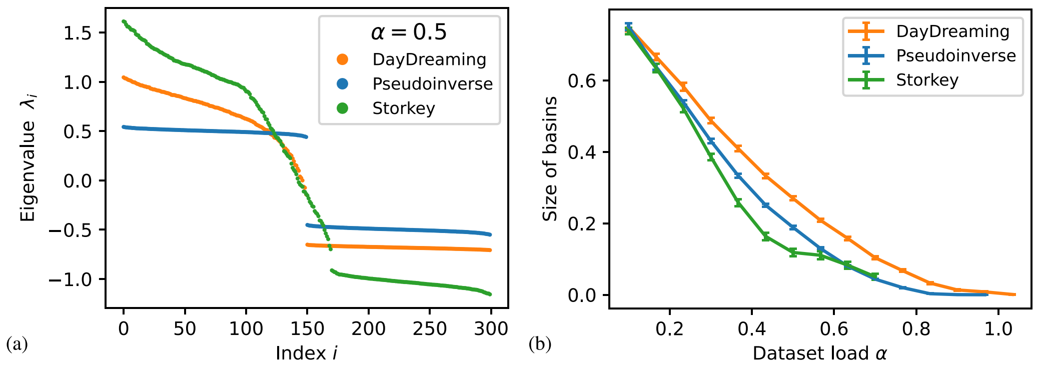

Fig. 3. The Daydreaming coupling matrix is substantially different from other rules and performs better retrieval. Left panel: we report the spectrum of eigenvalues of coupling matrices for $N = 300$ and load $\alpha = 0.5$ obtained using the Daydreaming algorithm, the Pseudo-inverse rule and the Storkey rule. Right panel: we compare mean sizes of the basins of attraction for $N = 300$ and different values of α. These are obtained by initializing the dynamics with a given magnetization m_I with respect to an example ξ^μ and measuring the highest value for which the final magnetization is $m_F > 0.99$. The value on the vertical axis corresponds to the hamming distance $d = 1 - |m_I|$. The curves are obtained by evaluating m_I from 1 to 0 with a step of 0.05 and averaging over 30 samples.

Fig. 3 we plot the eigenvalues of the coupling matrix J obtained with the 3 different learning rules. Since the pseudo-inverse rule is built exactly to make eigenvectors orthogonal, these do not interfere and eigenvalues concentrate on two well-separated values, with the number of positive eigenvalues being exactly the number of patterns to be stored. In principle, these properties look ideal, but we will see that Daydreaming does better. The spectrum of the J matrix obtained by Daydreaming has a smaller gap, but the number of eigenvalues above the gap is still exactly the number of patterns to be stored (this means that the rule is still perfectly efficient and indeed achieves maximum capacity $\alpha_c = 1$). Instead, the Storkey rule is sub-optimal as the gap is already very small at $\alpha = 0.5$ and the number of eigenvalues above the gap is larger than the number of patterns to be stored (indeed for the Storkey rule the capacity is not maximal, $\alpha_c < 1$).

More interesting is the right panel of Fig. 3 where we report the average size of the basins of attraction in the entire range of $\alpha \in [0, 1]$. The Storkey rule produces the smallest (i.e. worst) basins of attraction. Moreover, it cannot be used for $\alpha \gtrsim 0.7$ due to numerical instabilities. The pseudo-inverse rule, notwithstanding the "ideal" spectrum, has smaller basins of attraction than the coupling matrix learned via the Daydreaming algorithm for any α value. We also remind the reader that approaching $\alpha = 1$ the pseudo-inverse rule cannot be used due to the numerical instabilities related to the almost singular matrix to be inverted, while Daydreaming works in a very smooth way.

3.3. Retrieval of correlated data

In panels (a), (b), and (c) of Fig. 4 we show how the retrieval map changes during the training with the Daydreaming algorithm for uncorrelated data (red triangles) and for correlated data generated by the hidden features model (blue squares). The plotted results have been obtained with $N = 10^3$, and values of α and α_D shown in the legend. Different shades of the colors represent different training times: the lightest color is $t = 1$ and the darkest is $t = 32\,768$ (logarithmic spacing). We observe that the convergence is faster for smaller α values and slower for larger α values. In any case, the Daydreaming algorithm converges well and without the need for any fine-tuning of parameters, even when it is run at the network's maximum load ($\alpha = 1$ in the case of uncorrelated examples).

Even for a low load — see panel (a) of Fig. 4, where $\alpha = 0.1$ — we see a clear improvement of the Daydreaming algorithm over the Hebbian rule. For uncorrelated data, we observe the formation of a slightly larger basin of attraction. For correlated data, with the Hebbian coupling matrix, the examples are unstable and get spontaneously stabilized by the Daydreaming algorithm. At convergence, the correlated examples have large basins of attraction than uncorrelated pattern.

Panel (b) of Fig. 4 shows results for a medium load ($\alpha = 0.2$). In this case, the dreaming procedure is strictly required to have stable examples. Although the learning from uncorrelated examples converges faster, the coupling matrix J trained on correlated examples eventually has larger basins of attraction. This is quite surprising and it means the Daydreaming algorithm has the right capabilities to extract useful

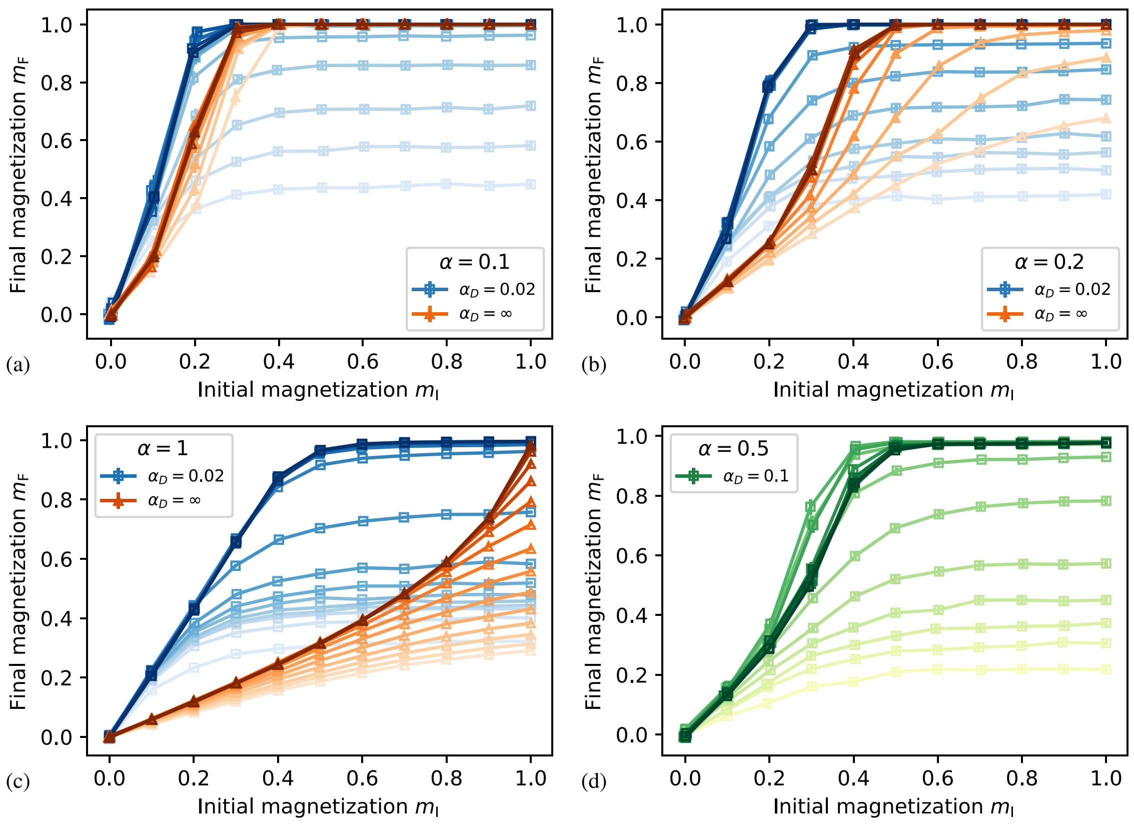

Fig. 4. For the Daydreaming algorithm, correlated examples are easier to store and retrieve than random ones. We show the retrieval maps during the training process with the Daydreaming algorithm. In panels (a–c), blue squares are for correlated data, while red triangles are for uncorrelated data. Different shades of the colors represent different training times: the lightest color is $t = 1$ and the darkest is $t = 32\,768$ (logarithmic spacing). Panel (a) shows results for $\alpha < 0.138$, where Daydreaming enlarges the basin of attraction of uncorrelated examples and stabilizes the correlated examples. Panel (b) shows results for $\alpha > 0.138$, where uncorrelated data become stable faster but correlated data end up with a larger basin of attraction. Panel (c) shows results for $\alpha = 1$, where uncorrelated data become stable attractors at the end of the training but with vanishing small basins of attraction; instead, correlated data have large basins of attraction. Panel (d) shows the retrieval map for the features hidden in the data, that get stabilized as well by the Daydreaming algorithm. We used $N = 10^3$ for these figures.

correlations from a dataset. And it can do it by itself, without being specifically instructed.

Finally panel (c) of Fig. 4 report results for the case of a large load ($\alpha = 1$). Although the learning dynamic is slower, the Daydreaming algorithm converges to very good retrieval maps. For uncorrelated data, the retrieval map reaches the point ($m_I = 1, m_F = 1$) with a vanishing plateau, meaning we are running at the maximum load $\alpha_c = 1$. Instead, the basin of attraction for correlated examples is still very large at $\alpha = 1$, meaning that the retrieval will also be possible for $\alpha > 1$. This implies that the Hopfield model endowed with the Daydreaming training process can exploit the correlation to overcome the storage bound of uncorrelated examples. Daydreaming extends the storage limit of correlated examples described in Negri et al. (2023), given that all the values of α and α_D reported in panels (a–c) of Fig. 4 are outside the storage phase of Negri et al. (2023). The basin of attraction of correlated examples vanishes around $\alpha = 2$, even though the shape of the retrieval map is much less steep than the one for uncorrelated examples (see Fig. A.2 in the appendix).

3.4. Retrieval of features

In Negri et al. (2023) it has been shown that the Hebb rule applied to correlated data can stabilize (and thus retrieve) the hidden features in the so-called learning phase ($\alpha_d < 0.138$ and α large enough). A natural question is whether the same happens if the coupling matrix J is learned via the Daydreaming algorithm. To this end, we define a *feature magnetization* $\mu_k = \frac{1}{N}\sum_{i=1}^{N} f_{ki}s_i$ and measure the feature retrieval map, plotted in panel (d) of Fig. 4 for different training times.

We notice that, for the values $\alpha = 0.5$ and $\alpha_D = 0.1$ used, the Hebbian rule is not able to make the features locally stable ($\mu_F < 1$ for $\mu_I = 1$). Instead, the Daydreaming algorithm spontaneously stabilizes the features and produces large basins of attraction around the hidden features. We do not have an explanation of why the optimal shape seems to be achieved at a finite training time and further studies are needed to answer this question.

In general, we can affirm that learning the coupling/synaptic matrix J via the Daydreaming algorithm strongly enhances the stability of (and thus the possibility of retrieving) the correlated examples stored by the network, as well as the hidden features that generated those examples.

4. Results on the MNIST dataset

Given the excellent performances of the Daydreaming algorithm in training the coupling matrix J from correlated data, it is natural to explore its behavior when the algorithm is fed with highly structured data. To this aim, we used the MNIST dataset (LeCun et al., 1998), the most widely used dataset in machine learning. We train the model with Daydreaming on a deskewed version of the MNIST dataset: the images are deformed and rotated so that they are all centered and aligned vertically. Moreover, the images are cropped to the central area of size 14×14 pixels and made binary (with values ± 1) using a threshold of 86. This whole procedure was suggested in Belyaev and Velichko (2020). Finally, we split the dataset into a balanced training set (where all the ten digits have the same frequency), from which we choose examples to feed the Daydreaming algorithm, and a test set, where we keep examples that remain unseen during training.

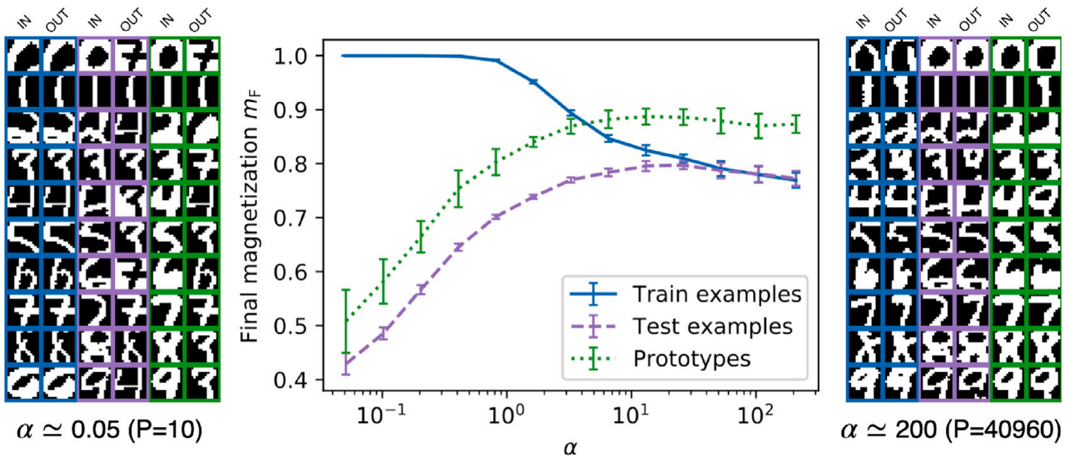

Fig. 5. Attractors close to prototypes emerge at high α when we train via Daydreaming on MNIST. *Central panel:* We show how the stability of examples evolves as a function of α. Local stability is quantified as the overlap m_F between the final state of a dynamics initialized with a specific example and the example itself. When α is small enough, the train images are the only stable examples. When α is sufficiently high, the curves for train and test data coincide, indicating uniform treatment of seen and unseen images (generalization). Moreover, prototype images, defined as the averages of the training images for each digit, exhibit higher stability in this region. *Lateral panels:* we show examples of the initial and final configurations of the dynamics for low and high load (left and right respectively). The colors represent training images, test images, and prototypes as in the central panel. For each couple of images, the left one is the input and the right one is the output. For this plot, we used $t = 2^{20}$ and $\tau = 64$ for each value of α and a threshold $J_{\max} = 0.5$ for the absolute value of the couplings.

Let us start by discussing a caveat of the Daydreaming algorithm when applied to the MNIST dataset for very high loads ($\alpha \geq 100$) and a simple modification to cure it. Consider a pair of pixels i and j such that $\xi_i^\mu \xi_j^\mu = 1$ for all the training examples (this happens often for the pixels on the boundaries that take always the value of the background color). This strong correlation in the data implies a strong coupling in the trained matrix J. Correctly the Daydreaming algorithm tries to assign $J_{ij} = \infty$ to force pixels in i and j to take always the same value. Unfortunately, if some matrix elements become extremely large, when the matrix is normalized, the remaining elements become very small and tend to vanish, thus erasing the relevant information stored in the coupling matrix. We have solved this problem by just introducing a threshold $J_{\max}$ on the maximum absolute value that a single coupling can assume. In the presence of a threshold value $J_{\max}$ for the couplings, it is important to properly normalize the coupling matrix in a way that the norm of J does not depend on the load α. Thus we chose to normalize the matrix obtained with the Hebb rule (i.e. the initial matrix of Daydreaming) by $P = \alpha N$ instead of N. We used a threshold $J_{\max} = 0.5$ for any value of α. This proved sufficient to prevent the collapse of the matrix. We also checked that the results do not depend on the precise value of $J_{\max}$ in a range $[0.25, 1.25]$.

We summarize in Fig. 5 the results for the Daydreaming algorithm trained with the MNIST dataset described above. The central panel of Fig. 5 shows m_F obtained starting from $m_I = 1$ with examples from the training dataset (blue continuous line), examples from the testing dataset (purple dashed line), and prototypes (green dotted line). A prototype is defined as the average of all the training images corresponding to the same digit (more details are provided below). The lateral panels provide typical examples of initial (IN) and final (OUT) configurations of the Hopfield spin update dynamics. Color codes are like in the central panel. Left and right panels are for low and high loads respectively. These results are discussed below.

4.1. Low load

We start commenting Fig. 5 from the small α region. Training examples are stable (i.e. $m_F \simeq 1$) for $\alpha < \alpha_c \simeq 0.8$, where the critical load α_c is identified as the point where the magnetization of the training examples starts decreasing (solid blue curve). This result is supported by Fig. A.1 in the Appendix, where we show the retrieval maps obtained for the MNIST dataset at several values of α: the size of the basins of attraction shrinks to zero around $\alpha \simeq 0.8$. The model perfectly stores training examples until it reaches a threshold between $P = 80$ and 160 (corresponding to α between 0.4 and 0.8) which is comparable with recent results (Nouri & Seyyedsalehi, 2023), where the authors use a much more complicated attractor network to store 100 MNIST examples. Note that they use full-resolution images and therefore more parameters than our model (their N is larger than ours), which means that the Daydreaming algorithm manages to store examples at a higher effective load α.

The results achieved by the Daydreaming algorithm not only are comparable with the state-of-the-art (Nouri & Seyyedsalehi, 2023) but also go beyond expectation. Indeed several modifications to the Hopfield model have been recently proposed based on the common idea that the standard Hopfield model is very inefficient at storing and retrieving structured data (as the MNIST dataset). Our results prove this is not the case if the coupling matrix is learned through a smart algorithm like the Daydreaming one.

In the left panel of Fig. 5 we can see that the model correctly recognizes only examples from the train set (blue boxes), and it makes mistakes when presented with examples from outside the train set (purple and green boxes). This is consistent with the behavior of the Hopfield model in the storage phase.

4.1.1. Classification

In the regime of low load, when the model is presented with images unseen during the training, the dynamics converge to one of the stored memories (i.e. one of the training images), usually the closest one, and is not trapped in spurious minima. This can be appreciated in the left panel of Fig. 5, where all the output images correspond to one of the training images (those in the blue columns). Inspired by this behavior and by the results presented in Belyaev and Velichko (2020), where a different learning method is used, we try to use the Hopfield model equipped with the Daydreaming algorithm for digit classification.

When the labels of the images are available, there are two main ways to construct the coupling matrix using the Hebb rule, as summarized in Alemanno, Aquaro, Kanter, Barra, and Agliari (2023):

- The supervised Hebbian learning, where label information is used to group data points into their respective classes, and the Hebb rule is then applied to the mean of each class.
- The unsupervised Hebbian learning, where labels are disregarded, and no grouping occurs before applying the Hebb rule.

Similarly, we define supervised and unsupervised versions of the Daydreaming algorithm, depending on whether data regrouping is performed before the algorithm is applied. We remark that in both cases we are using information obtained using all the patterns in the training dataset, but only the supervised version uses the extra information coming from the labels. For this reason, we decide to proceed with this approach. As a first step, we define the prototypes $\boldsymbol{\pi}^C$ for each class C (corresponding to each of the ten digits) as the binarized averages of the images in the training set within that class: $\boldsymbol{\pi}^C = \text{sign}\left(\mathbb{E}_{\mu\in C}\, \boldsymbol{\xi}^\mu\right)$. We then proceed to train the model exclusively on these ten patterns and evaluate its performance on the test set. The label predicted by the model for a test image corresponds to the prototype that corresponds to the final state of the dynamics initialized with that test image. Instead, if the dynamics converge to a fixed point different from all prototypes (spurious pattern), then no label is assigned to that image. For the MNIST dataset, we obtain a test accuracy (i.e. a fraction of correctly assigned label) of about 67.5%, which is higher than the accuracy found in Belyaev and Velichko (2020) (61.5%). For completeness, we also conducted a numerical analysis to evaluate the performance of the unsupervised learning version, where all labels are disregarded, and the Daydreaming method is directly applied to the images without any prior regrouping. Prototypes are still used to evaluate classification performance, resulting in an accuracy of approximately 66.2%. While this is slightly lower than the accuracy achieved with supervised learning, it remains an improvement over the results reported in Belyaev and Velichko (2020). More in-depth results are reported in Table A.1 in the Appendix.

4.2. High load

If we run the Daydreaming algorithm on the MNIST database with many more training examples (large α regime), we observe that the training examples are no longer locally stable. Instead, the dynamics initialized on a training image end in a fixed point whose magnetization m_F decreases as we increase P or α. At the same time, if we initialize the model in an example taken from the test set, we find that the final magnetization increases with P or α (purple dashed line in Fig. 5). Interestingly enough, for $P \geq 10240$ ($\alpha \geq 52.2$) the train and test magnetizations become equal, signaling that the network is acting similarly on seen and unseen examples (the generalization property).

To understand what the new attractors of the model might look like at high loads, we consider the prototypes defined above. Starting the dynamics on one of the prototypes, we observe that the curve for the final magnetization follows the same trend as the one for the test examples, increasing with α (green dotted line in Fig. 5). This curve is also systematically higher than that of test magnetization. This indicates that the new attractors of the model are closer to prototypes than to examples.

In the right panel of Fig. 5 we see that the model does a good job at recognizing both examples from the train set (blue boxes) and examples from outside the train set (purple and green boxes). This is consistent with a generalization behavior.

5. Discussion

To overcome a series of problems that afflicted the dreaming algorithms used to increase the storage capacity of Hopfield networks, we have designed a new learning procedure called Daydreaming. During the training process, the Daydreaming algorithm modifies the matrix J of synaptic couplings according to the rule in Eq. (4) that has the double effect of stabilizing the patterns to be stored in the network and destabilizing the spurious attractors, thus increasing the basin of attraction of the formers.

5.1. Advantages and limitations

Daydreaming proved to be a compact, straightforward, and streamlined algorithm, with the convergence rate notably depending only on the parameter τ, which can be fixed with a large degree of freedom since the results depend on it very mildly. The Daydreaming algorithm does not suffer the problem of dreaming too much and does not require any fine-tuning, thus solving the main limitation of previously known dreaming algorithms. Moreover, it seemingly does not require any assumption on the structure of the data, as it finds large basins of attraction even for highly correlated random-features examples.

Nonetheless, Daydreaming still has the same high computational cost of other unlearning procedures and of pseudo-inverse rules. This translates into a long training time, which can be a limitation at high values of N and α (the number of epochs necessary to converge grows with α).

Each step of Daydreaming requires the dynamics to converge from a random initial condition to a local minimum of the Hamiltonian. The cost for this dynamics can be computed as follows: a single spin update costs $O(N)$, a global update of all spins costs $O(N^2)$, and the number of global updates to converge slightly grows with the problem size as $O(N^\delta)$ with the small exponent δ depending on both the α value and the coupling matrix J. Furthermore, we need N steps of Daydreaming to perform an epoch and the convergence of the learning process is achieved with a finite number of epochs (of the order of τ). Therefore, the computational complexity of Daydreaming is $O(N^{3+\delta})$.

We compare this complexity with the one of the rules discussed in the introduction: the iterative ones have the same complexity as Daydreaming, barring the fact that if they are run too much they destroy retrieval; the analytical rules instead typically involve an inversion of a matrix, which is an operation with complexity $O(N^3)$.

The other main limitation of Daydreaming is that a regularization is needed in the extreme case of very large loads (e.g. $\alpha > 100$ for MNIST). However, the use of a maximum value for the coupling matrix does not increase the computational complexity of the algorithm.

5.2. Effectiveness on synthetic correlated data

The fact that Daydreaming finds large basins of attraction even for highly correlated random-features examples is somewhat surprising, as the classical picture in the literature is that correlation hinders retrieval (Amit et al., 1987a; Der et al., 1992; Fontanari & Theumann, 1990; Löwe, 1998; Van Hemmen, 1997). Another surprising result is that correlated examples have larger basins of attraction than uncorrelated examples, and can be retrieved even above $\alpha = 1$, which is the hard limit for uncorrelated examples (Gardner, 1988).

The above observations lead us to conjecture that the Daydreaming algorithm can automatically detect and exploit the correlation in the data thus improving the storage performances. This is supported by the fact that, in the case of synthetic data generated via the random-features model, the Daydreaming algorithm improves also the retrieval of the features. Note that this fact is again highly non-trivial, since the update rule in Eq. (2) ignores the structure of the hidden features that generated the examples. However, in the process of storing and reinforcing the examples the Daydreaming algorithm also learns the underlying hidden structure.

5.3. Non-trivial attractors on MNIST

By testing the Daydreaming algorithm on the MNIST dataset we have shown a proof of concept of how the exploitation of the hidden features might be relevant also for realistic datasets: new attractors emerge that are not exactly related to the training examples but rather to their underlying structure.

We have shown that the Daydreaming algorithm can perfectly store a large number of correlated examples (surprisingly large compared to recent results Agliari, Aquaro et al., 2023; Nouri & Seyyedsalehi, 2023). This is the storage phase.

For larger loads, the storage phase finishes and one could expect a catastrophic forgetting induced by the loss of the local stability of the training examples. Indeed when the training examples are too many it is impossible for a Hopfield network to have attractors on each of them. Surprisingly enough, in this high-load regime, a new non-trivial structure of basins of attraction emerges.

In the high-load regime, the model trained with the Daydreaming algorithm develops attractors that have a pretty large overlap ($\simeq 0.75$) with both training and testing examples (continuous blue and dashed purple curves in Fig. 5). The fact that the network responds in the same way when initialized on both training and testing examples is noteworthy, suggesting that the model approaches some sort of generalization. Moreover, the observation that class prototypes are even closer to the attractors (overlap $\simeq 0.85$, green dotted curve in Fig. 5) indicates that the emergent attractors outside the storage phase are related to the hidden structure in the data (similarly to the random-feature case).

If we interpret the training examples as noisy versions of their class prototype, the high-load energy landscape produced by the Daydreaming algorithm is reminiscent of what has been found in Agliari, Aquaro et al. (2023) and Alemanno et al. (2023). A noteworthy difference between those works and the Daydreaming algorithm is that the latter is completely unsupervised, as it finds the class prototypes without using the class labels, at variance to the so-called supervised Hebb rule that instead groups data according to the class label.

Also note that, in Agliari, Aquaro et al. (2023), the authors need to select a finite, fine-tuned value for the dreaming time. Instead, the Daydreaming procedure can be iterated indefinitely even when training on the MNIST dataset. We believe that the reason behind this difference might be related to the sampling procedures in the two algorithms: Daydreaming is essentially a non-equilibrium process. Further investigations on this point are left for the future.

To understand qualitatively how good the emergent attractors are, in the right panel of Fig. 5 we have shown some examples: the output images appear as slight deformations of the inputs, but the nature of the digits appears to be preserved in most cases – even for test examples and prototypes, which never appeared explicitly in the training phase. This is in stark contrast with the low-load regime, where the model perfectly stores train examples but fails on test examples and prototypes. More importantly, we do not observe the dynamics converging to spurious states, since these have been mostly removed by the Daydreaming procedure. So most of the test images do converge to meaningful attractors that can be eventually used for memory retrieval and classification. Unfortunately, we are not aware of any benchmark using the Hopfield model in this regime to compare with our results.

5.4. Relation to moment-matching rules

Daydreaming is closely related to the maximum likelihood principle, as its update rule resembles a way to satisfy a moment-matching condition: see for example (MacKay, 2003), where the "day" and "night" terms are explicitly identified, and also algorithms of the contrastive divergence family (Carreira-Perpiñán & Hinton, 2005), that are commonly used to train Restricted Boltzmann Machines (for some reviews, see for example Decelle & Furtlehner, 2021 or LeCun, Bengio, & Hinton, 2015). In this spirit, some learning rules related to ours (but less effective) have been discussed in Baldassi, Gerace, Saglietti, and Zecchina (2018) and Kojima, Nonaka, and Da-Te (1995) for a fully-connected symmetric model and generalized to sparse and asymmetric models in Braunstein, Ramezanpour, Zecchina, and Zhang (2011) and Saglietti, Gerace, Ingrosso, Baldassi, and Zecchina (2018).

The key difference between Daydreaming and all the mentioned moment-matching algorithms is how we sample the states to be unlearned: initializing the system to a random configuration and using Eq. (2) corresponds to zero-temperature dynamics, whereas moment-marching requires sampling from the model at thermodynamic equilibrium (which is hard to sample for multimodal distributions) or some out-of-equilibrium stochastic process like those used in contrastive divergence. Whether our zero-temperature sampling can be related to a form of contrastive diverge is an interesting question that we leave for the future.

Note that a zero-temperature sampling was also studied in Pöppel and Krey (1987), but instead of initializing the dynamics at random they use noisy versions of the training examples. This choice appears to produce smaller basins of attraction than our Daydreaming algorithm.

6. Perspectives

Given that at high loads we found indications of a rich and meaningful structure of basins of attraction, an extensive study of such structure seems promising. In particular, it is crucial to understand the mechanism beyond the formation of this rich structure and how the Daydreaming algorithm can automatically extract hidden information from data, in the same spirit to what was found in Kalaj et al. (2024) where the concept of retrieval in generalized to previously unseen examples.

Given the surprising results of Daydreaming on correlated data and its similarity to the well-known method to train Boltzmann Machines, it would be interesting to test it on more challenging datasets that would require higher-order interactions in the model. It would be interesting to interpret and discuss the Daydreaming algorithm as an out-of-equilibrium process (similarly to Agoritsas, Catania, Decelle, and Seoane (2023) and Decelle, Furtlehner, and Seoane (2021) in the case of Restricted Boltzmann Machines). The analogy with Boltzmann Machines also opens the door for supervised schemes such as the one presented in Scellier and Bengio (2017), where some of the nodes are interpreted as output, and the training is performed by clamping them to the desired labels. Overall, to compare our results to Boltzmann machines we would first need to extend our Daydreaming procedure to finite-temperature sampling.

Another interesting possibility is to extend Daydreaming to sparse and non-symmetric coupling matrices, which are biologically very relevant (Baldassi et al., 2018; Saglietti et al., 2018). The sparse case looks particularly promising from the perspective of the computational cost. Indeed, by reducing the computational cost for a single spin update from $O(N)$ to $O(1)$ the algorithm could be applied to much larger problems.

Noticing that the Daydreaming learning rule produces a coupling matrix which is not fixed, but changes in time taking values inside an ensemble of very good (in statistical sense) matrices, one may wonder how rapidly and efficiently such an asymptotic ensemble adapts to a dataset changing over the time. We expect the Daydreaming learning rule could be a valid algorithm for the continual learning problem (Hadsell, Rao, Rusu, & Pascanu, 2020; Wang, Zhang, Su, & Zhu, 2024).

CRediT authorship contribution statement

Ludovica Serricchio: Software, Methodology, Investigation. **Dario Bocchi:** Software, Methodology, Investigation. **Claudio Chilin:** Software, Methodology, Investigation. **Raffaele Marino:** Supervision, Methodology, Conceptualization. **Matteo Negri:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Methodology, Conceptualization. **Chiara Cammarota:** Supervision, Methodology, Conceptualization. **Federico Ricci-Tersenghi:** Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

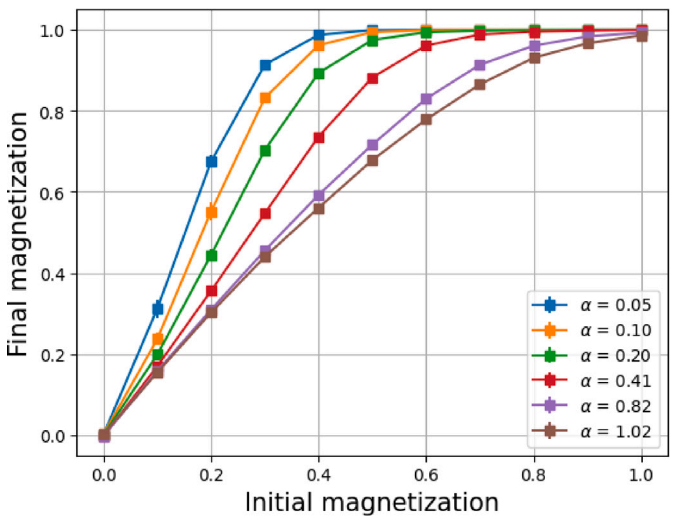

Fig. A.1. The Daydreaming algorithm can perfectly store patterns from MNIST for low loads. We show retrieval maps at the end of the training procedure for various values of the load α for the MNIST dataset. We can observe how, up to $\alpha = 0.4$, the model develops finite attraction basins corresponding to the memories, while further increasing alpha causes the patterns to progressively lose stability until they become unstable in the high-load regime. We used 14×14 images and $t = 2^{20}$ for each value of α for this figure.

Table A.1
Classification results for MNIST images using Daydreaming: MCE stands for Most Common Error. An image is classified as a spurious pattern if it does not coincide with any of the prototypes. Uncertainties are estimated using the empirical standard deviation between results obtained from 30 independent runs.

Digit	Correct class (%)	Incorrect class (%)	MCE	Spurious pattern (%)
0	73.0 ± 3.8	24.8 ± 3.9	6	2.2 ± 0.8
1	96.1 ± 1.4	3.6 ± 1.1	0	0.3 ± 0.6
2	51.6 ± 5.2	45.4 ± 5.2	0	3.0 ± 1.1
3	66.0 ± 5.4	31.8 ± 5.0	1	2.3 ± 0.7
4	54.7 ± 8.1	43.8 ± 7.9	9	1.5 ± 0.8
5	60.2 ± 4.7	38.1 ± 4.5	3	1.7 ± 0.7
6	68.2 ± 3.6	29.2 ± 3.5	0	2.6 ± 0.8
7	79.8 ± 4.2	19.0 ± 4.2	1	1.2 ± 0.9
8	54.9 ± 5.9	42.9 ± 5.4	9	2.1 ± 0.9
9	65.7 ± 7.1	33.4 ± 6.9	7	0.9 ± 0.6

Acknowledgments

We thank Enrico Ventura for pointing out a lot of interesting literature. CC and MN acknowledge the support of PNRR MUR, Italy project PE0000013-FAIR. MN also acknowledges LazioInnova - Regione Lazio under the program Gruppi di ricerca 2020 - POR FESR Lazio 2014–2020, Project NanoProbe (Application code A0375-2020-36761), for its support until July 31st 2023. RM is supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), Italy, National Recovery and Resilience Plan (NRRP), Italy, project MNESYS (PE0000006) “A Multiscale integrated approach to the study of the nervous system in health and disease” (DR. 1553 11.10.2022). FRT acknowledges financial support from ICSC – Italian Research Center on High-Performance Computing, Big Data, and Quantum Computing, funded by the European Union – NextGenerationEU.

Appendix

See Figs. A.1 and A.2.

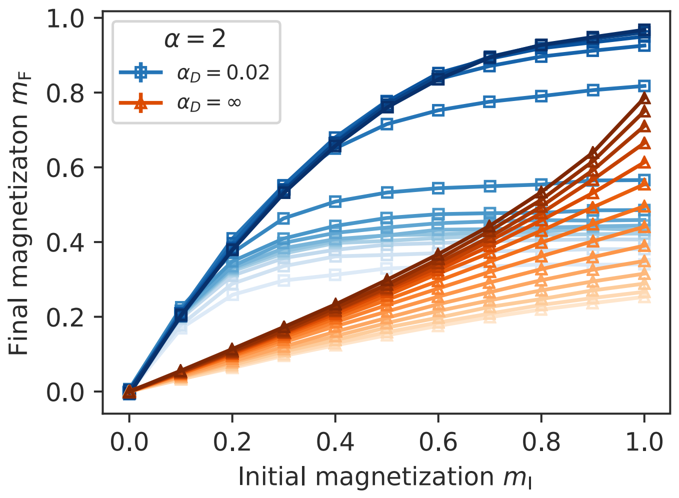

Fig. A.2. The basin of attraction of correlated examples vanish around $\alpha = 2$. Blue squares are for correlated data, while red triangles are for uncorrelated data. Different shades of the colors represent different training times: the lightest color is $t = 1$ and the darkest is $t = 32\,768$ (logarithmic spacing). We used $N = 10^3$ for this figure.

Data availability

Data will be made available on request.

References

Agliari, E., Albanese, L., Alemanno, F., Alessandrelli, A., Barra, A., Giannotti, F., et al. (2023). Dense Hebbian neural networks: a replica symmetric picture of supervised learning. *Physica A. Statistical Mechanics and its Applications*, Article 129076.

Agliari, E., Alemanno, F., Barra, A., & Fachechi, A. (2019). Dreaming neural networks: rigorous results. *Journal of Statistical Mechanics: Theory and Experiment*, *2019*(8), Article 083503.

Agliari, E., Aquaro, M., Alemanno, F., & Fachechi, A. (2023). Regularization, early-stopping and dreaming: a hopfield-like setup to address generalization and overfitting. arXiv preprint arXiv:2308.01421.

Agoritsas, E., Catania, G., Decelle, A., & Seoane, B. (2023). Explaining the effects of non-convergent MCMC in the training of Energy-Based Models. In *International conference on machine learning* (pp. 322–336). PMLR.

Alemanno, F., Aquaro, M., Kanter, I., Barra, A., & Agliari, E. (2023). Supervised hebbian learning. *Europhysics Letters*, *141*(1), 11001.

Amit, D. J. (1989). *Modeling brain function: The world of attractor neural networks*. Cambridge University Press.

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

Amit, D. J., Gutfreund, H., & Sompolinsky, H. (1987a). Information storage in neural networks with low levels of activity. *Physical Review A*, *35*(5), 2293.

Amit, D. J., Gutfreund, H., & Sompolinsky, H. (1987b). Statistical mechanics of neural networks near saturation. *Annals of Physics*, *173*(1), 30–67.

Baldassi, C., Gerace, F., Saglietti, L., & Zecchina, R. (2018). From inverse problems to learning: a statistical mechanics approach. *vol. 955*, In *Journal of physics: conference series*. IOP Publishing, Article 012001.

Baldassi, C., Lauditi, C., Malatesta, E. M., Pacelli, R., Perugini, G., & Zecchina, R. (2022). Learning through atypical phase transitions in overparameterized neural networks. *Physical Review E*, *106*(1), Article 014116.

Belyaev, M., & Velichko, A. (2020). Classification of handwritten digits using the hopfield network. *vol. 862*, In *IOP conference series: materials science and engineering*. IOP Publishing, Article 052048.

Benedetti, M., Carillo, L., Marinari, E., & Mèzard, M. (2023). Eigenvector dreaming. arXiv preprint arXiv:2308.13445.

Benedetti, M., & Ventura, E. (2023). Training neural networks with structured noise improves classification and generalization. arXiv preprint arXiv:2302.13417.

Benedetti, M., Ventura, E., Marinari, E., Ruocco, G., & Zamponi, F. (2022). Supervised perceptron learning vs unsupervised hebbian unlearning: Approaching optimal memory retrieval in Hopfield-like networks. *Journal of Chemical Physics*, *156*, Article 104107. http://dx.doi.org/10.1063/5.0084219.

Braunstein, A., Ramezanpour, A., Zecchina, R., & Zhang, P. (2011). Inference and learning in sparse systems with multiple states. *Physical Review E*, *83*(5), Article 056114.

Carreira-Perpiñán, M. Á., & Hinton, G. (2005). On contrastive divergence learning. In R. G. Cowell, & Z. Ghahramani (Eds.), *Proceedings of machine learning research*: *vol. R5*, *Proceedings of the tenth international workshop on artificial intelligence and statistics* (pp. 33–40). PMLR, URL: https://proceedings.mlr.press/r5/carreira-perpinan05a.html, Reissued by PMLR on 30 March 2021.

Christos, G. A. (1996). Investigation of the Crick-Mitchison reverse-learning dream sleep hypothesis in a dynamical setting. *Neural Networks*, *9*(3), 427–434. http://dx.doi.org/10.1016/0893-6080(95)00072-0, URL: https://www.sciencedirect.com/science/article/pii/0893608095000720.

Crick, F., & Mitchison, G. (1983). The function of dream sleep. *Nature*, *304*, 111–114.

Decelle, A., & Furtlehner, C. (2021). Restricted Boltzmann machine: Recent advances and mean-field theory. *Chinese Physics B*, *30*(4), Article 040202.

Decelle, A., Furtlehner, C., & Seoane, B. (2021). Equilibrium and non-equilibrium regimes in the learning of restricted Boltzmann machines. *Advances in Neural Information Processing Systems*, *34*, 5345–5359.

Demircigil, M., Heusel, J., Löwe, M., Upgang, S., & Vermet, F. (2017). On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, *168*, 288–299.

Der, R., Dotsenko, V., & Tirozzi, B. (1992). Modified pseudo-inverse neural networks storing correlated patterns. *Journal of Physics A (Mathematical and General)*, *25*(10), 2843.

Dotsenko, V., & Tirozzi, B. (1991). Replica symmetry breaking in neural networks with modified pseudo-inverse interactions. *Journal of Physics A (Mathematical and General)*, *24*(21), 5163.

Dotsenko, V., Yarunin, N., & Dorotheyev, E. (1991). Statistical mechanics of Hopfield-like neural networks with modified interactions. *Journal of Physics A (Mathematical and General)*, *24*(10), 2419.

Fachechi, A., Agliari, E., & Barra, A. (2019). Dreaming neural networks: forgetting spurious memories and reinforcing pure ones. *Neural Networks*, *112*, 24–40.

Fontanari, J. F., & Theumann, W. (1990). On the storage of correlated patterns in Hopfield's model. *Journal de Physique*, *51*(5), 375–386.

Gardner, E. (1987). Multiconnected neural network models. *Journal of Physics A (Mathematical and General)*, *20*(11), 3453.

Gardner, E. (1988). The space of interactions in neural network models. *Journal of Physics A (Mathematical and General)*, *21*(1), 257.

Gardner, E., Wallace, D., & Stroud, N. (1989). Training with noise and the storage of correlated patterns in a neural network model. *Journal of Physics A (Mathematical and General)*, *22*(12), 2019.

Gerace, F., Saglietti, L., Mannelli, S. S., Saxe, A., & Zdeborová, L. (2022). Probing transfer learning with a model of synthetic correlated datasets. *Machine Learning: Science and Technology*, *3*(1), Article 015030.

Goldt, S., Mézard, M., Krzakala, F., & Zdeborová, L. (2020). Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, *10*(4), Article 041044.

Hadsell, R., Rao, D., Rusu, A. A., & Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, *24*(12), 1028–1040.

Hayashi, K., Hashimoto, C., Kimoto, T., & Uezu, T. (2018). Unlearning of Mixed States in the Hopfield Model —Extensive Loading Case—. *Journal of the Physical Society of Japan*, *87*(5), Article 054004. http://dx.doi.org/10.7566/JPSJ.87.054004, URL: https://journals.jps.jp/doi/abs/10.7566/JPSJ.87.054004, Publisher: The Physical Society of Japan.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. http://dx.doi.org/10.1073/pnas.79.8.2554.

Hopfield, J. J., Feinstein, D. I., & Palmer, R. G. (1983). "Unlearning" has a stabilizing effect in collective memories. *Nature*, *304*(5922), 158–159. http://dx.doi.org/10.1038/304158a0.

Kalaj, S., Lauditi, C., Perugini, G., Lucibello, C., Malatesta, E. M., & Negri, M. (2024). Random features Hopfield networks generalize retrieval to previously unseen examples. arXiv preprint arXiv:2407.05658.

Kanter, I., & Sompolinsky, H. (1987). Associative recall of memory without errors. *Physical Review A*, *35*(1), 380.

Kojima, T., Nonaka, H., & Da-Te, T. (1995). Capacity of the associative memory using the Boltzmann machine learning. *vol. 5*, In *Proceedings of iCNN'95-international conference on neural networks* (pp. 2572–2577). IEEE.

Krotov, D., & Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems*, *29*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

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

Louart, C., Liao, Z., & Couillet, R. (2018). A random matrix approach to neural networks. *The Annals of Applied Probability*, *28*(2), 1190–1248.

Löwe, M. (1998). On the storage capacity of hopfield models with correlated patterns. *The Annals of Applied Probability*, *8*(4), 1216–1250.

Lucibello, C., & Mézard, M. (2023). The exponential capacity of dense associative memories. arXiv preprint arXiv:2304.14964.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.

Marinari, E. (2019). Forgetting memories and their attractiveness. *Neural Computation*, *31*(3), 503–516. http://dx.doi.org/10.1162/neco_a_01162.

Mei, S., & Montanari, A. (2022). The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, *75*(4), 667–766.

Negri, M., Lauditi, C., Perugini, G., Lucibello, C., & Malatesta, E. (2023). Storage and learning phase transitions in the random-features hopfield model. *Physical Review Letters*, *131*(25), Article 257301.

Nokura, K. (1996a). Paramagnetic unlearning in neural network models. *Physical Review E*, *54*(5), 5571.

Nokura, K. (1996b). Unlearning in the paramagnetic phase of neural network models. *Journal of Physics A (Mathematical and General)*, *29*(14), 3871.

Nouri, A., & Seyyedsalehi, S. A. (2023). Eigen value based loss function for training attractors in iterated autoencoders. *Neural Networks*.

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

Personnaz, L., Guyon, I., & Dreyfus, G. (1985). Information storage and retrieval in spin-glass like neural networks. *Journal de Physique Lettres*, *46*(8), 359–365.

Plakhov, A. Y. (1994). The converging unlearning algorithm for the Hopfield neural network: optimal strategy. *vol. 2*, In *Proceedings of the 12th IAPR international conference on pattern recognition, vol. 3-conference c: signal processing (cat. no. 94CH3440-5)* (pp. 104–106). IEEE.

Plakhov, A., & Semenov, S. (1992). The modified unlearning procedure for enhancing storage capacity in hopfield network. In *[proceedings] 1992 RNNS/iEEE symposium on neuroinformatics and neurocomputers* (pp. 242–251). IEEE.

Plakhov, A. Y., Semenov, S. A., & Shuvalova, I. B. (1995). Convergent unlearning algorithm for the Hopfield neural network. In *Proceedings 1995 second New zealand international two-stream conference on artificial neural networks and expert systems* (pp. 30–33). IEEE.

Pöppel, G., & Krey, U. (1987). Dynamical learning process for recognition of correlated patterns in symmetric spin glass models. *Europhysics Letters*, *4*(9), 979.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, *20*.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., et al. (2020). Hopfield networks is all you need. arXiv preprint arXiv:2008.02217.

Robins, A., & McCallum, S. (1998). Catastrophic Forgetting and the Pseudorehearsal Solution in Hopfield-type Networks. *Connection Science*, *10*(2), 121, URL: https://www.academia.edu/101394251/Catastrophic_Forgetting_and_the_Pseudorehearsal_Solution_in_Hopfield_type_Networks.

Saglietti, L., Gerace, F., Ingrosso, A., Baldassi, C., & Zecchina, R. (2018). From statistical inference to a differential learning rule for stochastic neural networks. *Interface Focus*, *8*(6), Article 20180033.

Scellier, B., & Bengio, Y. (2017). Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, *11*, http://dx.doi.org/10.3389/fncom.2017.00024, URL: https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2017.00024/full, Publisher: Frontiers.

Storkey, A. (1997). Increasing the capacity of a hopfield network without sacrificing functionality. In W. Gerstner, A. Germond, M. Hasler, & J.-D. Nicoud (Eds.), *Artificial neural networks — ICANN'97* (pp. 451–456). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/BFb0020196.

Storkey, A. J., & Valabregue, R. (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks*, *12*(6), 869–876. http://dx.doi.org/10.1016/S0893-6080(99)00038-6, URL: https://www.sciencedirect.com/science/article/pii/S0893608099000386.

Van Hemmen, J. (1997). Hebbian learning, its correlation catastrophe, and unlearning. *Network: Computation in Neural Systems*, *8*(3), V1.

Van Hemmen, J., Ioffe, L., Kühn, R., & Vaas, M. (1990). Increasing the efficiency of a neural network through unlearning. *Physica A. Statistical Mechanics and its Applications*, *163*(1), 386–392.

Van Hemmen, J., & Klemmer, N. (1992). Unlearning and its relevance to REM sleep: Decorrelating correlated data. In *Neural network dynamics: proceedings of the workshop on complex dynamics in neural networks, June 17–21 1991 at IIASS, vietri, Italy* (pp. 30–43). Springer.

Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.