Accelerating Hierarchical Associative Memory: A Deep Equilibrium Approach

Cédric Goemaere **Johannes Deleu** **Thomas Demeester**
IDLab, Ghent University – imec
Ghent, Belgium
first.last@ugent.be

Abstract

Hierarchical Associative Memory models have recently been proposed as a versatile extension of continuous Hopfield networks. In order to facilitate future research on such models, especially at scale, we focus on increasing their simulation efficiency on digital hardware. In particular, we propose two strategies to speed up memory retrieval in these models, which corresponds to their use at inference, but is equally important during training. First, we show how they can be cast as Deep Equilibrium Models, which allows using faster and more stable solvers. Second, inspired by earlier work, we show that alternating optimization of the even and odd layers accelerates memory retrieval by a factor close to two. Combined, these two techniques allow for a much faster energy minimization, as shown in our proof-of-concept experimental results. The code is available at https://github.com/cgoemaere/hamdeq.

1 Introduction and Related Work

In 1982, the Hopfield network was suggested as a model for associative memory retrieval [1]. It restores corrupted memories by minimizing an internal energy function, which holds the true memories at its minima. In recent years, there has been a renewed interest in Hopfield networks, which has lead to a series of architectural improvements over the original formulation [2, 3, 4, 5, 6]. In this paper, we work with the Hierarchical Associative Memory (HAM) [6], which extends the framework of continuous Hopfield networks [7] to arbitrary network architectures.

Accelerating the energy minimization process of such models is currently an underexplored research direction. However, we consider it an essential step in stimulating future research on Hopfield networks in general, especially at larger scales than currently investigated. One idea, proposed a few years ago, is to train a separate feed-forward model to initialize the state close to the energy minimum [8, 9]. In our paper, we propose and empirically verify two complementary strategies, which do not require augmenting the models with additional weights. First, we make an explicit connection between multi-layer HAMs and Deep Equilibrium Models [10]. Second, we identify and resolve a redundant optimization step that occurs in synchronously updated HAMs. Finally, we show in Section 4 that combining these two techniques maximizes convergence speed in HAMs.

Hopfield networks as Deep Equilibrium Models The research track of Deep Equilibrium Models (DEQs) has unfolded largely independently from the aforementioned evolutions in Hopfield networks. Still, DEQs were introduced as a framework for recurrent neural networks operating on static inputs [10], which essentially holds for Hopfield networks too. The specific formulation of DEQs as implicit fixed point equations allows for the use of advanced solvers, such as Anderson acceleration [11, 12] and Broyden's method [13]. Furthermore, unlike for Hopfield networks, the stability of DEQs is a widely studied area, that includes regularization terms and even parametrizations that are provably stable [10, 14, 15, 16, 17]. The close relationship between DEQs and Hopfield networks has been noticed before [6, 18, 19], and yet, remarkably, none of the many advantages that come with the

Associative Memory & Hopfield Networks in 2023. NeurIPS 2023 workshop.

DEQ framework are exploited in these works. In Section 4, we show the benefits of casting Hopfield networks (specifically, HAMs) as DEQs.

Even-odd splitting in Hopfield networks Bengio et al. (2016) [8] suggested that a Hopfield network may be accelerated through a layerwise energy minimization, conditioned on the values of all other layers. In a sequentially layered network, this enables an update scheme alternating between even and odd layers, fully optimizing one while keeping the other fixed. However, in their definition of a Hopfield network, the optimal value of a single neuron does not just depend on its neighbors, but also on its own value. Solving for this implicit optimal value requires numerical methods, thereby nullifying the original aim of speeding up the model in practice. We find that HAMs, on the other hand, are naturally suited for this procedure, which boosts their convergence speed by a factor close to two. In Appendix A, we explore the implicit optimization problem encountered in [8], and find that it can actually be reduced to an equivalent HAM.

2 A Deep Equilibrium formulation of Hierarchical Associative Memory

We define the energy function of our multi-layered HAM as follows:

$$E(\boldsymbol{s}) = (\boldsymbol{s} - \boldsymbol{b})^T \rho(\boldsymbol{s}) - \mathcal{L}(\boldsymbol{s}) - \frac{1}{2}\rho(\boldsymbol{s})^T \boldsymbol{W} \rho(\boldsymbol{s}) \tag{1}$$

in which $\boldsymbol{s} \in \mathbb{R}^n$ is the n-dimensional state vector, $E : \mathbb{R}^n \to \mathbb{R}$ is the HAM's global energy function, $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is a Lagrangian function such that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}} = \rho(\boldsymbol{s})$, whereby $\rho : \mathbb{R}^n \to \mathbb{R}^n$ is a non-linear activation function[1]. Finally, $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ are weights and biases, respectively (see Appendix B.1 for details on the layered structure of $\boldsymbol{W}$). The input $\boldsymbol{x} \in \mathbb{R}^d$ is applied through the first d states, which are kept equal to $\boldsymbol{x}$ at all times.

The energy E is guaranteed to decrease over time [20] using the following state update rule[2]:

$$\frac{d\boldsymbol{s}}{dt} = -\frac{\partial E}{\partial \boldsymbol{s}} = \rho'(\boldsymbol{s}) \odot (-\boldsymbol{s} + \boldsymbol{W}\rho(\boldsymbol{s}) + \boldsymbol{b}) \tag{2}$$

The equilibrium state $\boldsymbol{s}^*$ can be obtained by numerical integration of Eq. (2). In the literature on Hopfield networks, the forward Euler method is typically used [8, 20, 21, 22]. However, in the field of Neural ODEs [23], it is customary to use more advanced ODE solvers, and these techniques have already been suggested for Hopfield networks as well [6]. Nonetheless, in contrast with Neural ODEs, only the final equilibrium matters in a Hopfield network, not the entire trajectory. In this case, the ODE can be solved much faster by casting it as a DEQ [10, 24], by requiring $\frac{d\boldsymbol{s}^*}{dt} = \boldsymbol{0}$, and hence

$$\boldsymbol{0} = \rho'(\boldsymbol{s}^*) \odot (-\boldsymbol{s}^* + \boldsymbol{W}\rho(\boldsymbol{s}^*) + \boldsymbol{b}) \tag{3}$$

Solving this DEQ with a simple damped Picard iteration is mathematically equivalent to solving the ODE of Eq. (2) with the forward Euler method. However, using more advanced solvers, as is typically done for DEQs (e.g., see [10]), allows for faster convergence[3], as we will show in Section 4.

So far, we have not made the static input explicit. When the equilibrium state $\boldsymbol{s}^*$ is split up into the input $\boldsymbol{x}$ and hidden state $\tilde{\boldsymbol{s}}^*$, i.e., $\boldsymbol{s}^* = [\boldsymbol{x}; \tilde{\boldsymbol{s}}^*]$, Eq. (3) becomes

$$\boldsymbol{0} = \rho'(\tilde{\boldsymbol{s}}^*) \odot (-\tilde{\boldsymbol{s}}^* + \tilde{\boldsymbol{W}}\rho(\tilde{\boldsymbol{s}}^*) + \tilde{\boldsymbol{b}} + \boldsymbol{U}\rho(\boldsymbol{x})) \tag{4}$$

Similar to $\boldsymbol{W}$, the weight matrices $\tilde{\boldsymbol{W}}$ and $\boldsymbol{U}$ also have specific structures (see Appendix B.2).

Notice that there are two distinct solutions for components of $\tilde{\boldsymbol{s}}^*$ in Eq. (4). The first is the trivial solution $\rho'(\tilde{\boldsymbol{s}}^*) = \boldsymbol{0}$, corresponding to state saturation. The second solution corresponds to:

$$\tilde{\boldsymbol{s}}^* = \tilde{\boldsymbol{W}}\rho(\tilde{\boldsymbol{s}}^*) + \tilde{\boldsymbol{b}} + \boldsymbol{U}\rho(\boldsymbol{x}) \tag{5}$$

As the trivial solution sets states to saturation regardless of $\boldsymbol{x}$, this solution is undesirable. Therefore, we will henceforth use Eq. (5) to (implicitly) describe the dynamics of a HAM, instead of Eq. (2). For readability, we will leave out the tilde from the notation, from now on.

[1] Note that the particular type of HAM is determined by the choice of $\mathcal{L}$, and hence ρ. For the experiments in this paper, we assume an additive Lagrangian $\mathcal{L}$, leading to a scalar function ρ (see [6]), applied element-wise to the state vector as $\rho(\boldsymbol{s})$. Although the proposed strategies to speed up inference are not relying on the particular choice of ρ, extending our results to more general models remains a topic of future research.

[2] We use $\odot$ to represent the Hadamard (element-wise) product.

[3] This approach does not come with any guarantees for energy minimization, and may lead us to spurious extrema. In this work, however, we will assume that the advanced solver always returns the true energy minimum.

3 Insights in even-odd splitting for memory retrieval in HAMs

In this section, we provide new insights on the idea of even-odd splitting, particularly in the context of HAMs. First, we argue that even-odd splitting corresponds to parallelizing asynchronous updates (see Insight #1 below). Then, we explain how, for HAMS in particular, a single such update directly yields the locally optimal next state for a given layer (Insight #2). Finally, we show that even-odd splitting in HAMs allows for modeling only the even layers explicitly, or only the odd layers, depending on the parity of the output layer. This corresponds to performing two asynchronous update steps at a similar computational cost as a single synchronous update (Insight #3).

Insight #1 – Even-odd splitting corresponds to parallel asynchronous updates. In practice, Hopfield networks (including HAMs) typically update all states in parallel. This is referred to as synchronous updates, which are more computationally efficient, but may lead to oscillatory state behavior [25, 26]. Asynchronous updates, on the other hand, do guarantee stable state convergence, but are sequential by nature. Here, a single neuron is updated at a time, conditioned on all other neurons. Usually, this neuron is selected at random, but any order is technically allowed [25]. By grouping the neurons from all even/odd layers, we maximally parallelize these individual asynchronous updates, reducing the computational gap with synchronous updates.

Insight #2 – Asynchronous updates in HAMs are locally optimal. In a HAM, as defined by Eq. (5), a neuron interacts only with its direct neighbors, and its optimal value is not self-dependent, as was the case for the Hopfield network of Bengio et al. (2016). This avoids the aforementioned issue of an implicit optimal value, and enables us to quickly calculate the local energy minimum of a neuron, conditioned on its neighbors.

In fact, the optimal value of a neuron can be calculated in a single step. This becomes evident when introducing even-odd splitting in HAMs. Mathematically, this comes down to rearranging the state vector using a permutation matrix $\boldsymbol{P}$, converting $\boldsymbol{s}^* = [\boldsymbol{s}_1^*; \boldsymbol{s}_2^*; \boldsymbol{s}_3^*; \ldots]$ into $[\boldsymbol{s}_{\text{even}}^*; \boldsymbol{s}_{\text{odd}}^*]$, whereby $\boldsymbol{s}_{\text{even}}^* = [\boldsymbol{s}_2^*, \boldsymbol{s}_4^*, \ldots]$ and $\boldsymbol{s}_{\text{odd}}^* = [\boldsymbol{s}_1^*, \boldsymbol{s}_3^*, \ldots]$. Recall that the input layer $\boldsymbol{s}_0^* = \boldsymbol{x}$ has been separated from $\boldsymbol{s}^*$ in Eq. (4), and hence is not part of $\boldsymbol{s}_{\text{even}}^*$, nor of $\boldsymbol{s}_{\text{odd}}^*$.

Applying $\boldsymbol{P}$ to $\boldsymbol{W}$, $\boldsymbol{s}^*$, $\boldsymbol{b}$ and $\boldsymbol{U}$, we find:

$$\boldsymbol{P}\boldsymbol{W}\boldsymbol{P}^{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{W}_{\boldsymbol{P}}^{\boldsymbol{T}} \\ \boldsymbol{W}_{\boldsymbol{P}} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{P}\boldsymbol{s}^* = \begin{bmatrix} \boldsymbol{s}_{\text{even}}^* \\ \boldsymbol{s}_{\text{odd}}^* \end{bmatrix}, \quad \boldsymbol{P}\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_{\text{even}} \\ \boldsymbol{b}_{\text{odd}} \end{bmatrix}, \quad \boldsymbol{P}\boldsymbol{U} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{U}_{\text{odd}} \end{bmatrix}$$

In Appendix B.3), we provide more details on this procedure and on the structure of $\boldsymbol{W}_{\boldsymbol{P}}$, together with an interpretation on the architectural implications of even-odd splitting.

Transforming $\boldsymbol{W}, \boldsymbol{s}^*, \boldsymbol{b}, \boldsymbol{U} \rightarrow \boldsymbol{P}\boldsymbol{W}\boldsymbol{P}^{\boldsymbol{T}}, \boldsymbol{P}\boldsymbol{s}^*, \boldsymbol{P}\boldsymbol{b}, \boldsymbol{P}\boldsymbol{U}$ in Eq. (5), we find the following DEQ:

$$\begin{cases} \boldsymbol{s}_{\text{even}}^* & = \boldsymbol{W}_{\boldsymbol{P}}^{\boldsymbol{T}} \rho(\boldsymbol{s}_{\text{odd}}^*) + \boldsymbol{b}_{\text{even}} \\ \boldsymbol{s}_{\text{odd}}^* & = \boldsymbol{W}_{\boldsymbol{P}} \rho(\boldsymbol{s}_{\text{even}}^*) + \boldsymbol{b}_{\text{odd}} + \boldsymbol{U}_{\text{odd}} \rho(\boldsymbol{x}) \end{cases} \tag{6}$$

From Eq. (6), we can see that, given a fixed value of $\boldsymbol{s}_{\text{odd}}^*$, the optimal value for $\boldsymbol{s}_{\text{even}}^*$ can be found in a single step, and vice versa.

Insight #3 – Even-odd splitting in HAMs allows for omitting part of the states. Let's assume an odd number $2k+1$ of layers, so that the output layer $\boldsymbol{s}_{2k}^*$ belongs to $\boldsymbol{s}_{\text{even}}^*$. Now, $\boldsymbol{s}_{\text{odd}}^*$ consists only of internal layers, which we do not have to model explicitly. Hence, we can simplify the DEQ from Eq. (6) to:

$$\boldsymbol{s}_{\text{even}}^* = \boldsymbol{W}_{\boldsymbol{P}}^{\boldsymbol{T}} \rho\big(\boldsymbol{W}_{\boldsymbol{P}} \rho(\boldsymbol{s}_{\text{even}}^*) + \boldsymbol{b}_{\text{odd}} + \boldsymbol{U}_{\text{odd}} \rho(\boldsymbol{x})\big) + \boldsymbol{b}_{\text{even}} \tag{7}$$

A similar approach allows eliminating $\boldsymbol{s}_{\text{even}}^*$ when the output layer belongs to $\boldsymbol{s}_{\text{odd}}^*$.

Moreover, our formulation reveals an interesting phenomenon hidden within the HAM. Minimizing E with synchronous state updates, using the forward Euler method and a time step equal to 1, is equivalent to solving Eq. (6) using a fixed point iteration. As illustrated in Fig. 1, this scenario corresponds exactly to simultaneously solving two DEQs of the form of Eq. (7), one at time t (solid), the other at $t+1$ (dashed). State convergence is only guaranteed over

two time steps (i.e., a length-2 limit cycle exists), as has long been known for Hopfield networks [25, 26]. Here, however, absolute convergence can also be achieved, but only when both the solid and the dashed DEQ of Fig. 1 converge to the same equilibrium point. We can guarantee this behavior by simply modeling a single DEQ (e.g., the solid one in Fig. 1) and defining the second DEQ as a time-shifted copy of the first one. This is effectively what is happening in Eq. (7). Importantly, by modeling two time steps (i.e., $s^t_{\text{even}} \rightarrow s^{t+2}_{\text{even}}$) in a single iteration, this formulation should converge twice as fast as the HAM from Eqs. (5) and (6), at the same computational cost.

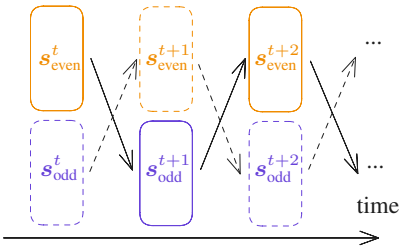

Figure 1: A view of synchronous updates across time reveals two separate even-odd DEQs (solid & dashed)

4 Experimental Results

Experimental setup We test our two strategies in a 3-layer HAM trained on the MNIST dataset [27]. Scellier et al. (2017) [20] advised that layerwise learning rates should be set so that $||\Delta \boldsymbol{W}_i||/||\boldsymbol{W}_i||$ stays constant throughout training. For that reason, we decided to use the Madam optimizer [28], which does this automatically, removing the need for a manual layerwise learning rate sweep. Further details are provided in Appendix C.

Model	#iters till conv.	Test accuracy
HAM	8.2 ($\pm$0.3)	96.9% ($\pm$0.2%)
HAM-DEQ	6.2 ($\pm$0.4)	96.7% ($\pm$0.3%)
HAM-EO	5.1 ($\pm$0.3)	97.2% ($\pm$0.1%)
HAM-EO-DEQ	**4.5** ($\pm$0.3)	96.7% ($\pm$0.3%)

Table 1: Impact of using Anderson acceleration ('DEQ') and even-odd splitting ('EO') on the mean number of iterations till convergence (defined by a relative residual below 10^{-4}) and MNIST test accuracy. Results averaged over 10 runs with mean and standard deviation shown.

Interpretation The results of our experiments are shown in Table 1. We see that both the use of Anderson acceleration, as enabled by the DEQ framework, and the use of even-odd splitting significantly accelerate the energy minimization of a HAM, without harming the test performance. Combining the two techniques maximizes convergence speed. While we theoretically derived that even-odd splitting should converge twice as fast, we see that this is not exactly the case in our experiments. We suspect that the state initialization might play a role here, as initial dynamics differ from the regular regime of the model. For a visual comparison of the state dynamics in the different models, we refer the reader to Fig. 3 in Appendix D.

5 Conclusion

We looked at HAMs through the lens of DEQs, and found a DEQ formulation that functionally corresponds to a HAM, allowing the use of more advanced fixed point solvers to speed up memory retrieval. Furthermore, we showed that HAMs could significantly benefit from even-odd splitting, an idea originally suggested in the context of continuous Hopfield networks. Introducing this technique in HAMs revealed a redundant optimization procedure hidden within the model. By resolving this redundancy, we were able to model two time steps at the computational cost of one. Our results indicate that both advanced DEQ solvers and even-odd splitting provide much faster convergence in HAMs, especially when combined. The presented work provides tools for the practical scaling-up of Hopfield networks, which we hope will stimulate further research into this exciting field.

As mentioned in Section 1, the field of DEQs focuses on stability and faster training, an angle that is often missing from work on Hopfield networks. With this paper, we hope to encourage the use of the DEQ framework in the Hopfield networks community, to benefit from the many advantages that come with it. A vectorized derivative-free notation improves readability, and the use of DEQ solvers and training methods significantly accelerates training. Additionally, DEQ metrics may provide more insight into why a system is or is not working properly. For example, tracking convergence statistics is critical in DEQs, and may explain an unexpectedly poor result from a model that simply did not converge within the given time (e.g., see [29]).

We provide a Limitations section in Appendix E.

Acknowledgements

We are grateful to Felix Koulischer and Tom Van Der Meersch for their thorough proofreading and valuable feedback on this paper.

This research was funded by the Research Foundation - Flanders (FWO-Vlaanderen) under grants G0C2723N and 11PR824N, the Flemish Government (AI Research Program), and the Special Research Fund (BOF) of Ghent University.

References

[1] J J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

[2] Dmitry Krotov and John J Hopfield. "Dense associative memory for pattern recognition". In: *Advances in neural information processing systems*. Vol. 29. 2016.

[3] Mete Demircigil et al. "On a Model of Associative Memory with Huge Storage Capacity". In: *Journal of Statistical Physics* 168 (2017), pp. 288–299. DOI: 10.1007/s10955-017-1806-y.

[4] Dmitry Krotov and John J. Hopfield. "Large Associative Memory Problem in Neurobiology and Machine Learning". In: *International Conference on Learning Representations*. 2021.

[5] Hubert Ramsauer et al. "Hopfield Networks is All You Need". In: *International Conference on Learning Representations*. 2021.

[6] Dmitry Krotov. "Hierarchical Associative Memory". In: *ArXiv* abs/2107.06446 (2021).

[7] John J Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons." In: *Proceedings of the National Academy of Sciences* 81.10 (1984), pp. 3088–3092. DOI: 10.1073/pnas.81.10.3088.

[8] Yoshua Bengio et al. "Feedforward Initialization for Fast Inference of Deep Generative Networks is biologically plausible". In: *ArXiv* abs/1606.01651 (2016).

[9] Peter O'Connor, Efstratios Gavves, and Max Welling. "Initialized Equilibrium Propagation for Backprop-Free Training". In: *International Conference on Learning Representations*. 2019.

[10] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "Deep Equilibrium Models". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[11] Donald G Anderson. "Iterative procedures for nonlinear integral equations". In: *Journal of the ACM (JACM)* 12.4 (1965), pp. 547–560. DOI: 10.1145/321296.321305.

[12] Homer F. Walker and Peng Ni. "Anderson Acceleration for Fixed-Point Iterations". In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735. DOI: 10.1137/10078356X.

[13] Charles G Broyden. "A class of methods for solving nonlinear simultaneous equations". In: *Mathematics of Computation* 19.92 (1965), pp. 577–593. DOI: 10.2307/2003941.

[14] L. Ghaoui et al. "Implicit Deep Learning". In: *SIAM Journal on Mathematics of Data Science* (2019). DOI: 10.1137/20M1358517.

[15] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. "Stabilizing Equilibrium Models by Jacobian Regularization". In: *International Conference on Machine Learning*. 2021.

[16] Max Revay, Ruigang Wang, and Ian Manchester. "Lipschitz-Bounded Equilibrium Networks". In: *Submitted to International Conference on Learning Representations* (2021). URL: https://openreview.net/forum?id=bodgPrarPUJ.

[17] Ezra Winston and J. Z. Kolter. "Monotone operator equilibrium networks". In: *Neural Information Processing Systems* (2020).

[18] Toshihiro Ota and Masato Taki. "iMixer: hierarchical Hopfield network implies an invertible, implicit and iterative MLP-Mixer". In: *ArXiv* abs/2304.13061 (2023).

[19] Axel Laborieux and F T Zenke. "Improving equilibrium propagation without weight symmetry through Jacobian homeostasis". In: *ArXiv* abs/2309.02214 (2023).

[20] Benjamin Scellier and Yoshua Bengio. "Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation". In: *Frontiers in Computational Neuroscience* 11 (2017). DOI: 10.3389/fncom.2017.00024.

[21] Yoshua Bengio. "Early Inference in Energy-Based Models Approximates Back-Propagation". In: *ArXiv* abs/1510.02777 (2015).

[22] Jimmy Gammell et al. “Layer-Skipping Connections Improve the Effectiveness of Equilibrium Propagation on Layered Networks”. In: *Frontiers in Computational Neuroscience* 15 (2021), p. 627357. DOI: 10.3389/fncom.2021.627357.

[23] Ricky TQ Chen et al. “Neural Ordinary Differential Equations”. In: *Neural Information Processing Systems*. Vol. 31. 2018.

[24] Avik Pal, Alan Edelman, and Christopher Rackauckas. “Continuous Deep Equilibrium Models: Training Neural ODEs faster by integrating them to Infinity”. In: *Submitted to Transactions on Machine Learning Research* (2022). URL: https://openreview.net/forum?id=mlI9f7u6Zo.

[25] Pascal Koiran. “Dynamics of Discrete Time, Continuous State Hopfield Networks”. In: *Neural Computation* 6.3 (May 1994), pp. 459–468. DOI: 10.1162/neco.1994.6.3.459.

[26] Lipo Wang. “On the dynamics of discrete-time, continuous-state Hopfield neural networks”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 45.6 (1998), pp. 747–749. DOI: 10.1109/82.686695.

[27] Gregory Cohen et al. “EMNIST: Extending MNIST to handwritten letters”. In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

[28] Jeremy Bernstein et al. “Learning compositional functions via multiplicative weight updates”. In: *Neural Information Processing Systems*. 2020.

[29] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. “Multiscale deep equilibrium models”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5238–5250.

[30] Yann LeCun. “The MNIST database of handwritten digits”. In: *http://yann. lecun. com/exdb/mnist/* (1998).

[31] Patrick J Grother. “NIST special database 19”. In: *Handprinted forms and characters database, National Institute of Standards and Technology* 10 (1995), p. 69. DOI: 10.18434/T4H01C.

[32] Axel Laborieux et al. “Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing its Gradient Estimator Bias”. In: *Frontiers in Neuroscience* (2020). DOI: 10.3389/fnins.2021.633674.

[33] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[34] Luis B. Almeida. “A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment”. In: *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA). Vol. II. Piscataway, NJ: IEEE, 1987, pp. 609–618. DOI: 10.5555/104134.104145.

[35] Fernando J Pineda. “Generalization of back-propagation to recurrent neural networks”. In: *Physical review letters* 59.19 (1987), p. 2229. DOI: 10.1103/PhysRevLett.59.2229.

Appendix

A Asynchronous local optimization in the continuous Hopfield network of Bengio et al. (2016) is equivalent to a HAM

In this appendix, we delve deeper into the implicit optimization problem that arises when trying to find the optimal value for a neuron in the continuous Hopfield network formulated in [8]. We start by casting the network to a DEQ form, analogous to Section 2. The procedure is exactly the same as for HAMs, hence, we only provide the most important equations. Using this DEQ form, we find that the implicit optimization can be solved analytically, by essentially casting the Hopfield network as a HAM.

Continuous Hopfield network as DEQ
In line with prior work [21], Bengio et al. (2016) define the energy function of the Hopfield network as follows:

$$E(\boldsymbol{s}) = \frac{1}{2}||\boldsymbol{s}||^2 - \frac{1}{2}\rho(\boldsymbol{s})^T \boldsymbol{W}\rho(\boldsymbol{s}) - \boldsymbol{b}^T\rho(\boldsymbol{s}) \tag{8}$$

For clarity, we use the same notation as in Eq. (1).

The state update rule becomes:

$$\frac{d\boldsymbol{s}}{dt} = -\frac{\partial E}{\partial \boldsymbol{s}} = -\boldsymbol{s} + \rho'(\boldsymbol{s}) \odot (\boldsymbol{W}\rho(\boldsymbol{s}) + \boldsymbol{b})$$

Or in our DEQ form (with implicit input dependence):

$$\boldsymbol{s}^* = \rho'(\boldsymbol{s}^*) \odot (\boldsymbol{W}\rho(\boldsymbol{s}^*) + \boldsymbol{b})$$

Locally optimal asynchronous updates in continuous Hopfield networks
The optimal value of a single neuron, conditioned on all other neurons, is given by

$$s_i^* = \rho'(s_i^*) \cdot C_i$$

where C_i is a constant representing the combined influence of the neighboring neurons of s_i.

Instead of solving this implicit optimization problem with numerical methods, we can also solve it analytically. We define

$$f(x) = x/\rho'(x)$$

If f is invertible, we can compute s_i^* as

$$s_i^* = f^{-1}(C_i)$$

From continuous Hopfield network to HAM
We can relax the condition of full bijective invertibility by working directly on the DEQ instead of on the neuron level. A multivariate version of f can easily be defined using elementwise division.

By introducing $\boldsymbol{s}_f^* = f(\boldsymbol{s}^*)$ and assuming invertibility of f, we can rearrange the DEQ to:

$$\boldsymbol{s}_f^* = \boldsymbol{W}\rho(f^{-1}(\boldsymbol{s}_f^*)) + \boldsymbol{b}$$

Comparing this equation with Eq. (5), we see that this is exactly a HAM with non-linearity $\rho \circ f^{-1}$. Instead of requiring the bijective invertibility of f, we only need $\rho \circ f^{-1}$ (or an analytical continuation thereof) to be a bijection. Hence, f is allowed to be non-injective, as long as all inputs belonging to a certain output value are also mapped to a single value under ρ.

Essentially, this means that every layered continuous Hopfield network with an energy function of the form of Eq. (8) can be converted into an equivalent HAM, as long as ρ is chosen properly. For full equivalence, one also needs to properly preprocess x, by replacing it with $f(x)$, such that, under the mapping of $\rho \circ f^{-1}$, we still get the original value of $\rho(x)$ that we would find in the Hopfield network.

B Structure of different weight matrices in layered Hopfield networks

We can represent the different layerwise weight matrices $\boldsymbol{W_i}$ in one single large weight matrix $\boldsymbol{W}$. The structure of $\boldsymbol{W}$ is the same for both the Hopfield network from Appendix A and the HAM from the main body. To stay consistent with the primary focus of the paper, we work with HAMs in this appendix, although the results apply to any layered Hopfield network.

B.1 Weight matrix in a HAM

For clarity, we restate the DEQ-form of the multi-layer HAM of Eq. (3):

$$\mathbf{0} = \rho'(\boldsymbol{s}^*) \odot (-\boldsymbol{s}^* + \boldsymbol{W}\rho(\boldsymbol{s}^*) + \boldsymbol{b})$$

To ensure stability, we exclude self-interaction by enforcing a zero block diagonal on $\boldsymbol{W}$. Additionally, the form of the energy function in Eq. (1) only utilizes the symmetric part of $\boldsymbol{W}$. Hence, for clarity, $\boldsymbol{W}$ is typically chosen to be symmetric by definition. A HAM consisting of multiple layers gives rise to a block tridiagonal $\boldsymbol{W}$. For example, a 5-layer HAM would have the following weight matrix:

$$\boldsymbol{W} = \begin{bmatrix} \mathbf{0} & \boldsymbol{W_0^T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{W_0} & \mathbf{0} & \boldsymbol{W_1^T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{W_1} & \mathbf{0} & \boldsymbol{W_2^T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{W_2} & \mathbf{0} & \boldsymbol{W_3^T} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{W_3} & \mathbf{0} \end{bmatrix} \tag{9}$$

For a nice visualization of $\boldsymbol{W}$, see Figure (5, left) of [22].

B.2 Weight matrix in a HAM with explicit input dependence

For clarity, we restate the multi-layer HAM with explicit input dependence of Eq. (4):

$$\mathbf{0} = \rho'(\tilde{\boldsymbol{s}}^*) \odot (-\tilde{\boldsymbol{s}}^* + \tilde{\boldsymbol{W}}\rho(\tilde{\boldsymbol{s}}^*) + \tilde{\boldsymbol{b}} + \boldsymbol{U}\rho(\boldsymbol{x}))$$

To get the structure of $\tilde{\boldsymbol{W}}$ and $\boldsymbol{U}$, we must simply look at the structure of $\boldsymbol{W}$ in Eq. (9). We drop the first row, as this represents the influence that other states have on $\boldsymbol{x}$. The first column represents the influence that $\boldsymbol{x}$ has on other states, i.e., $\boldsymbol{U}$. The rest constitutes $\tilde{\boldsymbol{W}}$. In other words, in a 5-layer HAM, $\tilde{\boldsymbol{W}}$ and $\boldsymbol{U}$ have the following structure:

$$\tilde{\boldsymbol{W}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{W_1^T} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{W_1} & \mathbf{0} & \boldsymbol{W_2^T} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{W_2} & \mathbf{0} & \boldsymbol{W_3^T} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{W_3} & \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{W_0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

One may be tempted to also include the bias term corresponding to $\boldsymbol{x}$ in Eq. (3). However, looking at the original formulation in Eq. (2), we see that this term only influences the first d states of $\boldsymbol{s}$, which are clamped to $\boldsymbol{x}$ at every time step. In essence, the bias term corresponding to $\boldsymbol{x}$ has no influence on any part of $\boldsymbol{s}$, and is therefore also left out of Eq. (5), leaving only $\tilde{\boldsymbol{b}}$.

B.3 Permuted weight matrix in even-odd split HAMs

Even-odd splitting of the layers in a HAM is equivalent to applying a permutation matrix $\boldsymbol{P}$ to $\boldsymbol{W}$, $\boldsymbol{s}^*$, $\boldsymbol{b}$ and $\boldsymbol{U}$. For example, in a 5-layer HAM, we get:

$$\boldsymbol{PWP}^T = \left[\begin{array}{cc|cc} \mathbf{0} & \mathbf{0} & \boldsymbol{W_1} & \boldsymbol{W_2^T} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{W_3} \\ \hline \boldsymbol{W_1^T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{W_2} & \boldsymbol{W_3^T} & \mathbf{0} & \mathbf{0} \end{array}\right], \quad \boldsymbol{Ps}^* = \left[\begin{array}{c} \boldsymbol{s}_2^* \\ \boldsymbol{s}_4^* \\ \hline \boldsymbol{s}_1^* \\ \boldsymbol{s}_3^* \end{array}\right], \quad \boldsymbol{Pb} = \left[\begin{array}{c} \boldsymbol{b}_2 \\ \boldsymbol{b}_4 \\ \hline \boldsymbol{b}_1 \\ \boldsymbol{b}_3 \end{array}\right], \quad \boldsymbol{PU} = \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \hline \boldsymbol{W_0} \\ \mathbf{0} \end{array}\right]$$

Note that $\boldsymbol{U}_{\text{even}} = \mathbf{0}$, since $\boldsymbol{x}$ is clamped onto $\boldsymbol{s}_0^*$ and hence is part of the even layers, which do not interact with one another. Because of the explicit input dependence notation, $\boldsymbol{s}_{\text{even}}^*$ starts at $\boldsymbol{s}_2^*$.

Mapping $\boldsymbol{PWP}^T$ to the structure of Eq. (9), we can see that the permutation effectively allows us to express a multi-layer HAM as if it had only a single hidden layer, as illustrated in Fig. 2. The bottom left quadrant can be considered a single weight matrix, and this is exactly $\boldsymbol{W_P}$ from Eqs. (6) and (7).

Instead of the structure of Eq. (9), $\boldsymbol{W_P}$ now takes a staircase-like structure, varying between regular and transposed submatrices. When adding another layer, the extra term $\boldsymbol{W_4}$ would be situated below $\boldsymbol{W_3^T}$.

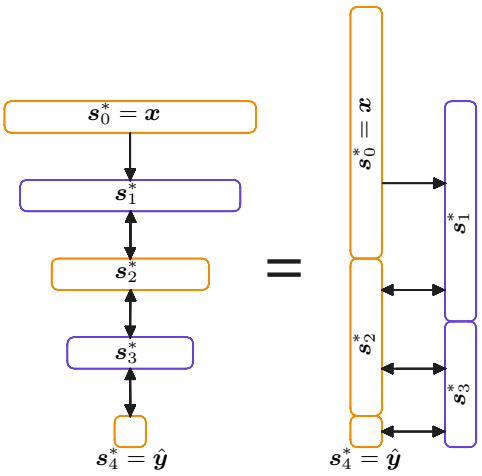

Figure 2: A layered HAM can be split into even and odd layers, and modelled as a non-fully-connected 2-layer network

C Experimental setup

Below is a list of all information required to reproduce the results outlined in Section 4. Moreover, the code is available at `https://github.com/cgoemaere/hamdeq`.

Data

- Dataset: EMNIST-MNIST [27]. This is a drop-in replacement for the MNIST dataset [30], but with a known conversion process from the original NIST digits [31].
- Input preprocessing: rescaling pixel intensities from [0, 255] to [0, 1]
- Batch size: 64
- Epochs: 10
- No data augmentation

Model

- Neurons per layer: [784, 512, 10]
- Non-linearity ρ: sigmoid$(4x - 2)$ (shifted sigmoid; same as [32])
- State initialization: zero initialization, i.e., $\boldsymbol{s}^{t=0} = \mathbf{0}$
- Weight initialization: Xavier initialization [33] per layer (not on large $\boldsymbol{W}$), as we want bidirectional operation between layers. The biases were initialized at zero.
- Forward iterations: 40 (chosen high enough to ensure convergence at all times during training)
- No damping, i.e., if the DEQ is $\boldsymbol{s}^* = f(\boldsymbol{s}^*)$, then we use $\boldsymbol{s}^{t+1} = f(\boldsymbol{s}^t)$ as update rule.

Training

- Loss function: Mean Square Error
- Backward method: Recurrent Backpropagation [34, 35]
- Backward iterations: 8
- Optimizer
  - Type: Madam [28] (chosen as a substitute for layerwise learning rates; Madam automatically scales weight updates according to $||\Delta W||/||W||$, as advised by [20])
  - Learning rate: 0.01 (not tuned)
- No gradient clipping, dropout or other commonly used training techniques
- GPU: 1x GTX-1080Ti

D Visual comparison of state dynamics in different HAM models

Below, we provide a visual comparison of the state dynamics in the different HAM models from Section 4. We see that using Anderson acceleration (as indicated by 'DEQ') helps guarantee convergence in samples that would otherwise not have converged. Additionally, even-odd splitting (as indicated by 'EO') seems to boost convergence speed by a factor close to two, as expected. We can see that the initial dynamics of the models differ from their regular regime, as the trajectories of all samples start out similarly, and only diverge after a few iterations. As for the low density region in the models using Anderson acceleration (most noticeable in the bottom right plot), this is likely caused by the advanced solver finding the exact fixed point solution, bringing the relative residual to zero.

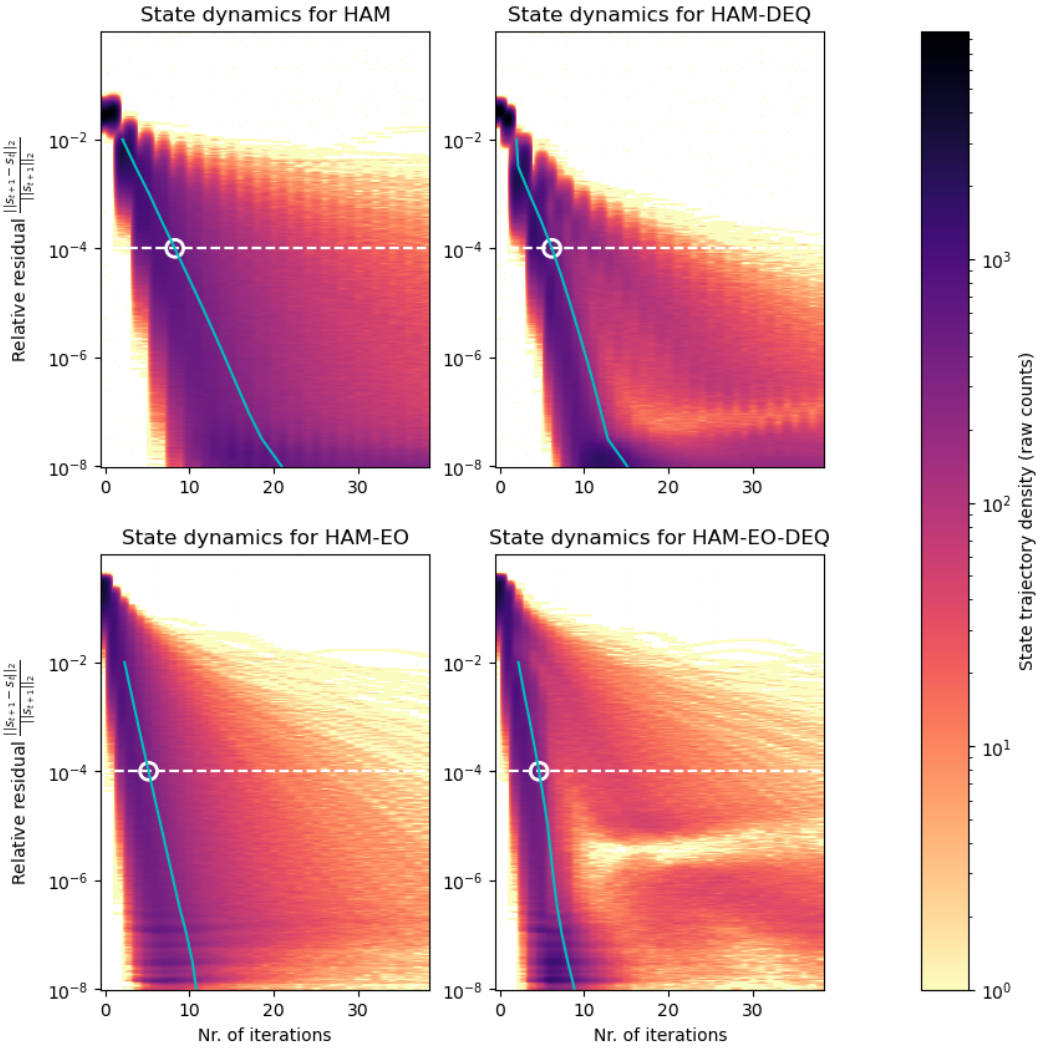

Figure 3: Density heatmap of the state trajectories for different HAM models. The horizontal axis represents the number of iterations of the DEQ. The vertical axis represents the relative residual, which is used to determine the state convergence (the lower, the more converged). The limit of 10^{-4} as chosen criterion for convergence is indicated with a white dashed line. For every model, we show the cumulative results of 10 different seeds, run on the entire MNIST test set. In cyan, we show the mean number of iterations corresponding to a given convergence criterion. The circular marker at the limit of 10^{-4} corresponds to the value reported in Table 1.

E Limitations

We only performed a limited hyperparameter sweep to ensure the stability of our models. The impact of designer choices (e.g., in state/weight initialization, choice of non-linearity, choice of optimizer) is not yet fully understood for Hopfield networks, and we believe there is much room for improvement in these areas. An important parameter is the choice of Lagrangian that determines the considered family of HAM models. In particular, we have not yet applied our results to the HAM extension of the Modern Hopfield Network [4, 5] (corresponding to using the SoftMax function as $\rho(\boldsymbol{s})$), which we plan to work on in the near future.

As a work in progress, our experiments are currently limited to shallow models. In fact, the 3-layer HAM from our experiments actually corresponds to a regular continuous Hopfield network. We expect greater gains from the proposed techniques on deeper models. Preliminary results indicate that the relative difference in convergence speed is maintained as expected, however, we encountered some stability issues in training these deeper models, and could therefore not provide any conclusive results in this paper. Solving these stability issues is left for future work.

In theory, the two proposed techniques should not alter the equilibrium state of a HAM, given its parameters. However, checking whether this holds at all times during training is left for future work as well.