Dyadic Learning in Recurrent and Feedforward Models

Rasmus Kjær Høier * **Kirill Kalinin**† **Maxence Ernoult**‡ **Christopher Zach**§

Abstract

From electrical to biological circuits, feedback plays a critical role in amplifying, dampening and stabilizing signals. In local activity difference based alternatives to backpropagation, feedback connections are used to propagate learning signals in deep neural networks. We propose a saddle-point based framework using dyadic (two-state) neurons for training a family of parameterized models, which include the symmetric Hopfield model, pure feedforward networks and a less explored *skew-symmetric Hopfield variant*. The resulting learning method reduces to equilibrium propagation (EP) for symmetric Hopfield models and to dual propagation (DP) for feedforward networks, while the skew-symmetric Hopfield setting yields a new method with desirable robustness properties. Experimentally we demonstrate that the new skew-symmetric Hopfield model performs on par with EP and DP in terms of the resulting model predictive performance, while exhibiting enhanced robustness to input changes and strong feedback and is less inclined to neural saturation. We identify the fundamentally different types of feedback signals propagated in each model as the main cause of differences in robustness and saturation.

1 Introduction

It has been proposed that the brain may perform some instance of approximate backpropagation, using neural activity differences to represent error signals [1, 2]. Activity difference based approaches come in two main flavours: (i) using *temporal* differences between activities at different times to represent error signals [3, 4, 5], or (ii) using *spatial* differences by subtracting activities of distinct neurons or neuronal compartments [6, 7, 8]. Temporal activity difference methods require at least two relaxations to equilibrium, or infinitely many through slow adiabatic oscillations of the *same* circuit [9], yet at the expense of latency. Conversely, spatial activity difference methods require neurons to multiplex by propagating activities and finite difference error signals simultaneously and some degree of weight sharing, which may be problematic from a biological perspective.

In this manuscript, we take a closer look at the temporal activity difference based algorithm equilibrium propagation (EP) [4] for symmetric Hopfield models (HM), the spatial activity difference based algorithm dual propagation (DP) for feedforward models (FFM) [8], and the fundamental role feedback plays in each of the algorithms. In the context of this work, feedback refers to both the error signal propagated in a supervised learning setting and to top-down connections in a layered network architecture (and therefore introducing cycles in the underlying computational graph). In DP, the feedback signals reduce to top-down (finite difference) errors due to the lack of recurrent connections in feedforward models. While this is an asset for inference speed in practice, recurrent connections are a key feature underpinning attention and robustness in biological systems. On the

*Chalmers University of Technology. Work begun during internship at Microsoft Research.
†Microsoft Research
‡RAIN AI
§Chalmers University of Technology

Second Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2024(MLNCP 2024).

other hand, in EP (and Hopfield nets in general) *positive* feedback is always present: the sign of the signal send from neuron A to neuron B always matches the sign of the signal sent from B to A. This is a consequence of the symmetry of the underlying Hopfield model ($W_{ij} = W_{ji}$). Positive feedback connections in symmetric Hopfield nets yield many desirable properties, but may also lead to excessive amplification of neural activities. This is not too surprising as high sensitivity to input changes and excessive saturation are known features of positive feedback [10].

As FFMs without an explicit error signal lack any kind of feedback from downstream layers and HMs suffer from the aforementioned limitations of persisting positive feedback, it is natural to ask what a *negative* feedback model would look and behave like? As a symmetric Hopfield model results in positive feedback, a *skew-symmetric Hopfield model* (SSHM), with with $W_{ij} = -W_{ji}$, would instead provide negative feedback. Negative feedback loops are typically used to decrease sensitivity to input perturbations and increase the linearity of a system [10] which could respectively endow the resulting model with a degree of biological plausibility and alleviate vanishing gradient issues during training.

We establish a family of models and their associated learning dynamics, which recover as special cases symmetric Hopfield models (HM), feedforward models (FFM) and skew-symmetric Hopfield models (SSHM). In the HM setting, the neural state and synaptic weight dynamics correspond to EP, and the FFM setting results in DP. The SSHM scenario yields a two-phased algorithm which exhibits reduced sensitivity to input perturbations and also to decreased neural saturation.

While the role of feedback in the brain is much more diverse and subject to constraints we are not modelling (such as weight transport and, Dale's principle) we hope this study helps bring attention to the importance of negative feedback in the context of learning. The importance of negative feedback is well known in both neuroscience and control theory, yet in the context of biologically inspired learning algorithms negative feedback has received little to no attention.

2 Related work

Contrastive learning algorithms. In energy-based models, inference amounts to energy minimization. Training can in this setting be achieved by contrasting the states inferred when applying different perturbations to the output units. In variations of contrastive Hebbian learning (CHL) [3, 11, 5], this is done by clamping output units. In EP, the output units are instead weakly nudged by a teaching signal derived from a loss function. EP and CHL typically assume a symmetric connectivity, which for instance naturally map to resistive networks where top-down currents inherently flow through the weights transpose [12, 13, 14, 15].

Credit assignment with dual compartments. DP introduces "dyadic" neurons, which permits representing errors and activities simultaneously as the difference and mean of each neurons compartments, which in terms permits turning the two distinct inference phases of CHL or EP into a single phase. The underlying motivation is conceptually similar, albeit different in terms of resulting algorithms, to the neuroscience inspired approximations of backpropagation [16, 7] which achieve single-phased learning by modelling pyramidal neurons with distinct compartments for integrating top-down and bottom-up signals.

Lifted neural networks. Lifted neural networks have roots in the community of mathematical optimization and frame neural network training as bilevel optimization over weights and activations [17, 18, 19, 20, 21, 22]. Although predating the term *lifted network* [23] also fits in this category. As in variants of CHL, neurons in lifted neural networks rely only on local information and require less synchronization than backpropagation. In the context of lifted networks robustness to input perturbations has previously been explored in [24].

3 A Saddle point objective for dyadic learning

We let subscript k denote row index and subscript $\backslash k$ denote the removal of the $k'\mathrm{th}$ element from a vector. Starting from the optimization problem

$$\min_{W} \ell(s, y) \quad \text{s.t.} \; \forall \; \mathrm{k} \; s_k = \arg\min_{s'_k} E_k(s'_k, s_{\backslash k}, W_k, \theta_0) \tag{1}$$

we derive the following relaxation (see details in appendix A).

$$\min_{W} \min_{s^+} \max_{s^-} \tfrac{1}{2}\ell(s^+, y) + \tfrac{1}{2}\ell(s^-, y) + \tfrac{1}{\beta}\sum_k E_k(s_k^+, \bar{s}_{\setminus k}) - E_k(s_k^-, \bar{s}_{\setminus k}) \tag{2}$$

In the case of a Hopfield-like E_k this correponds to performing gradient descent on the following objective with respect to the weights.

$$\begin{aligned}\mathcal{L}(\theta_0, W) = &\min_{s^+} \max_{s^+} \tfrac{\beta}{2}\ell(s^+, y) + \tfrac{\beta}{2}\ell(s^-, y) \\ &+ G(s^+) - G(s^-) - (s^+ - s^-)^\top \theta_0 x - \tfrac{1}{4}\begin{pmatrix} s^+ \\ s^- \end{pmatrix}^\top \underbrace{\begin{pmatrix} W+W^\top & W-W^\top \\ -W+W^\top & -W-W^\top \end{pmatrix}}_{:=\widetilde{W}} \begin{pmatrix} s^+ \\ s^- \end{pmatrix}\end{aligned} \tag{3}$$

Here θ_0 is an projection mapping datapoints x to the space of neural activities s, while W governs interactions between elements of s. The structure of W has profound impact on how this optimization can be physically realized. The general case of an entirely unstructured weight matrix W requires neurons to perform substantial multiplexing, making it challenging for physical implementations. Choosing instead to parametrize $W_\lambda(\theta) := \theta + (2\lambda - 1)\theta^\top$, where θ is lower triangular, allows smoothly interpolating from symmetric Hopfield models, through feedforward models, to skew-symmetric Hopfield models (each of which offer certain benefits for physical implementations). Employing this parametrization yields the following objective.

$$\begin{aligned}\mathcal{L}_\lambda(\theta_0, \theta) = &\min_{s^+} \max_{s^+} \tfrac{\beta}{2}\ell(s^+, y) + \tfrac{\beta}{2}\ell(s^-, y) \\ &+ G(s^+) - G(s^-) - (s^+ - s^-)^\top \theta_0 x - \tfrac{1}{4}\begin{pmatrix} s^+ \\ s^- \end{pmatrix}^\top \underbrace{\begin{pmatrix} \lambda(\theta+\theta^\top) & \bar{\lambda}(\theta-\theta^\top) \\ \bar{\lambda}(-\theta+\theta^\top) & \lambda(-\theta-\theta^\top) \end{pmatrix}}_{:=\widetilde{W}_\lambda} \begin{pmatrix} s^+ \\ s^- \end{pmatrix}\end{aligned} \tag{4}$$

In the following we will restrict our focus to the more hardware friendly settings offered by $\mathcal{L}_\lambda$.

Dynamics. Training amounts to solving $\min_\theta \mathcal{L}_\lambda(\theta)$. Let the accent $*$ denote optimality, then the gradient of $\mathcal{L}_\lambda(\theta)$ with respect to θ is

$$\tfrac{1}{\beta}\nabla_\theta \mathcal{L}_\lambda(\theta) = -\tfrac{\lambda}{2\beta}(\overset{*}{s}{}^+\overset{*}{s}{}^{+\top} - \overset{*}{s}{}^-\overset{*}{s}{}^{-\top}) + \tfrac{\bar{\lambda}}{2\beta}(\overset{*}{s}{}^-\overset{*}{s}{}^{+\top} - \overset{*}{s}{}^+\overset{*}{s}{}^{-\top}) \tag{5}$$

As $\widetilde{W}_\lambda$ is already symmetric, the gradient with respect to $s^\pm$ is simple to compute

$$\begin{pmatrix} \nabla s^+ \mathcal{L}_\lambda \\ \nabla s^- \mathcal{L}_\lambda \end{pmatrix} = \begin{pmatrix} \nabla_{s^+}\ell(s^+) \\ \nabla_{s^-}\ell(s^-) \end{pmatrix} + \frac{1}{\beta}\left(\begin{pmatrix} f^{-1}(s^+) \\ -f^{-1}(s^-) \end{pmatrix} - \widetilde{W}_\lambda \begin{pmatrix} s^+ \\ s^- \end{pmatrix} - \theta_0 \begin{pmatrix} x \\ -x \end{pmatrix}\right). \tag{6}$$

and the stationary conditions for $\overset{*}{s}{}^\pm$ are

$$\overset{*}{s}{}^\pm = f\big(\theta_0 x + \tfrac{1}{2}W_\lambda(\overset{*}{s}{}^+ + \overset{*}{s}{}^-) \pm \tfrac{1}{2}W_\lambda^\top(\overset{*}{s}{}^+ - \overset{*}{s}{}^-) \pm \beta\nabla\ell(\overset{*}{s}{}^\pm)\big). \tag{7}$$

We determine $\overset{*}{s}{}^\pm$ using dampened fixed-point iterations (see appendix B for details) of the preactivations $a^\pm$ (defined through the relation $s^\pm = f(a^\pm)$) of the form

$$a^\pm \leftarrow (1-\eta)a^\pm + \eta\left(\theta(\lambda s^\pm + \bar{\lambda}s^\mp) + \theta^\top(\lambda s^\pm - \bar{\lambda}s^\mp) + \theta_0 x \mp \beta\nabla\ell(s^\pm)\right). \tag{8}$$

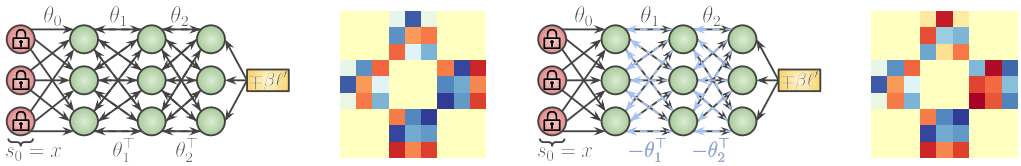
In the remaining part of the paper we will further restrict θ to be strictly lower block triangular (rather than just strictly lower triangular). We write this in terms of layerwise weight matrices θ_k.

$$\theta = \begin{pmatrix} 0 & 0 & \dots & 0 \\ \theta_1 & 0 & \dots & 0 \\ 0 & \theta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \end{pmatrix} \tag{9}$$

This approach is commonly applied to obtain layered Hopfield networks. In this setting we get mirror descent dynamics for the hidden layers of the form

$$a_k^\pm \leftarrow (1-\eta)a_k^\pm + \eta\left(\theta_{k-1}(\lambda s_{k-1}^\pm + \bar{\lambda}s_{k-1}^\mp) + \theta_k^\top(\lambda s_{k+1}^\pm - \bar{\lambda}s_{k+1}^\mp)\right), \tag{10}$$

where the subscript k now denotes layer index. When $\eta = 1$ this reduces to layer-wise closed form updates akin to those employed in EP [25].



(a) (b) (c) (d)

Figure 1: Sketch **1a** and adjacency matrix **1b** of a symmetric Hopfield model. Sketch **1c** and adjacency matrix **1d** of a skew-symmetric Hopfield model. Dashed blue arrows denotes opposite sign of the corresponding forward connection. Note that the input projection θ_0 is not part of the adjacency matrix W_λ.

Recovering symmetric Hopfield models trained with EP. In the case $\lambda = 1$ and (i.e. $\bar{\lambda} = 0$) then the saddle-point problem equation 4 decouples into two separate minimization problems over s^+ and s^-, respectively. This corresponds to the two phases of EP in a deep Hopfield network, yielding the hidden layer dynamics:

$$a_k^\pm \leftarrow (1-\eta)a_k^\pm + \eta\left(\theta_{k-1}s_{k-1}^\pm + \theta_k^\top s_{k+1}^\pm\right) \tag{11}$$

In a hardware realization, this means the two sub-problems can be either solved in parallel (which requires maintaining two identical sets of weights) or sequentially using the same hardware (which avoids weight doubling, but increases runtime). Figure 1a and 1b illustrates the connectivity matrix $\widetilde{W}_{\lambda=1}$ for a small network.

Recovering feedforward models trained with DP. The choice $\lambda = \bar{\lambda} = 1/2$ recovers the DP dynamics.

$$a_k^\pm \leftarrow (1-\eta)a_k^\pm + \eta\left(\tfrac{1}{2}\theta_{k-1}(s_{k-1}^\pm + s_{k-1}^\mp) + \tfrac{1}{2}\theta_k^\top(s_{k+1}^\pm - \bar{s}_{k+1}^\mp)\right) \tag{12}$$

In the absence of a teaching signal ($\beta \to 0$) then $s_k^+ = s_k^-$ for all k, hence the feedback term $\frac{1}{2}\theta_k^\top(s_{k+1}^\pm - \bar{s}_{k+1}^\mp)$ vanishes and the model behaves exactly like a feedforward model. For hardware implementations it is noteworthy that the two compartments $s_{k,i}^+$ and $s_{k,i}^-$ of a neuron i in layer k receive the same bottom up and the same absolute feedback signal (but opposite sign), which allows them to share weights in physical realizations.

Skew-symmetric Hopfield models Fixing $\lambda = 0$ (i.e. $\bar{\lambda} = 1$) implies that $W_{\lambda=0}$ is skew-symmetric (see Fig. 1(c–d)), and only the blocks governing interactions between s^+ and s^- remain in $\widetilde{W}_\lambda$. The dynamics in the layered setting we consider here are

$$a_k^\pm \leftarrow (1-\eta)a_k^\pm + \eta\left(\theta_{k-1}s_{k-1}^\mp - \theta_k^\top s_{k+1}^\mp\right) \tag{13}$$

A critical observation here is that states with odd layer indices of $a^\pm$ only interact directly with $a^\mp$ states with even indices, and vice versa. This means that the global minmax problem over a^+ and a^- decouples into two independent minmax problems: one minmax problem over odd layers of a^+ and even layers of a^-, and another over even layers of a^+ and odd layers of a^-. Consequently, as in the case of the symmetric Hopfield model this setting has two phases, but unlike the symmetric setting each phase here corresponds to a minmax rather than a pure minimization problem.

In early research on *binary* skew-symmetric Hopfield networks it was shown that skew-symmetric weights lead to cycles in binary Hopfield models [26, 27]. We do observe oscillating behaviour in our continuous SSHM when applying analogous closed form updates, but not when employing a small inference stepsize (we used $\eta = 0.05$) in the dampened fixed point iterations. A similar skew-symmetric connectivity has previously been explored in [28] to preserve long term dependencies and avoid vanishing gradients in the context of sequence data (this is different from our setting where input data is static).

Lipschitz properties The skew-symmetric Hopfield model gives rise to remarkably different Lipschitz properties. Since a fixed point satisfies $s = f(W_\lambda s + x)$, we have via implicit differentiation

$$\frac{ds}{dx} = f'(W_\lambda s + x)\left(W_\lambda \frac{ds}{dx} + \mathtt{I}\right). \tag{14}$$

Table 1: Test accuracy and Lipschitz constants (of hidden layers and the full network).

	MNIST			FashionMNIST		
	Test acc (%)	L_{hidden}	L_{output}	Test acc (%)	L_{hidden}	L_{output}
SSHM	98.46 ± 0.04	$1.0\pm$ 7e-7	0.41 ± 0.05	89.45 ± 0.07	$1.0\pm$ 3e-7	0.53 ± 0.03
FFM	98.37 ± 0.09	116.4 ± 12.9	95.75 ± 7.44	89.11 ± 0.17	141.6 ± 43.6	138.17 ± 43.12
HM	97.97 ± 0.12	931.0 ± 492.0	24.07 ± 10.41	89.16 ± 0.25	1257.1 ± 576.7	31.83 ± 16.88

By *assuming linear activation mappings* for the moment, we obtain $\frac{ds}{dx} = (\mathtt{I} - W_\lambda)^{-1}$. In the skew-symmetric setting ($\lambda = 0$), W_0 has a decomposition of the form $W_0 = Q\Sigma Q^\top$ where Q is orthogonal and Σ is a matrix with only zeros and 2×2 blocks of the form $\left(\begin{smallmatrix} 0 & \lambda_i \\ -\lambda_i & 0 \end{smallmatrix}\right)$ on its diagonal. Consequently,

$$W_0^\top W_0 = Q\Sigma^\top \Sigma Q^\top = Q \operatorname{diag}(\lambda_i^2)_i Q^\top, \tag{15}$$

and therefore $W_0^\top W_0$ has only non-negative eigenvalues. Hence, $(\mathtt{I}+W_0)^\top(\mathtt{I}+W_0) = \mathtt{I}+W_0^\top + W_0 + W_0^\top W_0 = \mathtt{I} + W_0^\top W_0$ has only eigenvalues ≥ 1 and therefore $(\mathtt{I} + W_0^\top W_0)^{-1}$ has all eigenvalues $\in [0, 1]$. Finally,

$$\|(\mathtt{I} + W_0)^{-1}\|_2 = \sqrt{\lambda_{\max}((\mathtt{I} + W_0^\top W_0)^{-1})} \in [0, 1] \tag{16}$$

and the mapping $x \mapsto (\mathtt{I}+W_0)^{-1}x$ is 1-Lipschitz. In Appendix C we show that this property carries over to element-wise non-expansive activation functions. This is exciting as it means the network has a degree of built-in robustness to adversarial perturbations [29].

4 Experiments

We train MLPs with 4 hidden layers with 500 neurons each, using the three special cases $\lambda = 1$ (HM), $\lambda = 1/2$ (FFM) and $\lambda = 0$ (SSHM). The models were trained on MNIST and FashionMNIST. EP (which our learning dynamics reduces to for $\lambda = 1$) typically works best with bounded activation functions so we trained all models with a tempered hard sigmoid activation function $\sigma(x) = \max(0,\ \min(1,\ \frac{x}{2}))$. This activation function was introduced in [30] to reduce the degree of saturation (neurons with value 0 or value 1). For reduced runtime we used undampened fixed-point iterations when possible (FFM and HM) and dampened one otherwise (SSHM) in order to ensure stable inference.

Weak nudging. We carry out experiments applying weak centered nudging (analogous to the centered nudging introduced in [30]). As shown in table 1 each of the models manage to solve both the MNIST and Fashion MNIST tasks, although the HM model lacks a bit behind on MNIST. As predicted by the theory, the upper bound on the sensitivity (measured by the Lipschitz constant L_{hidden}) of the hidden units to changes in their input $y = \theta_0 x$ is smallest for the skew-symmetric Hopfield model. We also report the Lipschitz estimate L_{output} for the output units as a function of the network input x, where L_{output} is the spectral norm of

$$(\mathtt{I}\ \mathtt{0} \cdots \mathtt{0})(\mathtt{I} - W_\lambda)^{-1} \begin{pmatrix} \vdots \\ \mathtt{0} \\ \theta_0 \end{pmatrix}. \tag{17}$$

The symmetric Hopfield model achieves a smaller L_{output} than the feedforward model, which is in line with a recent study finding EP to be more adversarially robust than feedforward models [31]. Figure 2b and 2e show that the SSHM model consistently generalizes better (difference between validation and training performance is smaller). Figure 2c and 2f show that the fraction of saturated neurons is highest for the HM model and lowest for the SSHM model.

Strong nudging. Although EP and DP to a large extent are motivated as algorithms for new non-von Neumann compute platforms they require β to be small. This is problematic as a small nudging signal $\pm\beta\ell'$ can drown out in noisy circuits. We carry out an experiment where all settings are as in the weak feedback experiments except that for each epoch we increment β by 0.05. As expected the use of positive feedback in the symmetric Hopfield model leads to rapid performance degradation. The use of negative feedback in the skew-symmetric Hopfield model permits stable learning with significantly stronger nudging strength.

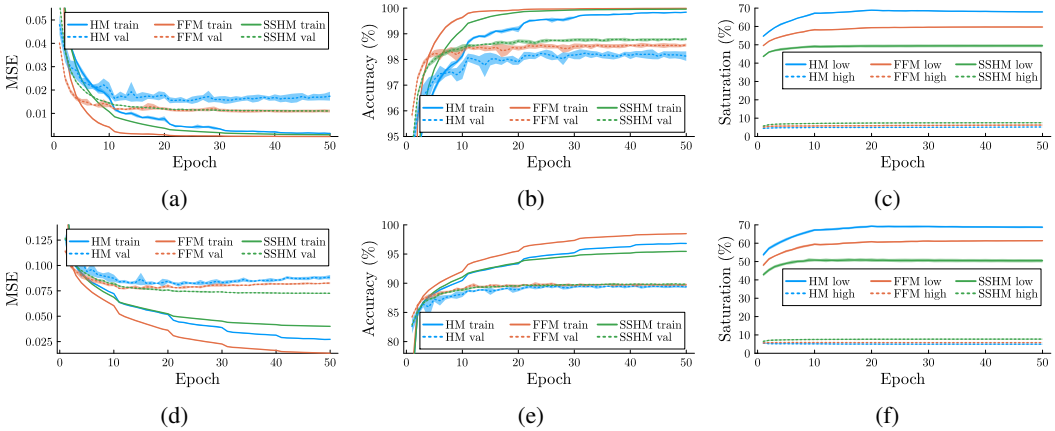
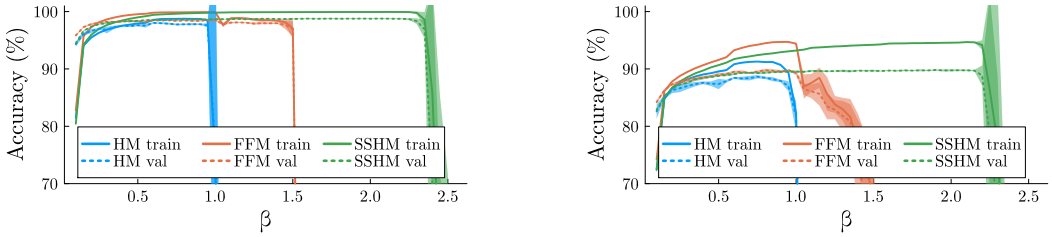


Figure 2: MNIST (top) and FashionMNIST (bottom) loss (1st column), accuracy (middle) and fraction of saturated neural activations (right) for a 500-500-500-500-10 hard sigmoid network. Shaded areas denote one standard deviation.



(a) MNIST (b) FashionMNIST

Figure 3: Impact of gradually increasing β by 0.05 per epoch for MNIST and FashionMNIST.

5 Discussion

The proposed saddle-point framework in Eq. 4 encompasses training of a spectrum of algorithms parameterized in terms of λ. From a physical implementation perspective, only the Hopfield model with symmetric, skew-symmetric, and strictly lower triangular (feedforward model) connectivity matrices are truly appealing as they avoid excessive weight copying and multiplexing. Conveniently, such models can be implemented in analog compute platforms with programmable weights. For example, spatial light modulators allow one to program weights with up to 10 bit precision [32]. With such a device, the matrix-vector multiplication could be efficiently computed in the optical domain using light, while the required nonlinearities and other operations could be further realized in the analog electronic domain, resulting in the efficient opto-electronic Hopfield networks. In practice, real physical systems suffer from various nonidealities, which may result in approximately symmetric or skew-symmetric matrices. The impact of such physical imperfections on the dynamical properties of Hopfield-like networks is an exciting direction to investigate.

Due to the negative feedback connections the skew-symmetric Hopfield model exhibits a strong inherent degree of robustness to perturbations, and is able to learn in the presence of a strong teaching signal. These are aspects we expect to be important in noisy substrates such as the brain as well as neuromorphic hardware. We hope this work will encourage further research into the role of negative feedback in learning.

Outlook. Due to hardware concerns we have not explored the general case $\lambda \notin \{0, 1/2, 1\}$. One use for $\lambda \notin \{0, 1/2, 1\}$ would be to start from a pre-trained feedforward model and slowly alter λ while fine-tuning the weights. This would allow a kind of translating between between feedforward ($\lambda = 1/2$), Hopfield models ($\lambda = 1$) and skew-symmetric Hopfield models ($\lambda = 0$). It is also possible to choose λ on a layer by layer basis. This would be an alternative way to approach the joint training of chained feedforward and Hopfield models proposed in [33].

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Chalmers AI Research Centre (CHAIR). The experiments were enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation. Many thanks to Jannes Gladrow for valuable guidance and suggestions during the early stages of this project and to Jack Kendall and Benjamin Scellier for helpful discussions, regarding implementations of negative feedback in resistive networks.

References

[1] James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

[2] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[3] Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pages 10–17. Elsevier, 1991.

[4] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[5] Menachem Stern, Daniel Hexner, Jason W Rocks, and Andrea J Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2):021045, 2021.

[6] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *Elife*, 6:e22901, 2017.

[7] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

[8] Rasmus Høier, D Staudt, and Christopher Zach. Dual propagation: Accelerating contrastive hebbian learning with dyadic neurons. In *International Conference on Machine Learning*, pages 13141–13156. PMLR, 2023.

[9] Axel Laborieux and Friedemann Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Advances in Neural Information Processing Systems*, 35:12950–12963, 2022.

[10] Rodolphe Sepulchre, Guillaume Drion, and Alessio Franci. Control across scales by positive and negative feedback. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):89–113, 2019.

[11] Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

[12] Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

[13] Sam Dillavou, Menachem Stern, Andrea J Liu, and Douglas J Durian. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1):014040, 2022.

[14] Su-in Yi, Jack D Kendall, R Stanley Williams, and Suhas Kumar. Activity-difference training of deep neural networks using memristor crossbars. *Nature Electronics*, 6(1):45–51, 2023.

[15] Benjamin Scellier. A fast algorithm to simulate nonlinear resistive networks. *arXiv preprint arXiv:2402.11674*, 2024.

[16] Konrad P Körding and Peter König. Supervised and unsupervised learning with two sites of synaptic integration. *Journal of computational neuroscience*, 11:207–215, 2001.

[17] Armin Askari, Geoffrey Negiar, Rajiv Sambharya, and Laurent El Ghaoui. Lifted neural networks. *arXiv preprint arXiv:1805.01532*, 2018.

[18] Jia Li, Cong Fang, and Zhouchen Lin. Lifted proximal operator machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4181–4188, 2019.

[19] Christopher Zach and Virginia Estellers. Contrastive learning for lifted networks. In Kirill Sidorov and Yulia Hicks, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 163.1–163.12. BMVA Press, 9 2019.

[20] Anna Choromanska, Benjamin Cowen, Sadhana Kumaravel, Ronny Luss, Mattia Rigotti, Irina Rish, Paolo Diachille, Viatcheslav Gurev, Brian Kingsbury, Ravi Tejwani, et al. Beyond backprop: Online alternating minimization with auxiliary variables. In *International Conference on Machine Learning*, pages 1193–1202. PMLR, 2019.

[21] Fangda Gu, Armin Askari, and Laurent El Ghaoui. Fenchel lifted networks: A lagrange relaxation of neural network training. In *International Conference on Artificial Intelligence and Statistics*, pages 3362–3371. PMLR, 2020.

[22] Christopher Zach. Bilevel programs meet deep learning: A unifying view on inference learning methods. *arXiv preprint arXiv:2105.07231*, 2021.

[23] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pages 10–19, 2014.

[24] Rasmus Høier and Christopher Zach. Lifted regression/reconstruction networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2020.

[25] Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Updates of equilibrium prop match gradients of backprop through time in an rnn with static input. *Advances in neural information processing systems*, 32, 2019.

[26] Eric Goles. Antisymmetrical neural networks. *Discrete Applied Mathematics*, 13(1):97–100, 1986.

[27] Jehoshua Bruck. On the convergence properties of the hopfield model. *Proceedings of the IEEE*, 78(10):1579–1585, 1990.

[28] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[29] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[30] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

[31] Siddharth Mansingh, Michal Kucer, Garrett Kenyon, Juston Moore, and Michael Teti. How robust are energy-based models trained with equilibrium propagation? *arXiv preprint arXiv:2401.11543*, 2024.

[32] Kirill P Kalinin, George Mourgias-Alexandris, Hitesh Ballani, Natalia G Berloff, James H Clegg, Daniel Cletheroe, Christos Gkantsidis, Istvan Haller, Vassily Lyutsarev, Francesca Parmigiani, et al. Analog iterative machine (aim): using light to solve quadratic optimization problems with mixed variables. *arXiv preprint arXiv:2304.12594*, 2023.

[33] Timothy Nest and Maxence Ernoult. Towards training digitally-tied analog blocks via hybrid gradient computation. *arXiv preprint arXiv:2409.03306*, 2024.

[34] Jirí V Outrata. A note on the usage of nondifferentiable exact penalties in some special optimization problems. *Kybernetika*, 24(4):251–258, 1988.

[35] Arkadij Semenovic Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[36] Thibaut Lienart. Mirror descent algorithm. `https://tlienart.github.io/posts/2018/10/27-mirror-descent-algorithm/`. Accessed: 2024-05-13.

[37] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[38] Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richad Hartley, and Puneet Dokania. Mirror descent view for neural network quantization. In *International conference on artificial intelligence and statistics*, pages 2809–2817. PMLR, 2021.

A Deriving the objective

Consider the following constrained optimization problem.

$$\min_W \ell(s,y) \quad \text{s.t. } \forall \text{ k } s_k = \arg\min_{s'_k} E_k(s'_k, s_{\setminus k}, W_k) \tag{18}$$

We introduce the notation s_k denoting the k'th neuron in the state vector s and $s_{\setminus k}$ denoting the state vector s with the k'th element removed. $W_{k\setminus k}$ denotes the $k'th$ row of W with the $k'th$ element removed, but for brevity we omit this argument in the following and simply write $E_k(s_k, s_{\setminus k})$. Furthermore we make the assumption that $E_k(s_k, s_{\setminus k})$ has a unique minimizer s_k^*. Next we define $\bar{s} := \frac{1}{2}(s^+ + s^-)$ then we can rewrite this as

$$\begin{aligned} \min_W \min_{s^+} \max_{s^+} \tfrac{1}{2}\ell(s^+,y) + \tfrac{1}{2}\ell(s^-,y) \quad \text{s.t. } \forall \text{ k } s_k^+ &= \arg\min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}) \\ s_k^- &= \arg\min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}). \end{aligned} \tag{19}$$

Since we (at the moment) are strictly enforcing $s_k^+ = s_k^- = s_k^*$ via the constraints the additional minimization and maximization does not influence the objective and problem 19 is equivalent to problem 18. The constraints associated with $s_k^\pm$ can now be rewritten using the optimal value reformulation (OVR) [34] yielding

$$\begin{aligned} &\min_W \min_{s^+} \max_{s^+} \tfrac{1}{2}\ell(s^+,y) + \tfrac{1}{2}\ell(s^-,y) \\ &\text{s.t. } \forall \text{ k} \begin{cases} E_k(s_k^+, \bar{s}_{\setminus k}) \leq \min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}) \\ E_k(s_k^-, \bar{s}_{\setminus k}) \leq \min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}) \end{cases} \end{aligned} \tag{20}$$

Rather than strictly enforcing these constraints we add penalty terms that enforce that approximately enforce the constraint weighted by a gain factor $1/\beta$, where $\beta > 0$. In the limit $\beta \to 0$ the constraints are satisfied. The constraints associated with the s^- (a maximization) results in the penalty term

$$\tfrac{1}{\beta} \sum_k \min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}) - E_k(s_k^-, \bar{s}_{\setminus k}) \tag{21}$$

and the constraints associated with the minimization over s^+ results in the penalty term

$$\tfrac{1}{\beta} \sum_k E_k(s_k^+, \bar{s}_{\setminus k}) - \min_{s'_k} E_k(s'_k, \bar{s}_{\setminus k}). \tag{22}$$

After accounting for terms canceling out the inclusion of these penalty terms yield the following Relaxation of problem 18.

$$\min_W \min_{s^+} \max_{s^-} \tfrac{1}{2}\ell(s^+,y) + \tfrac{1}{2}\ell(s^-,y) + \tfrac{1}{\beta} \sum_k E_k(s_k^+, \bar{s}_{\setminus k}) - E_k(s_k^-, \bar{s}_{\setminus k}) \tag{23}$$

A.1 Specializing to the Hopfield energy case

Next we consider the special case of a Hopfield-like energy. We explicitly define an input projection weight matrix θ_0 that maps data x to the space of the activities s. By $\theta_{0,k}$ we denote the $k'th$ row of θ_0.

$$E_k(s_k, s_{\setminus k}) = G(s_k) - s_k W_{k\setminus k} s_{\setminus k} - s_k \theta_{0,k} x \tag{24}$$

Remark A.1. This formulation yields closed form updates of the form $s = f(Ws + \theta_0 x)$ the typical Hopfield energy employed in EP and CHL yields dynamics $s = f(\frac{1}{2}(W + W^\top)s + \theta_0 x)$. So in CHL and EP only the symmetric part of the weight matrix is used. In the setting we are considering here both the symmetric and skew-symmetric part takes part in the dynamics.

With this choice of energy equation 23 becomes

$$\min_{\{W,\theta_0\}} \min_{s^+} \max_{s^+} \tfrac{1}{2}\ell(s^+, y) + \tfrac{1}{2}\ell(s^-, y) + \tfrac{1}{\beta}\left(G(s^+) - G(s^-) - (s^+ - s^-)(W\bar{s} + \theta_0 x)\right). \tag{25}$$

We rewrite the term $(s^+ - s^-)W\bar{s}$ as

$$\begin{aligned}(s^+ - s^-)W\bar{s} = \tfrac{1}{4}\Big(& s^{+\top}(W + W^\top)s^+ + s^{+\top}(W - W^\top)s^- \\ & - s^{-\top}(W + W^\top)s^- - s^{-\top}(W - W^\top)s^+\Big),\end{aligned} \tag{26}$$

which allows us to rewrite the entire objective as

$$\begin{aligned}&\min_{\{W,\theta_0\}} \min_{s^+} \max_{s^+} \tfrac{1}{2}\ell(s^+, y) + \tfrac{1}{2}\ell(s^-, y) \\ &+ \tfrac{1}{\beta}\Bigg(G(s^+) - G(s^-) - (s^+ - s^-)^\top \theta_0 x \\ &- \tfrac{1}{4}\begin{pmatrix} s^+ \\ s^- \end{pmatrix}^\top \underbrace{\begin{pmatrix} W + W^\top & W - W^\top \\ -W + W^\top & -W - W^\top \end{pmatrix}}_{\mathcal{W}(W)} \begin{pmatrix} s^+ \\ s^- \end{pmatrix} \Bigg).\end{aligned} \tag{27}$$

There are four cases of primary interest here.

- If we do not impose any structure upon W (beyond not allowing self-connections), then equation 27 simply gives us a principled way to train models with arbitrary connectivity. This is significant as CHL and EP rely on symmetric connectivity (see remark A.1) and DP relies on lower triangular (feedforward) connectivity. The drawback is that at training time neurons propagate errors/differences $\frac{1}{2}(s^+ - s^-)$ in both bottom-up and top-down direction and the mean $\frac{1}{2}(s^+ + s^-)$) in both directions as well. This is a more complicated type of multiplexing than the kind used in DP (where errors flow from top to bottom and means flow from bottom to top).

- Restricting W to be lower triangular. In this setting W transports the mean of activities, $\frac{1}{2}(s^+ + s^-)$, forward and $W^\top$ transports the difference backwards $\frac{1}{2}(s^+ - s^-)$. This is equivalent to dual propagation in a feedforward model.

- If W is symmetric, then $W - W^\top = 0$, so two blocks of $\mathcal{W}$ vanish, and s^+ and s^- decouple, permitting us to infer them sequentially. This is equivalent to equilibrium propagation in a Hopfield model.

- If W is skew-symmetric, then $W + W^\top = 0$, so two blocks of $\mathcal{W}$ vanish. Only the blocks governing interactions between s^+ and s^- survive. In general this will require inferring s^+ and s^- simultaneously. However, in the case of a layered network (achieved by introducing a particular block sparsity into W), then the training dynamics decouple into one minmax problem over odd layers of s^+ and even layers of s^- and another minmax problem over even layers of s^+ and odd layers of s^-.

The three structured weight matrices (lower triangular, symmetric and skew-symmetric) and any smooth combinations thereof can be expressed as $W_\lambda = \theta + (2\lambda - 1)\theta^\top$, where θ is a lower triangular matrix. The case $\lambda = 1$ yields a symmetric Hopfield model, the case $\lambda = 1/2$ yields a feedforward model and the case $\lambda = 0$ yields a skew-symmetric Hopfield model. Written in terms of W_λ equation 27 becomes

$$\min_{\{\theta,\theta_0\}} \min_{s^+} \max_{s^+} \tfrac{1}{2}\ell(s^+, y) + \tfrac{1}{2}\ell(s^-, y) \tag{28}$$
$$+ \tfrac{1}{\beta}\Bigg(G(s^+) - G(s^-) - (s^+ - s^-)^\top \theta_0 x$$
$$- \tfrac{1}{4}\begin{pmatrix} s^+ \\ s^- \end{pmatrix}^\top \begin{pmatrix} \lambda(\theta + \theta^\top) & \bar{\lambda}(\theta - \theta^\top) \\ \bar{\lambda}(-\theta + \theta^\top) & \lambda(-\theta - \theta^\top) \end{pmatrix} \begin{pmatrix} s^+ \\ s^- \end{pmatrix} \Bigg).$$

This formulation is less general than equation 27, but encompasses the three hardware friendly cases. Furthermore, the ability to smoothly interpolate between these models may open the door to efficient pretraining of (skew-) symmetric Hopfield models (by defining an appropriate schedule for λ).

B Mirror descent/ascent dynamics

Here we show that inference of states in the dyadic models can be viewed as mirror descent/ascent (a generalization of the projected gradient method) [35]. In projected gradient descent we aim to take a negative gradient step on an objective $\mathcal{L}$ with respect to a variable s subject to the constraint that the the updated value s_{t+1} is inside a chosen domain C [36] (note that we here use subscripts to denote timesteps rather than layers). This can be formulated as the following problem

$$s_{t+1} = \arg\min_{s \in C} \left\{ s^\top \nabla_s \mathcal{L} + \frac{1}{\eta} ||s - s_t||_2^2 \right\}. \tag{29}$$

This yields the updates $s_{t+1} = \pi_C(s_t - \eta \nabla_s E)$, where π_C is a projection onto the domain C.

In mirror descent the squared Euclidean distance penalizer in equation 29 is replaced with with a Bregman divergence. Bregman divergences are non-negative, but not necessarily symmetric. Familiar examples of Bregman divergences include the squared Euclidean distance, the KL divergence and the Mahalanobis distance. The Bregman divergence between s and y induced by a convex function G is equal to $G(s)$ minus its first order Taylor expansion at y evaluated at s.

$$B_G(s, y) = G(s) - \big(G(y) + (s - y)^\top \nabla G(y)\big) \tag{30}$$

Mirror descent with an appropriately chosen Bregman divergence can yield better convergence rates and in some cases turn constrained optimization problems into unconstrained optimization problems [37]. This has recently been used to train quantized neural networks [38].

Applying mirror descent to the Hopfield setting We restrict our analysis to the setting where G is chosen such that it satisfies $\nabla G(u) = f^{-1}(u)$, where f is a (typically nonlinear) activation function. We use as objective $\beta\mathcal{L}_\lambda$ (scaling eq. 4 simplifies things). Using a Bregman divergence B_G induced by G we can replace equation 29 with the following:

$$s^+_{t+1} = \arg\min_{s^+ \in C} \left\{ s^{+\top} \nabla_{s^+_t} \beta\mathcal{L}_\lambda + \frac{1}{\eta} B_G(s^+, s^+_t) \right\}. \tag{31}$$

$$s^-_{t+1} = \arg\max_{s^- \in C} \left\{ s^{-\top} \nabla_{s^-_t} \beta\mathcal{L}_\lambda - \frac{1}{\eta} B_G(s^-, s^-_t) \right\}. \tag{32}$$

Next we solve for first order optimality with respect to s using the definition of B_G and the fact that $B_G(s^+, s^+_t)$ is convex with respect to its first argument.

$$0 = \pm\nabla_{s^\pm} \beta\mathcal{L}_\lambda + \tfrac{1}{\eta}(\nabla G(s^\pm_{t+1}) - \nabla G(s^\pm_t)) \tag{33}$$
$$\iff$$
$$s^\pm_{t+1} = f(f^{-1}(s^\pm_t) \mp \eta \nabla_{s^\pm_t} \beta\mathcal{L}_\lambda), \tag{34}$$

Using that $\nabla_{s_t^\pm} \beta \mathcal{L}_\lambda = \pm G(s^\pm) \mp \lambda(\theta + \theta^\top)s^\pm \mp \bar{\lambda}(\theta - \theta^\top)s^\mp \mp \theta_0 x \pm \beta \ell'$, and expressing things in terms of the preactivations $a^\pm := f^{-1}(s)$ yields the following dampened fixed point iterations.

$$a_{t+1}^\pm = (1-\eta)a_t^\pm + \eta\left(\lambda(\theta+\theta^\top)s^\pm + \bar{\lambda}(\theta-\theta^\top)s^\mp + \theta_0 x \mp \beta\ell'\right) \tag{35}$$

Mirror descent allows us to avoid the derivative of the activation function and ensures that the updated states are always feasible (this is tricker to ensure when using projected gradient descent). Non-invertible activation functions such as the sign and relu functions can be seen as limit cases of invertible activation functions such as tanh and leaky-relu.

C Adding element-wise contractive non-linearities

We apply implicit differentiation on

$$\nabla G(s) = Ws + x \implies \nabla^2 G(s)\frac{ds}{dx} = W\frac{ds}{dx} + \mathtt{I}.$$

We introduce $\mathtt{D} := \nabla^2 G(s) = (f'(s))^{-1}$, which is a diagonal matrix with elements ≥ 1 on the diagonal (under the assumption that the activation mapping is 1-Lipschitz, i.e. non-expansive). Therefore

$$\frac{ds}{dx} = (\mathtt{D} - W)^{-1}.$$

On the other hand

$$\mathtt{D} - W = \mathtt{D}^{1/2}(\mathtt{I} - \underbrace{\mathtt{D}^{-1/2}W\mathtt{D}^{-1/2}}_{=:\tilde{W}})\mathtt{D}^{1/2} \tag{36}$$

with $\tilde{W}$ also being skew-symmetric. Consequently

$$\|(\mathtt{D} - W)^{-1}\|_2 = \|\mathtt{D}^{-1/2}(\mathtt{I} - \tilde{W})^{-1}\mathtt{D}^{-1/2}\|_2 \leq \|\mathtt{D}^{-1}\|_2 \|(\mathtt{I} - \tilde{W})^{-1}\|_2 \leq 1 \tag{37}$$

since $\mathtt{D}^{-1}$ is a diagonal matrix with entries in $[0, 1]$ and $\|(\mathtt{I} - \tilde{W})^{-1}\|_2 \leq 1$ for any skew-symmetric matrix $\tilde{W}$ (as shown in Section 3, Eq. 16).