Scaling Equilibrium Propagation to Deeper Neural Network Architectures

Sankar Vinayak E P
Dept. of Computer Science and Engineering,
Indian Institute of Technology, Madras
cs24m041@smail.iitm.ac.in

Gopalakrishnan Srinivasan
Dept. of Computer Science and Engineering,
Robert Bosch Centre for Data Science and AI,
Indian Institute of Technology, Madras
sgopal@cse.iitm.ac.in

arXiv:2509.26003v1 [cs.NE] 30 Sep 2025

***Abstract*—Equilibrium propagation has been proposed as a biologically plausible alternative to the backpropagation algorithm. The local nature of gradient computations, combined with the use of convergent RNNs to reach equilibrium states, make this approach well-suited for implementation on neuromorphic hardware. However, previous studies on equilibrium propagation have been restricted to networks containing only dense layers or relatively small architectures with a few convolutional layers followed by a final dense layer. These networks have a significant gap in accuracy compared to similarly sized feedforward networks trained with backpropagation. In this work, we introduce the Hopfield-Resnet architecture, which incorporates residual (or skip) connections in Hopfield networks with clipped ReLU as the activation function. The proposed architectural enhancements enable the training of networks with nearly twice the number of layers reported in prior works. For example, Hopfield-Resnet13 achieves 93.92% accuracy on CIFAR-10, which is ≈3.5% higher than the previous best result and comparable to that provided by Resnet13 trained using backpropagation. Our implementation source code is available at https://github.com/BrainSeek-Lab/Scaling-Equilibrium-Propagation-to-Deeper-Neural-Network-Architectures.**

I. Introduction

Advancements in deep learning, powered by artificial neural networks (ANNs), have formed the backbone of rapid growth in modern artificial intelligence. These improvements can be attributed to backpropagation (BP) based training algorithms, which solve the error credit assignment problem by applying chain rule to propagate gradients from the output layer. However, backpropagation is regarded as biologically implausible since the underlying gradient computations are non-local and require access to global network information. Alternative algorithms such as forward propagation [1], predictive coding [2], and no-prop [3] have been proposed to address the non-locality issue of backpropagation. These emerging algorithms offer the potential for on-device learning with higher energy efficiency compared to backpropagation.

Contrastive Hebbian [4] learning is one such algorithm that has been shown to converge to a steady state over time for static input, effectively operating as a static convergent recurrent neural network (RNN). Equilibrium propagation (EP) [5] is a category of contrastive Hebbian learning rule based on the network energy function, which is computed using both the neuronal states and the network parameters. The EP algorithm operates in two phases. Initially, the network dynamics are governed exclusively by the energy function. Subsequently, a weak clamping force proportional to the loss function of the output layer is applied, via a weighting parameter, causing the network to reach an equilibrium state with reduced loss and lower total energy. The combination of input layer clamping and weak output layer clamping enables gradient computation without requiring backpropagation across the entire network. The credit assignment problem is solved by leveraging the difference in the gradient of the energy function with respect to network parameters between the two phases, thereby providing a learning rule with greater biological plausibility. Previous works have implemented EP in hardware, demonstrating its potential for on-device learning [6], [7].

The primary limitation of EP is that, although the gradient computed by this method closely matches that obtained using backpropagation through time [8], its performance is typically lower than that of a comparable feedforward network trained with backpropagation. In particular, the bias on the estimated gradient arising from the theoretical requirement for infinitely small nudging introduces noise into the computations, corrupting the gradient and thereby hindering learning. This issue was addressed by methods such as centered equilibrium propagation (CEP) [9] and holomorphic equilibrium propagation (HEP) [10], which increase the number of nudging phases during network training. However, these methods have been validated primarily on shallow networks (≤ 6 trainable layers), which still exhibit performance degradation compared to that achieved through backpropagation. We propose architectural enhancements to improve the scalability and performance of EP-based training. Overall, the key contributions of our work are as follows.

- We propose *clipped ReLU* activation function to simplify the energy function and gradient computation.
- We introduce the residual Hopfield network architecture, termed *Hopfield-Resnet*, which consists of residual or skip connections that enable EP-based methods to successfully train deeper networks (> 12 layers) with minimal performance loss relative to backpropagation baselines.
- We validate the proposed Hopfield-Resnet architecture

This work was supported in part by the RISC-V Knowledge Centre of Excellence (RKCoE), sponsored by the Ministry of Electronics and Information Technology (MeitY).

and training methodology across the CIFAR-10, CIFAR-100, and Fashion MNIST datasets.

II. Related Works

A. Static Convergent RNN

In a supervised training setting, given an input (x), the network is trained to produce the required output (y). Equilibrium propagation, on the contrary, uses a convergent RNN, wherein the network evolves its neuronal state s to a steady state s_* over time for the input x. An energy function Φ is computed based on the neuronal state s_t and the network parameters θ. The network evolves as

$$s_{t+1} = \frac{\partial \Phi(x, s_t, \theta)}{\partial s}, \tag{1}$$

and the equilibrium state $s_*(= s_{t+1} = s_t)$ is specified by

$$s_* = \frac{\partial \Phi(x, s_*, \theta)}{\partial s}. \tag{2}$$

That is, the network converges to steady state. The parameters are updated in such a way that, at equilibrium, the state of the output neurons matches the expected output y.

B. Equilibrium Propagation

Equilibrium propagation (EP) [5] was originally proposed for continuous-time dynamics. Subsequent works extended the method to support discrete-time dynamics [8]. EP computes the gradient in two phases. During the first phase (*free* phase), the input layers of the network are clamped to x, and the network is allowed to evolve based only on the energy function without any label information. In the second phase (or *weakly clamped* phase), the output layer is perturbed by an additional term proportional to gradient of the loss L with respect to the neuronal states, with its magnitude scaled by the parameter β. The updated neuronal dynamics is specified by

$$s_{t+1} = \frac{\partial \Phi(x, s_t, \theta)}{\partial s} + \beta \frac{\partial L(x, s_t, \theta)}{\partial s}. \tag{3}$$

The network then converges to new steady state s_*^β. It has been shown that if the energy function is differentiable with respect to both β and the network parameters θ, the gradient of the loss function L with respect to the parameters can be obtained from the gradient of the energy function at equilibrium as

$$-\frac{\partial L}{\partial \theta} = \frac{1}{\beta}\left[\frac{\partial \Phi(x, s_*^\beta, \theta)}{\partial \theta} - \frac{\partial \Phi(x, s_*, \theta)}{\partial \theta}\right]. \tag{4}$$

Equation 4 holds as $\beta \to 0$.

C. Centered Equilibrium Propagation

The theoretical requirement for an infinitesimally small β (refer Equation 4) introduces gradient estimator bias for non-zero values of beta, thereby limiting the practical application of the vanilla EP method. The centered equilibrium propagation (CEP) algorithm [9] mitigates this by computing the gradient using both $+\beta$ and $-\beta$, and using a second-order approximation for improving the gradient estimation. The equation then becomes

$$-\frac{\partial L}{\partial \theta} = \frac{1}{2\beta}\left[\frac{\partial \Phi(x, s_*^{+\beta}, \theta)}{\partial \theta} - \frac{\partial \Phi(x, s_*^{-\beta}, \theta)}{\partial \theta}\right]. \tag{5}$$

Further research has focused on computing stable equilibrium at multiple points within a finite-sized oscillation of radius β, along the complex plane, to further reduce the bias in gradient estimation [10]. This approach leads to the equation

$$\frac{\partial L}{\partial \theta} = \frac{1}{T|\beta|}\int_0^T \frac{\partial \Phi}{\partial \theta}\left(\theta, s_*^{\beta(t)}, \beta(t)\right) e^{-2i\pi t/T}\, dt, \tag{6}$$

where T is the period of the teaching signal (or the number of points for computing the equilibrium), $t \in [0, T]$, and $\beta(t) = |\beta| e^{-2i\pi t/T}$.

D. Convolutional Network with Equilibrium Propagation

The theory of EP, originally proposed for dense networks, has been extended to networks with convolutional operations. In a network with dense and convolutional layers, the energy function Φ is the sum of the contributions from each part [9], given by

$$\begin{aligned}\Phi(\theta, \{s^n\}) = &\sum_{n=0}^{N_{\text{conv}}-1} s^{n+1} \cdot \mathcal{P}(w_{n+1} \star s^n) \\ &+ \sum_{n=N_{\text{conv}}}^{N_{\text{tot}}-1} s^{n+1\top} w_{n+1} s^n\end{aligned} \tag{7}$$

where s^n is the neuronal state per layer, w denotes the weight matrix, $\mathcal{P}$ represents the pooling operation, and $\star$ indicates the convolution operation. The layer-wise state evolution can then be formulated as

$$s_{t+1}^n = \sigma\Big(\mathcal{P}\big(w_n \star s_t^{n-1}\big) + \tilde{w}_{n+1} \star \mathcal{P}^{-1}\big(s_t^{n+1}\big)\Big), \quad \text{for } 1 \le n \le N^{\text{conv}} \tag{8}$$

$$s_{t+1}^n = \sigma\Big(w_n s_t^{n-1} + w_{n+1}^\top s_t^{n+1}\Big), \quad \text{for } N^{\text{conv}} < n < N^{\text{tot}} \tag{9}$$

where σ is the activation function used, $\tilde{w}$ is the flipped kernel used for transpose convolution, and $\mathcal{P}^{-1}$ is the inverse pooling operation.

E. Asynchronous Update of Neuronal States

The inherent characteristics of state update dynamics in EP method cause extended convergence times for larger network architectures. Gradient computation without complete steady-state convergence introduces significant noise into the learning process. One method used to reduce the time to equilibrium is the asynchronous update of the neuronal state [11]. Instead of performing global state updates following a single energy evaluation, this method uses an alternating update scheme in which energy is computed to first update even-indexed layers, followed by energy recomputation with the new states to adjust odd-indexed layers within each iteration cycle. Although there

is no proof of convergence for this technique, it helps reduce the number of discrete iterations needed to reach equilibrium, thus shortening the overall training time.

III. Proposed Architecture

Previous works utilizing equilibrium propagation conducted experiments using relatively smaller networks, such as $VGG5$ with 4 convolutional layers and 1 dense layer [9], [11], [12], or $VGG6$ with 2 dense layers [10], or networks consisting only of dense layers [5], limiting the achievable depth and constraining performance across benchmark datasets. Our work addresses the scalability bottleneck by introducing *Hopfield-Resnet*, featuring residual connections between layers and modifying the activation function, both of which contribute towards improved training of deeper networks with higher accuracy.

A. Scaling Deeper with Residual Hopfield Network

The non-feedforward architecture of convergent RNNs imposes a primary scalability limitation. As the network becomes deeper, it requires more parameters and a longer time to reach steady state, thereby making training increasingly challenging. Our experiments indicate that, beyond a certain depth, adding more layers yields diminishing returns. Residual connections are a widely used technique for scaling networks deeper while ensuring training convergence [13]. In the case of a Hopfield network [14], there exists indirect interaction between different layers through the energy function. We further propose a residual Hopfield network, referred to as *Hopfield-Resnet*, which incorporates residual connections to improve the scalability, as shown in Fig. 1. The basic Hopfield-Resnet block consists of three convolutional operations: two forming the main pathway and one skip connection that directly links the final state of the preceding block to the final state of the current block. The convolution on the main path uses 3×3 kernels, while the one in skip connection uses 1×1 kernels. Residual connections are implemented in two ways: direct identity connections and connections using a 1×1 kernel. Both approaches were tested, with the latter demonstrating better performance and therefore selected for the experiments. The network architecture used in this work consists of four Hopfield-Resnet blocks and a dense layer. This design yields 13 sets of trainable parameters, with 12 convolution layers, dense output layer, and 9 neuronal states which can be updated.

The Hopfield-Resnet architecture enhances interactions between neuronal states during energy computation while preserving the energy function previously defined in Equation 7. The neuronal state update equation is modified so that, instead of considering only adjacent states, it accounts for all paths and states that directly interact with the current state. This leads

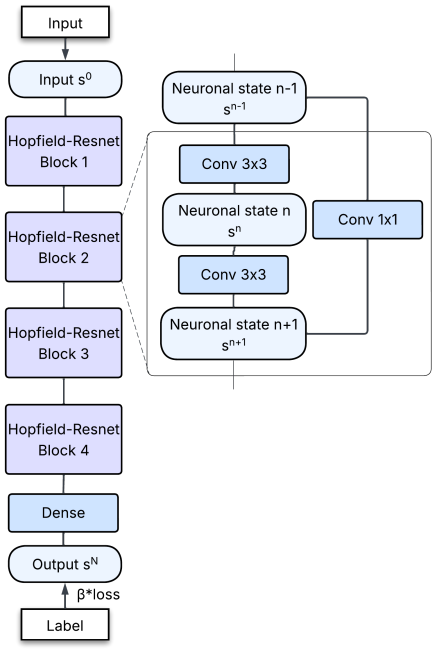

Fig. 1. Architecture of Hopfield-Resnet13, consisting of four Hopfield-Resnet blocks, each containing three convolutional layers (two of them in the main pathway and one skip connection), followed by a dense output layer.

to summation over all such interaction paths, as described by

$$s_{t+1}^{n} = \sigma\Big(\sum_{i=pre(n)} \mathcal{P}\big(w_i \star s_t^i\big) + \sum_{j=post(n)} \tilde{w}_j \star \mathcal{P}^{-1}\big(s_t^j\big)\Big), \tag{10}$$

$$\text{for } 1 \leq n \leq N^{\text{res}}$$

$$s_{t+1}^{n} = \sigma\Big(\sum_{i=pre(n)} w_i \cdot s_t^i + \sum_{j=post(n)} w_j^{\top} \cdot s_t^j\Big), \tag{11}$$

$$\text{for } N^{\text{res}} < n < N$$

where N^{res} is the number of neuronal states in the Hopfield-Resnet blocks, and N is the total number of neuronal states. The terms $pre(n)$ and $post(n)$ denote all previous and subsequent states relative to state n that interact directly with it. This modification increases the number of trainable parameters and causes the network to take a longer time to reach steady state. Neuronal states interact through the weight parameters interconnecting them. Since the network is not feedforward, symmetric weight matrix is used in both directions during energy computation. Additionally, the newly computed neuronal pre-activation state is passed through an activation function to compute the updated neuronal state. Details of the activation function are provided in the following section.

B. Alternative Activation Function

Previous works relied on specialized activation functions; for example, in HEP [10], the activation function is required to be holomorphic. Similarly, other implementations employed modified versions of the $sigmoid$ or $tanh$ functions, adjusted

so that their outputs remain within the interval $[0, 1]$. Bounding the energy function is essential to prevent explosive growth in energy values, which would otherwise destabilize the training process. We experimentally observed that existing activation functions limit the achievable accuracy as the network scales deeper. We propose a simplified bounded activation function, denoted as $\mathrm{ReLU}\alpha$, which restricts the output values to the range $[0, \alpha]$. A special case, $\mathrm{ReLU1}$, was previously introduced as an approximation to the $softmax$ function and shown to improve network accuracy [11]. In our experiments, we evaluated different initialization strategies for α, including random initialization. We adopted $\mathrm{ReLU6}$ for the final experiments reported in this work.

IV. Results

We validated the Hopfield-Resnet architecture on CIFAR-10, CIFAR-100, and Fashion-MNIST datasets. Centered equilibrium propagation (CEP, described in Section II-C) was implemented using Nesterov accelerated gradient optimizer [15]. The value of nudge parameter β was tuned empirically. Larger values ($\beta \geq 0.8$) prevented learning progress, while smaller values (for example, $1e-4$) allowed training but introduced instabilities and increased sensitivity to other hyperparameters. In contrast, values in the range $[0.1, 0.4]$ demonstrated greater stability during training. The experiments were performed on NVIDIA RTX 4090 and 6000 Ada GPUs using Pytorch.

As highlighted in Table I, our experiments outperformed the state-of-the-art accuracy reported for the deep convolutional Hopfield network (DCHN) [11] trained using EP on both CIFAR10 and CIFAR100. The experimental findings further indicate that our approach yielded results much closer to those achieved by standard backpropagation (BP) training on similar network architectures. As shown in Table II, the performance gap between the two methods has been significantly narrowed, at times matching the performance of BP, demonstrating the viability of this alternative training paradigm.

Dataset	Model Architecture	Prior Best (%)	Our work (%)
CIFAR-10	VGG5	90.3	92.84
	Hopfield-Resnet13	–	93.92
CIFAR-100	VGG5	68.4	70.78
	Hopfield-Resnet13	–	71.05
F-MNIST	VGG5	93.53	94.34
	Hopfield-Resnet13	–	94.15

TABLE I

ACCURACY COMPARISON ACROSS MODELS AND DATASETS FOR THE PROPOSED ARCHITECTURAL ENHANCEMENTS OVER THE BASELINE IMPLEMENTATION [11].

Dataset	Model Architecture	BackProp (%)	EquiProp (%)
CIFAR-10	VGG5	92.11	92.84
	Hopfield-Resnet13	93.78	93.92
CIFAR-100	VGG5	72.54	70.78
	Hopfield-Resnet13	75.12	71.05

TABLE II

ACCURACY COMPARISON OF MODELS TRAINED WITH EQUILIBRIUM PROPAGATION VERSUS BACKPROPAGATION. NOTE THAT BP WAS APPLIED TO THE FEEDFORWARD EQUIVALENT OF THE NETWORK.

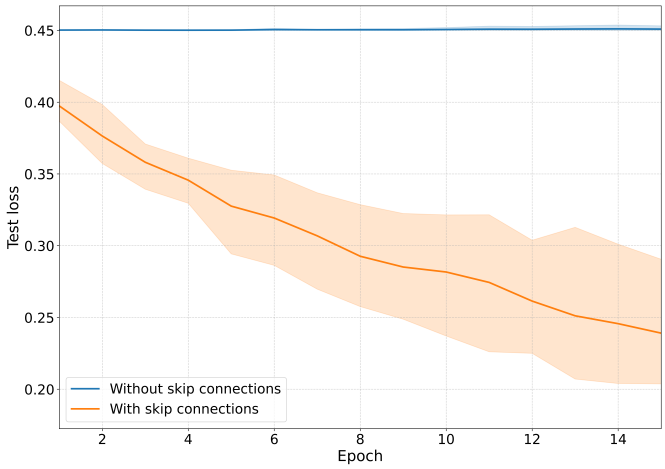

Fig. 2. Test loss for the Hopfield-Resnet13 architecture trained using centered equilibrium propagation (CEP), with and without the skip connections, on the CIFAR-10 test set.

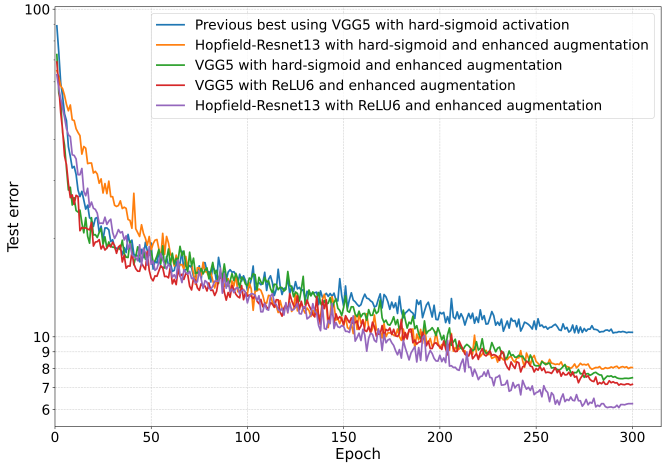

Fig. 3. Test error over epochs for VGG5 and Hopfield-Resnet architectures, shown for different combinations of activation function and data augmentation on the CIFAR-10 test set. Note that the y-axis is displayed in log scale.

A. Ablation Studies

Our experiments showed that CEP struggles to perform well on deeper networks without the proposed skip connections. Having more than five consecutive convolutional layers slows down the training process without significant improvements in accuracy. Fig. 2 shows the training loss progression for the aforementioned network architecture, both with and without skip connections, averaged over five experimental runs with varying hyperparameter combinations. Without skip connections, the training loss remains stagnant, showing negligible or no improvement over time. In contrast, when skip connections are included, the CEP algorithm successfully trains the model across a wide range of hyperparameters.

Regarding the activation function, we observed that setting $\alpha = 6$ ($\mathrm{ReLU6}$) yielded higher accuracy compared to $\mathrm{ReLU1}$ (hard-sigmoid function) for both VGG5 and Hopfield-Resnet architectures, as shown in Fig. 3. The performance of $\mathrm{ReLU}\alpha$,

where α is initialized uniformly at random to lie in the range $[0, 10]$, was found to be between that of $\mathrm{ReLU6}$ and $\mathrm{ReLU1}$. Fig. 3 further demonstrates that the overall performance improvement results from the combination of all the suggested modifications. While enhancements to the data augmentation pipeline alone allowed our algorithm to surpass the previous best results on CIFAR-10, they did not improve the accuracy for larger networks. To achieve performance comparable to similarly sized networks trained with backpropagation, adjustments to the activation function were also necessary. These modifications, along with residual connections, enabled deeper networks to outperform shallower ones. In summary, the proposed Hopfield-Resnet architecture with residual connections, trained using centered equilibrium propagation with $\mathrm{ReLU6}$ activation and enhanced input data augmentation, achieved the best accuracy compared to similarly sized baseline model.

B. Training Time

We adopted the same training methodology as in previous works, using a fixed number of timesteps rather than running the simulation until true equilibrium was reached [9]–[11]. For both the best-performing VGG5 and Hopfield-ResNet13 architectures, we used 120 timesteps in the free phase, followed by 50 timesteps each with $+\beta$ and $-\beta$ in the weakly clamped phase. Increasing the number of timesteps to 200 in the free phase and 60 in the clamped phase, without incorporating data augmentation or activation changes, did not enable the network to achieve its best performance.

The training time for the best performing Hopfield-Resnet13 architecture, with $\mathrm{ReLU6}$ and enhanced data augmentation, exceeded 30 hours for 300 epochs on an RTX 6000 Ada GPU. Analysis of the program execution showed that a considerable amount of time was expended on kernel launches and CPU-GPU synchronization during GPU training. This indicates that optimization opportunities exist similar to those in feedforward networks, and that substantial reductions in training time can be achieved by optimizing the implementation of CEP training algorithm for GPU execution.

C. Memory Utilization

The memory utilization of centered equilibrium propagation (CEP) during training phase is comparable to that of a similar feedforward network trained using backpropagation (BP). For Hopfield-Resnet13 model, with a batch size of 128 on CIFAR-10, CEP required 1612 MiB of GPU memory on an RTX 4090. In contrast, the feedforward BP counterpart (Resnet13), which includes additional batch normalization layers, consumed 1324 MiB. Notably, Resnet13 without batch normalization significantly underperforms, achieving only 89% test accuracy on the CIFAR-10 dataset. This suggests that equilibrium propagation, even without batch normalization, effectively handles data distribution through its energy function, thereby reducing the necessity for additional regularization.

D. Weight Distribution

Our experiments reveal that networks trained with centered equilibrium propagation (or CEP) exhibit distinctly different weight distributions compared to those obtained using backpropagation (or BP) training, underscoring their fundamentally different optimization objectives. CEP-trained networks tend to have both smaller absolute weight values and lower weight variance than their BP-trained counterparts. Fig. 4 illustrates these disparities: BP-trained networks exhibit a wider spread of weight values with relatively consistent distributions across layers, whereas CEP-trained networks have weights confined to a much narrower range. A notable trend emerges as network depth increases: in CEP-trained networks, weight magnitudes in deeper layers progressively approach zero, with this effect becoming more pronounced as depth increases. Although the initial layers maintain weight distributions similar to those in BP-trained networks, the deeper convolutional layers contain a much higher proportion of near-zero weights. The concentration of weights around zero can be partially attributed to weight decay, which is crucial for CEP performance. However, under the same weight decay setting, BP is unable to achieve the performance level of CEP. This tendency towards near-zero weight distributions could explain the difficulties CEP-trained models face when scaling to deeper architectures. Notably, the prevalence of near-zero values is considerably lower in layers with skip connections than in those along the direct connection path. This reduction in sparsity helps mitigate the challenges associated with training deeper networks using CEP.

V. Conclusion

In this work, we introduced residual connections between the hidden layers of convolutional Hopfield networks, enabling equilibrium propagation (EP) to scale to deeper networks than previously reported. In addition, we used $\mathrm{ReLU}\alpha$ as the nonlinear activation instead of hard-sigmoid to improve training stability. These architectural enhancements yielded significant accuracy gains on the CIFAR-10, CIFAR-100, and Fashion MNIST datasets, surpassing previous results and approaching the performance of comparably sized feedforward networks trained with backpropagation (BP). Recent efforts to apply EP to modern Hopfield networks [16] for sequence learning [17] highlight exciting directions for future research. Integrating these approaches with the architectures proposed in this work, and investigating other unique properties resulting from EP's distinct training methodology, represent promising avenues to extend EP's applicability.

Despite these promising directions, the practical adoption of EP as an alternative to BP faces notable challenges. Training deeper networks with EP remains computationally intensive due to the inherently sequential nature of the algorithm and the limitations of current GPU architectures, which are optimized for parallel processing rather than iterative computations. To fully realize the potential of EP, two critical advancements are required: the development of specialized hardware tailored to EP's computational demands, and algorithmic optimizations that better leverage existing hardware capabilities. Such innovations could significantly reduce training times and establish EP as a viable and efficient alternative to BP.

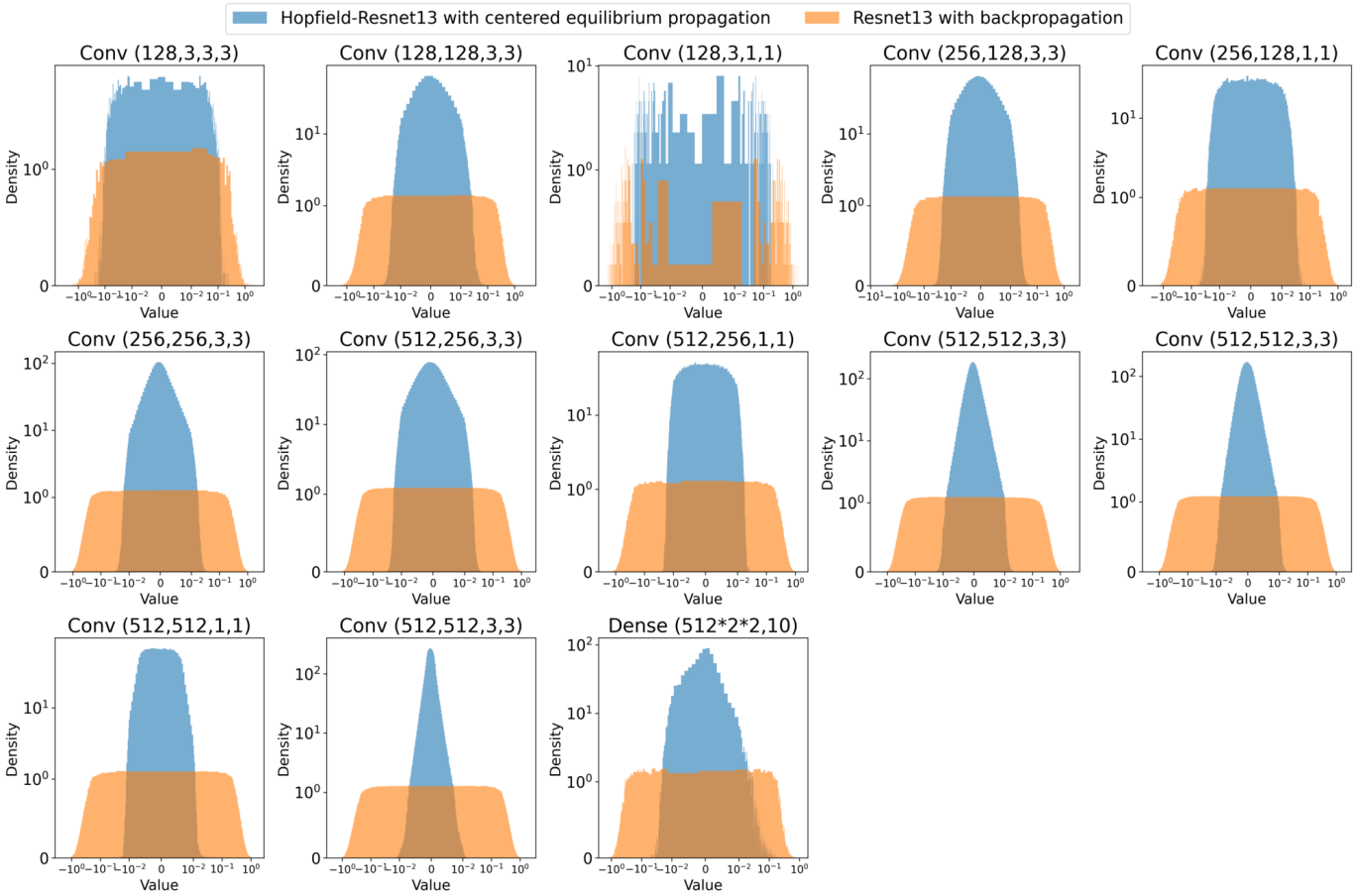

Fig. 4. Layer wise distribution of weight values in Resnet13 trained with backpropagation and Hopfield-Resnet13 trained with centered equilibrium propagation on the CIFAR-10 dataset.

REFERENCES

[1] G. Hinton, "The forward-forward algorithm: Some preliminary investigations," 2022. [Online]. Available: https://arxiv.org/abs/2212.13345

[2] K. Friston and S. Kiebel, "Predictive coding under the free-energy principle," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1521, pp. 1211–1221, 2009.

[3] Q. Li, Y. W. Teh, and R. Pascanu, "Noprop: Training neural networks without full back-propagation or full forward-propagation," 2025. [Online]. Available: https://arxiv.org/abs/2503.24322

[4] G. Detorakis, T. Bartley, and E. Neftci, "Contrastive hebbian learning with random feedback weights," 2018. [Online]. Available: https://arxiv.org/abs/1806.07406

[5] B. Scellier and Y. Bengio, "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation," *Frontiers in Computational Neuroscience*, vol. Volume 11 - 2017, 2017. [Online]. Available: https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2017.00024

[6] S. Oh, J. An, S. Cho, R. Yoon, and K.-S. Min, "Memristor crossbar circuits implementing equilibrium propagation for on-device learning," *Micromachines*, vol. 14, no. 7, 2023. [Online]. Available: https://www.mdpi.com/2072-666X/14/7/1367

[7] Z. Ji and W. Gross, "Towards efficient on-chip learning using equilibrium propagation," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[8] M. Ernoult, J. Grollier, D. Querlioz, Y. Bengio, and B. Scellier, "Updates of equilibrium prop match gradients of backprop through time in an rnn with static input," in *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/679742339917cea0e42a49a2fb7eb4cf4-Paper.pdf

[9] A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz, "Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias," *Frontiers in Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.633674

[10] A. Laborieux and F. Zenke, "Holomorphic equilibrium propagation computes exact gradients through finite size oscillations," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 12 950–12 963. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/545a114e655f9d25ba0d56ea9a01fc6e-Paper-Conference.pdf

[11] B. Scellier, M. Ernoult, J. Kendall, and S. Kumar, "Energy-based learning algorithms for analog computing: a comparative study," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 52 705–52 731. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/a52b0d191b619477cc798d544f4f0e4b-Paper-Conference.pdf

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554

[15] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.
[16] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "Hopfield networks is all you need," 2021. [Online]. Available: https://arxiv.org/abs/2008.02217
[17] M. Bal and A. Sengupta, "Sequence learning using equilibrium propagation," *arXiv preprint arXiv:2209.09626*, 2022.