
Can Local Learning Match Self-Supervised Backpropagation?

Wu S. Zihan¹ Ariane Delrocq¹ Wulfram Gerstner¹ Guillaume Bellec²

Abstract

While end-to-end self-supervised learning with backpropagation (global BP-SSL) has become central for training modern AI systems, theories of *local* self-supervised learning (local-SSL) have struggled to build functional representations in deep neural networks. To establish a link between global and local rules, we first develop a theory for deep linear networks: we identify conditions for local-SSL algorithms (like Forward-forward or CLAPP) to implement exactly the same weight update as a global BP-SSL. Starting from the theoretical insights, we then develop novel variants of local-SSL algorithms to approximate global BP-SSL in deep non-linear convolutional neural networks. Variants that improve the similarity between gradient updates of local-SSL with those of global BP-SSL also show better performance on image datasets (CIFAR-10, STL-10, and Tiny ImageNet). The best local-SSL rule with the CLAPP loss function matches the performance of a comparable global BP-SSL with InfoNCE or CPC-like loss functions, and improves upon state-of-the-art for local SSL on these benchmarks.

1. Introduction

As a model of brain plasticity, Backpropagation (BP) poses multiple problems (Crick, 1989; Lillicrap et al., 2020; Richards et al., 2019; Lillicrap et al., 2020). Whereas the forward pass of the BP algorithm might be an acceptable model of rapid biological neural computations (Yamins et al., 2014; Schrimpf et al., 2018), the backward pass requires a structured feedback machinery that is absent in biological networks (Figure 1 A). Biologically, learning is likely implemented by connectivity changes controlled by a Hebb rule, loosely paraphrased as "neurons that fire together wire together" (Hebb, 1949; Shatz, 1992). While many classic

experiments confirmed the Hebbian paradigm (Lomo, 1966; Castellucci et al., 1970; Bi & Poo, 2001; Caporale & Dan, 2008; Markram et al., 2011), the modern view emphasizes that Hebbian learning also depends on neuromodulators (Lisman et al., 2011; Yagishita et al., 2014; Bittner et al., 2017; He et al., 2015) that could signal predictions, expectations, reward, or novelty (Schultz et al., 1997; Gerstner et al., 2018). In parallel, in the modeling community, there has been a shift from Hebbian models (Oja, 1982; Barlow, 1989; Miller et al., 1989; Hyvarinen & Oja, 1998) to more general *local* learning rules in which Hebbian plasticity is influenced by neuromodulation or predictive mechanisms (Williams, 1992; Földiák, 1991; Rao & Ballard, 1999; Wiskott & Sejnowski, 2002; Whittington & Bogacz, 2017; Gerstner et al., 2018; Bredenberg et al., 2021; 2023). However, in practice, none of these local algorithms has worked as well as global BP in training deep neural networks (Bartunov et al., 2018; Lillicrap et al., 2020).

To train deep networks in a *supervised* setup with labeled data, local learning algorithms have been proposed (Roelfsema & van Ooyen, 2005; Lee et al., 2015; Lillicrap et al., 2016; Sacramento et al., 2018; Akrouf et al., 2019; Meulemans et al., 2022; Song et al., 2024; Salvatori et al., 2024). They can approximate BP in theory (Scellier & Bengio, 2017; Whittington & Bogacz, 2017) and approach its performance on benchmarks (Akrouf et al., 2019; Launay et al., 2020; Ernoult et al., 2022). However, for *self-supervised learning* without labeled data, the performance gap is larger, and we miss a theory to relate local-SSL algorithms with global representation learning principles. Existing partially-local learning rules for self-supervised learning cut the backward flow into blocks and gradient propagation is stopped outside of the block (Löwe et al., 2019; Kappel et al., 2023). Each network block has one self-supervised InfoNCE loss as used in Contrastive Predictive Coding (CPC) (van den Oord et al., 2019), SimCLR (Chen et al., 2020) or CLIP (Radford et al., 2021). In the extreme case of a "single layer per block", this approach yields local self-supervised learning (local-SSL) algorithms, which we see as completely local and BP free. This construction yields currently the best-performing local unsupervised learning algorithms, and in some cases the weight update could also be interpreted as Hebbian rule variants (Illing et al., 2021; Halvagal & Zenke, 2023; Chen et al., 2025). The three major types of

¹Brain Mind Institute, EPFL, Lausanne, Switzerland ²Machine Learning Research Unit, TU Wien, Vienna, Austria. Correspondence to: Wu S. Zihan <zihan.wu@epfl.ch>, Guillaume Bellec <guillaume.bellec@tuwien.ac.at>.

local-SSL algorithms are: contrastive local and predictive plasticity rules (CLAPP) (Illing et al., 2021; Delcrocq et al., 2024), Forward-forward algorithm variants (Hinton, 2022; Momeni et al., 2023; Chen et al., 2025), and non-contrastive methods like LPL (Halvagal & Zenke, 2023). Variations of such local-SSL algorithms enable representation learning in cost-efficient or energy-efficient physical computing systems (Momeni et al., 2023; Weilenmann et al., 2024) and digital event-based hardware (Graf et al., 2024; Su & Indiveri, 2025).

Building upon existing local-SSL algorithms (Illing et al., 2021; Hinton, 2022; Halvagal & Zenke, 2023; Zhu et al., 2025; Chen et al., 2025), we ask two questions: (1) Can we find a theoretical foundation for local-SSL algorithms? (2) Can we reduce the existing performance gap between local-SSL and global BP-SSL? Our main contributions are:

- *Theory*: For deep linear networks with orthonormal weight matrices, we prove that local-SSL implements updates identical to those of global BP-SSL, under the conditions in Theorem 3.1.
- *Theory*: If the orthonormality condition is broken by a shrinking number of neurons for higher layers, we show both theoretically and numerically that adding direct feedback from the last layer makes local-SSL updates more similar to global BP-SSL.
- *Simulation*: In convolutional neural networks, theory indicates that feedback weights should have a structured 2D spatial dependence. Numerically, we show that adding such dependence in local-SSL improves its similarity with global BP-SSL.
- *Performance on standard data sets*: Our theory driven variations of local-SSL (CLAPP++) improve the classification performance on CIFAR-10, STL-10, and Tiny ImageNet (Figure 1D; Table 2). The performance reaches that of a comparable global BP-SSL baseline and sets a new state-of-the-art performance for local-SSL on these datasets.
- *Conceptual*: The generality of our results arises from a novel simplified notation that encompasses multiple recent local-SSL algorithms.

2. State of the Art

Local approximation of supervised BP Alternatives to supervised BP can be classified into two categories. The first category approximates or replaces the feedback network, necessary for the backward pass (Figure 1A), by random or trainable feedback weights (Lillicrap et al., 2016; Akrouf et al., 2019; Kunin et al., 2020), or by direct feedback from

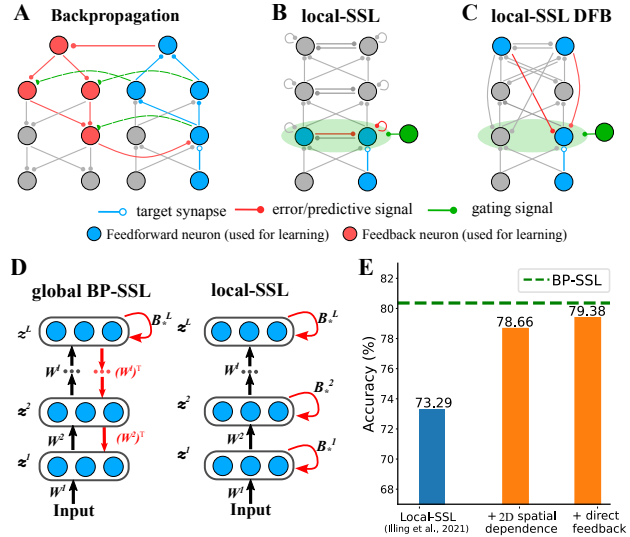


Figure 1. Overview of BP and local-SSL (A) To train the target synapse (blue), BP gradients need to be propagated down a one-to-one matching error network (red) that is gated (green) by the feedforward activations. (B): In Local-SSL, plasticity is modulated by predictive signals from same-layer lateral projections (red arrow) and global scalar values (green). (C) We show that local-SSL with feedback from the top layer better approximates BP. (D) **Notations for theoretical analysis**: In Section 3, we compare gradient updates of local-SSL and global BP-SSL. Both algorithms share the same feedforward weight W^l and projection B_*^l in the last layer. Trainable projections B_*^l are assumed to reach optimum B_*^l instantaneously. (Details in Section 3). (E) **Performance summary**: On STL-10, our theory-guided improvements (orange bars) improve local-SSL algorithms. More details in Section 4.

the top layer (Nøkland, 2016; Bellec et al., 2020; Lounay et al., 2020; Meulemans et al., 2022; Srinivasan et al., 2024). The second category uses bidirectionally connected networks. This dynamical system converges to an equilibrium while the output units are **simultaneously** “free” and then “nudged” or “clamped” to the target value (Lee et al., 2015; Scellier & Bengio, 2017; Sacramento et al., 2018; Whittington & Bogacz, 2017; Ernout et al., 2019; Laborieux et al., 2021; Meulemans et al., 2022; Aceituno et al., 2023; Salvatori et al., 2024; Max et al., 2024). A common finding across both categories has been that a better approximation of BP gradients also yields better performance. Below, we apply this philosophy to the case of SSL.

In principle, some of the BP-approximating methods, like equilibrium propagation or predictive coding, could be adapted for arbitrary loss functions (Scellier et al., 2023; Salvatori et al., 2024), including those of SSL. Yet, in our hands, naive implementations of these theories did not achieve competitive performance on SSL benchmarks (Appendix C and Table 2), so we focus on existing local-SSL rules.

Review of local self-supervised learning (local-SSL)

Self-supervised learning algorithms can be categorized according to their loss functions as either contrastive (e.g., CPC (van den Oord et al., 2019), SimiLR (Chen et al., 2020), and CLIP (Radford et al., 2021)); non-contrastive (e.g., Barlow Twins (Zbontar et al., 2021) and VICReg (Bardes et al., 2022)); or masked input prediction (Bengio et al., 2003; Mikolov et al., 2013). Here we focus on contrastive learning, which is defined as a classification problem between "positive" and "negative" sample pairs, where positive pairs are data augmentations of the same data sample, and negative pairs come from different data samples. In non-contrastive methods, negative pairs are not used, but batched neuron activations are regularized to maintain high variance and decorrelation in order to prevent representational collapse.

The loss of local-SSL is defined per layer and gradients are blocked at every layer (see Halvagal & Zenke (2023); Illing et al. (2021)). For instance, CLAPP (Illing et al., 2021) and LPL (Halvagal & Zenke, 2023) can be interpreted as layerwise contrastive (CPC) and non-contrastive (VICReg) SSL, respectively. Importantly for computational neuroscience, the update steps of both algorithms can be implemented as an extension of Hebbian learning. So CLAPP and LPL demonstrated that generalized Hebbian learning rules can be stacked to generate hierarchical representations in multi-layer convolutional neural networks.

The Forward-forward algorithm (Hinton, 2022) is another example of a local contrastive learning algorithm for deep neural networks. A positive sample is a real image from the dataset. A negative sample is a fake image engineered by mixing two different images; or an image with an erroneous label. The Forward-forward algorithm maximizes the squared norm of the activations for negative samples and minimizes it for positive samples (it reflects the increase of activity in the cortex after surprising events (Homann et al., 2022)). The activations are therefore normalized before transmission to the next layer to avoid having a trivial optimization in subsequent layers. Forward-forward and a variation thereof (Papachristodoulou et al., 2024) work on MNIST, but label information is used for scaling to CIFAR10. Self-Contrastive Forward-forward (SCFF) (Chen et al., 2025) enables full self-supervised training and sets the present state of the art for local-SSL on image datasets (Chen et al., 2025). PhyLL (Momeni et al., 2023) also adapts the local loss of Forward-forward for training physical neural networks.

3. Local-SSL approximates global BP-SSL

In this section, we first formalize the above contrastive local-SSL methods using a common formalism that will facilitate theoretical analysis.

Table 1 summarizes how the common formalism can be applied to CLAPP, Forward-forward, PhyLL, and SCFF. To define notations, we consider the case of a multi-layer perceptron. Given an L -layer network, the activity in each layer l is defined as a real-valued vector $z^l = \rho(a^l)$ where ρ is the non-linear activation function and $a^l = W^l z^{l-1}$ implements a linear transformation of activities from the previous layer by multiplication with weight matrix W^l . The input x of the neural network is given as: $z^0 = x$.

In the context of contrastive algorithms, layerwise activities corresponding to positive and negative input samples are denoted as z_{pos}^l and z_{neg}^l . They are compared to reference vectors c_{pos}^l and c_{neg}^l . (For a subset of algorithms, $c_{\text{pos}}^l = c_{\text{neg}}^l$). The exact definitions of $c_{\text{pos}}^l, c_{\text{neg}}^l$ are different from one algorithm to another (Table 1).

The loss function of all local-SSL algorithms evaluates a scalar score that should be high for z_{pos}^l and low for z_{neg}^l .

The scalar score for z_{pos}^l is defined as $z_{\text{pos}}^{l\top} B^l c_{\text{pos}}^l$ with a matrix B^l ; and analogously for z_{neg}^l . Once the two scores have been evaluated, they are combined into a nonlinear loss function \mathcal{L}^l for layer l that takes one of two forms:

$$\mathcal{L}^l = f\left((z_{\text{pos}}^l - z_{\text{neg}}^l)^\top B^l c_{\text{pos}}^l\right) \quad (\text{type 1})$$

$$\mathcal{L}^l = f\left(z_{\text{pos}}^{l\top} B^l c_{\text{pos}}^l\right) + f\left(-z_{\text{neg}}^{l\top} B^l c_{\text{neg}}^l\right) \quad (\text{type 2})$$

where f is a decreasing function. The matrix B^l is either trainable in CLAPP-like algorithms or fixed to I in Forward-forward algorithms. Note the different locations of the nonlinearity f when combining the positive and negative score values. For both forms, the essential idea is to "push" (in each layer!) the representation of positive and negative inputs into different directions. Table 1 summarizes the definitions of the function f for different algorithms.

3.1. Exact equivalence in deep linear networks

For all layerwise local-SSL algorithms, the absence of gradient flow across layers implies that the weight updates in earlier layers are not explicitly optimizing the SSL objective of later layers. Indeed, optimizing all layers with backprop using an SSL loss function in the *last* layer (global BP-SSL) leads to better performance than local-SSL with the same loss function applied separately in each layer (Illing et al., 2021; Halvagal & Zenke, 2023) (performance is measured as the accuracy of a downstream linear classifier). In this theory section, we examine the mathematical relationship between the gradient update of contrastive local-SSL and the corresponding gradient update of global BP-SSL. We will show for deep linear networks that the layerwise optimization with local-SSL and backpropagation gradient update with global BP-SSL are equal under certain conditions.

We consider a deep linear network and a loss function of type 2, where f is convex with $f' \leq 0$ (the theorems shown

Table 1. Comparison of contrastive local-SSL rules. Choices of f , B^l , c_{neg}^l , and c_{pos}^l for various local-SSL. The reference ξ^l for PhyLL is a fixed vector randomly chosen for each layer, and σ is the sigmoid function. In CLAPP, z^l is the layer activity corresponding to a different augmented copy of the same image that gives z_{pos}^l . The matrix $W^{\text{pred},l}$ is trained to minimize \mathcal{L}^l .

LEARNING RULE	c_{neg}^l	c_{pos}^l	B^l	NORMALIZATION	DECREASING FUNCTION $f(x)$	LOSS TYPE
FORWARD-FORWARD	z_{neg}^l	z_{pos}^l	I	YES	$-x$	2
PHYLL	ξ^l	ξ^l	I	YES	$\text{softplus}(-x)$	1
SCFF	z_{neg}^l	z_{pos}^l	I	YES	$-\log \sigma(x - \theta)$	2
CLAPP	z^l	z^l	$W^{\text{pred},l}$	NO	$\max(0, 1 - x)$	2

below are also valid with losses of type 1, with the same proof technique). For the analytical proofs, we added the L_2 -regularizer using the Frobenius norm $\|\cdot\|_F$:

$$\mathcal{L}^l = f(z_{\text{pos}}^{l\top} B^l c_{\text{pos}}^l) + f(-z_{\text{neg}}^{l\top} B^l c_{\text{neg}}^l) + \lambda \|B^l\|_F^2 \quad (1)$$

The additional regularization enables us to consider a separation of timescales: if B^l is trainable and evolves on a faster timescale than W^l , it is always at the optimal value with respect to the loss. Convexity ensures that the corresponding loss $\mathcal{L}_*^l = \min_{B^l} \mathcal{L}^l$ is reached by a unique B_*^l . Furthermore, we assume orthonormal weight matrices W^l , i.e., square matrices such that, at the moment of gradient update, the incoming weight vectors to all neurons in any given layer l are orthonormal. Then we prove the following relationships between local-SSL and global BP-SSL. Training with global BP-SSL is formalized using the gradient of the loss \mathbf{L} at the last layer: $\mathbf{L} = \mathcal{L}^L$ and $\mathbf{L}_* = \mathcal{L}_*^L$. (To simplify notations, we assume a batch size of one here, and provide the proof for larger batches in Appendix A):

Theorem 3.1. *We assume a deep linear network with orthonormal weight matrices W^l , $l = 1, \dots, L$. Let’s consider \mathcal{L}_*^l with $B_*^l = I$ (e.g. forward-forward) or $B_*^l = \text{argmin}_{B^l} \mathcal{L}^l$ (e.g. CLAPP) and f convex. Then gradients of a layer-wise loss \mathcal{L}_*^l are equal to the gradient of \mathbf{L}_* backpropagated:*

$$(\text{local-SSL}) \quad \frac{\partial \mathcal{L}_*^l}{\partial W_{ij}^l} = \frac{\partial \mathbf{L}_*}{\partial W_{ij}^l} \quad (\text{global BP-SSL}) \quad (2)$$

Proof. Deriving the backpropagated gradient for the last layer loss we have: $\frac{\partial \mathbf{L}_*}{\partial z_{\text{pos}}^L} = f'(z_{\text{pos}}^{L\top} B_*^L c_{\text{pos}}^L) \cdot (W^L \dots W^{l+1})^\top B_*^L c_{\text{pos}}^L$. For the local loss function, we have: $\frac{\partial \mathcal{L}_*^l}{\partial z_{\text{pos}}^l} = f'(z_{\text{pos}}^{l\top} B_*^l c_{\text{pos}}^l) \cdot B_*^l c_{\text{pos}}^l$. The gradients with respect to the parameter are the sum of the outer product of z^{l-1} and these gradients, as well as the opposite contribution of the negative sample. We can prove theorem 3.1 by showing that those two gradients are equal $\frac{\partial \mathbf{L}_*}{\partial z_{\text{pos}}^l} = \frac{\partial \mathcal{L}_*^l}{\partial z_{\text{pos}}^l}$, which is true if we have the equality $B_*^l = (W^L \dots W^{l+1})^\top B_*^L (W^L \dots W^{l+1})$. In the case $B_*^l = I$, we simply observe that the products of matrices are cancelling each other because the matrices are orthonormal. Otherwise, we use the strict convexity of \mathcal{L}^l

with respect to B^l , which implies that B_*^l is uniquely defined. Then using the orthonormality of W^l , it means that the norm of $(W^L \dots W^{l+1})^\top B^L (W^L \dots W^{l+1})$ is the norm of B^L . Thus, we have $\mathbf{L}(B^L) = \mathcal{L}^l((W^L \dots W^{l+1})^\top B^L (W^L \dots W^{l+1}))$. Therefore, if we denote B_*^L the minimum of \mathbf{L} , this equality implies that the minimum of \mathcal{L}^l is reached for $B_*^l = (W^L \dots W^{l+1})^\top B_*^L (W^L \dots W^{l+1})$. \square

Comments: (i) If there are lateral connections $B^l \neq I$ as in CLAPP, the proof requires B^l to be learnable and its search space to be *unconstrained* to include B_*^l . (ii) Orthonormality implies that all layers have an equal number of neurons. (iii) Linear networks are redundant: if an optimum ($\frac{\partial \mathbf{L}_*}{\partial B^L} = 0$) is possible in layer L , the same optimum is also reachable in layer $l < L$ by optimizing B^l . The theorem does not carry over to nonlinear networks.

To verify Theorem 3.1 numerically, we implement a 6-layer linear neural network of 128 neurons per layer with feedforward weights $\{W_l\}_{l=1}^L$ initialized as orthonormal matrices. We compute B_*^l by numerically minimizing the loss \mathcal{L}^l for all layers on a fixed batch of MNIST input samples (Details in Appendix B). Our simulation confirms that the gradient step of local-SSL and global BP-SSL are numerically identical (Figure 2 green line, achieving the maximum cosine similarity of one). To study the importance of the different conditions used in the theorem, we remove one condition at a time: (i) fixed random B^l removes learnability of B^l . (We also show the effect of adding low-rank constraints to B^l in Figure A1); (ii) non-orthogonal W removes orthonormality of weight matrices; (iii) ReLU MLP removes linearity. We also study the case of dropping orthonormality and linearity at the same time. As expected, the gradient alignment drops as more conditions are removed, and the drop is more significant for earlier layers (Figure 2).

Theorem 3.1 is applicable to CLAPP in linear networks, but we have not been able to extend Theorem 3.1 to linear networks that include normalization of layer-activities as required for FF algorithms. Consequently, we focus our subsequent analysis on CLAPP-like (Illing et al., 2021; Delrocq et al., 2024) algorithms, which are normalization-free and feature a learnable matrix B^l . However, our analysis re-

mains valid for any type 1 or type 2 local-SSL loss function, as long as there is a trainable B^l and no normalization.

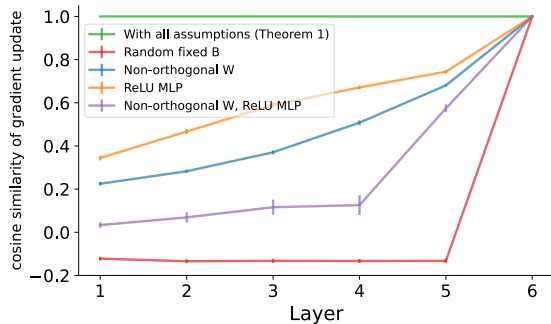


Figure 2. **Numerical verification of Theorem 3.1.** Cosine similarity of the gradient update between BP-SSL training and local-SSL rules across layers (error bar indicating 95% confidence interval computed through different batches of input). Simulations are for theorem 3.1 as well as the cases when conditions are dropped. Random fixed B means that the B^l are not optimized to be B_*^l . ReLU MLP means adding ReLU activation at each layer. Non-orthogonal W means that each W^l is randomly initialized with the default uniform distribution in PyTorch.

3.2. Direct feedback in layerwise loss

One strong implication of the orthonormality in Theorem 3.1 is that the number of neurons has to be constant across layers. When the network has a number of neurons that reduces across layers, we now show that a modified local-SSL algorithm, where the reference is defined using the last layer $c^l = z^{lL}$ (Figure 1, C), approximates global BP-SSL better than the classic local-SSL algorithm, where $c^l = z^l$. Here, CLAPP-like algorithms share the same reference for positive and negative samples: $c_{neg}^l = c_{pos}^l = c^l$. We refer to this algorithm as a local-SSL algorithm with direct feedback (local-SSL DFB) and define it as the gradient update of the loss \mathcal{L}_{fb}^l :

$$\mathcal{L}_{fb}^l = \mathcal{L}^l \quad \text{with } c^l = z^{lL} \quad (3)$$

Since c^l is considered as constant with respect to W^l , the proof of Theorem 3.1 is valid under the same assumptions. So when the layer dimension is still constant across layers, we obtain the same result as previously:

Corollary 3.2. *For a deep linear network with orthonormal weights W^l , gradients of a local-SSL with direct feedback are equal to the BP-SSL gradients of \mathbf{L}_* :*

$$(\text{local-SSL DFB}) \quad \frac{\partial \mathcal{L}_{fb,*}^l}{\partial W_{ij}^l} = \frac{\partial \mathbf{L}_*}{\partial W_{ij}^l} \quad (\text{global BP-SSL}) \quad (4)$$

More interestingly, we now consider a deep network where the number of neurons reduces at each layer. We assume a

deep linear network with semi-orthonormal matrices (meaning that $W^l \in \mathbb{R}^{m^l \times n^l}$, $m^l < n^l$ and W^l has orthonormal row vectors). Then Theorem 3.1 and Corollary 3.2 are not valid anymore. But when f is a linear function, we can prove that local-SSL DFB aligns better with global BP-SSL updates than the standard local-SSL:

Theorem 3.3. *Let’s consider a deep linear network with semi-orthonormal weights W^l and assume that f is a linear function. Then, assuming batch size one, in comparison to the gradient of global BP-SSL, a local-SSL with direct feedback ($c^l = z^{lL}$) is more similar than the standard local-SSL ($c^l = z^l$):*

$$\left\| \frac{\partial \mathcal{L}_*^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} \right\|_F^2 \geq \left\| \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} \right\|_F^2 \quad (5)$$

The proof is written in the Appendix A.

As previously, we verified the Theorem 3.3 with numerical simulations. We implement a deep linear network where the dimensionality decreases by half in each layer (from 128 to 4). As shown in Figure 3 (A), when f is linear, the gradient alignment between local-SSL and BP-SSL gradients increases with the introduction of direct feedback. Although the proof only addresses the sum of squared difference for batch size one, the simulation extends to the cosine similarity of the gradient updates for larger batches. We also observe the same trend when f is a softplus function, which is no longer linear but is still a decreasing convex function.

To measure the similarity with the BP-SSL gradients in a more realistic setting, we drop all conditions considered before: we train a 6-layer ReLU MLP with standard initialization for W^l on the MNIST dataset and use batched inputs. Additionally, B^l is optimized by gradient descent of local objective \mathcal{L}^l at each iteration, so it does not reach the optimum B_*^l for each input. Then, we evaluate the cosine similarities with BP-SSL updates on held-out data (Details in Appendix B). Even in this more realistic setting, local-SSL DFB demonstrates a better approximation to BP updates (orange line in Figure 3, B). As a control, we also train local-SSL with randomly initialized and fixed B^l to confirm that training B^l is important for approximating BP updates (grey line in Figure 3, B).

Local-SSL DFB is close to optimal in ReLU MLP Since it is impossible to exactly match BP with a single feedback matrix B^l in deep non-linear networks, we also define a theoretical upper bound for BP approximation, which is given by an “optimal local-SSL” gradient Δ_\circ . Δ_\circ is defined as a weight update that has the same form as a local-SSL gradient, but the feedback weights are explicitly optimized to match the BP-SSL gradients by definition 3.4. Although it is possible to make measurements and evaluate Δ_\circ for

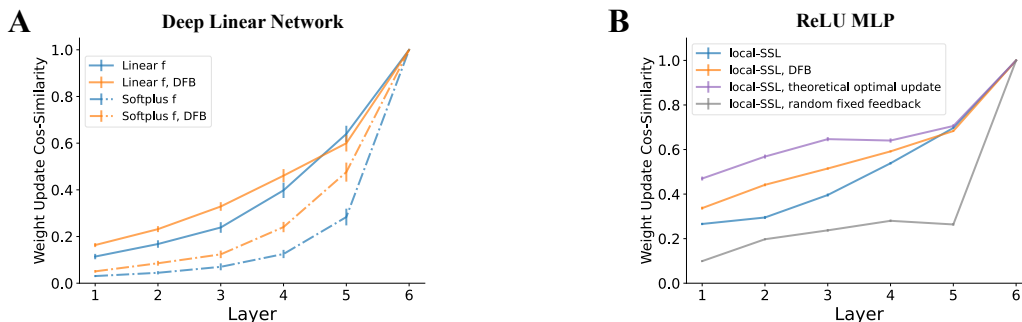


Figure 3. **Numerical verification that direct feedback improves backpropagation approximation.** Cosine similarity of the gradient update between local-SSL and BP-SSL across layers. **A:** Simulation for theorem 3.3 on a 6-layer deep linear network with shrinking widths, both for linear $f(x) = -x$ and non-linear (softplus) $f(x) = \log(1 + \exp(-x))$. **B:** Comparison of local-SSL and theoretical optimum of BP approximation. Simulations are conducted on a 6-layer ReLU MLP trained with MNIST input. Gray: top-down feedback projection B^l is fixed with random initialization. Purple: train top-down feedback explicitly with definition 3.4.

our analysis, its computation requires BP in practice, so it cannot be local. We only use this mathematical construction to evaluate an upper bound of the cosine similarity that is achievable between BP-SSL and local-SSL gradients.

Definition 3.4. We define the *theoretical optimum of local-SSL* update Δ_{\circ} as the weight update $\Delta_{\circ} = \frac{\partial \mathcal{L}_{\text{BP}}^l}{\partial W^l}(B_{\circ}^l)$ where B_{\circ}^l is **not** trained to minimize \mathcal{L}^l but to minimize directly the mean squared error with the gradient of global BP-SSL $\left\| \frac{\partial \mathcal{L}_{\text{BP}}^l}{\partial z^l}(B_{\circ}^l) - \frac{\partial \mathbf{L}_{*}}{\partial z^l} \right\|_F^2$ averaged over the dataset.

To perform a numerical comparison between local-SSL and this theoretical upper bound, we train the ReLU MLP with the ideal Δ_{\circ} update and then compute the cosine similarity between the local update (local-SSL update with DFB or Δ_{\circ}) and the BP gradients on held-out data (Details in Appendix B). The narrow margin between local-SSL DFB (orange) and the upper bound (purple) suggests that our local-SSL is closing the gap with this optimal local algorithm. It is non-trivial that local-SSL algorithms would approach close to this upper bound because B^l minimizes \mathcal{L}^l and not directly the similarity with the BP-SSL gradient.

3.3. Spatially dependent local-SSL for convnets

Beyond ReLU MLP, we intend to design a local-SSL algorithm that is competitive with CLAPP and SCFF with convnet architectures. We follow the same approach and seek algorithmic modifications that would improve the gradient approximation of BP-SSL in the convnet architecture. We explain below how we found that adding spatial structure to the loss \mathcal{L}^l was important in our case.

In convnets, z^l are 2D feature maps and the feedforward matrix product $W^l z^{l-1}$ is replaced with a 2D convolution $W^l * z^{l-1}$. To compute the value $z^{l\top} B^l c^l$ in local objectives \mathcal{L}^l , Illing et al. (2021) first average the feature z^l, c^l over all

spatial locations. However, it means that B^l is independent of the 2D spatial locations of the feature map z^l and c^l , leading to a constrained search space of B^l .

We construct instead a linear operator B^l that reflects the structured 2D spatial dependence of the BP-SSL gradient. BP errors from the last layer are propagated through the successive backward operators of the convolution $W^l * z^{l-1}$ and non-linearities. This results in a BP-SSL gradient that is spatially dependent, with different gradients for different spatial locations of the feature maps. Following this intuition, we use different B^l depending on the spatial locations of z^l and c^l in each feature map (Figure 4, A), and nearby features share the same B^l to reduce the memory cost. On a 4-layer linear convnet, we simulate the effect of using spatially independent and spatially dependent projection B^l (Details in Appendix B). We found that introducing spatial dependence improves the alignment with BP-SSL gradients (Figure 4, B). Using spatially dependent feedback from the last layer further improves the gradient alignment.

4. Empirical results

Although our analysis shows that local-SSL is almost the ideal approximation of BP-SSL update, the gap in cosine similarity in early layers may appear substantial. The decisive test is to measure whether local-SSL matches the functional performance of global BP-SSL pre-training. We train the same convolutional networks as Illing et al. (2021). The network is first pre-trained using different learning rules. To measure the quality of the learned representation, we freeze the network and report the accuracy of a supervised linear classifier (detail in Appendix B).

Before studying the changes of the algorithm guided by our theory, we build a baseline upon the original CLAPP (Illing et al., 2021). We made two notable modifications from the

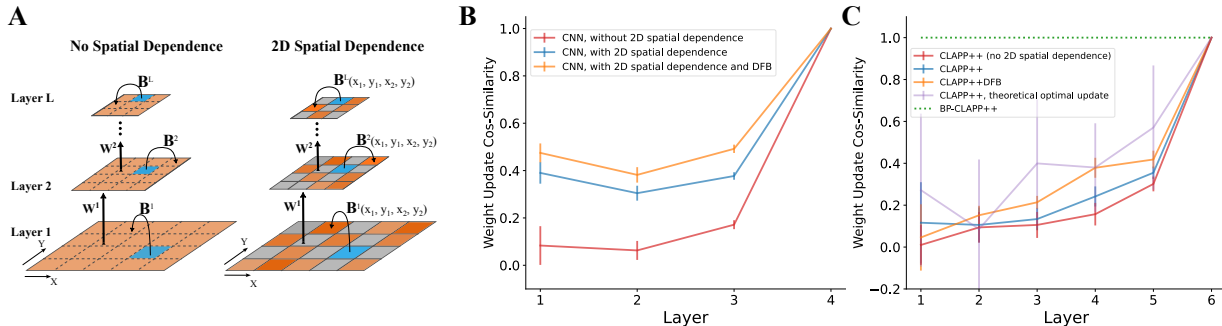


Figure 4. Numerical verification that spatial dependence improves BP approximation. (A): Illustration of spatial dependence: without spatial dependence (left), the same B^l is used to project c^l (blue) onto neurons across the feature map, so they share the same color; with spatial dependence (right), lateral projections B^l depend on the locations (x_1, y_1) and (x_2, y_2) , of z^l and c^l in their layer’s feature map. Only nearby neurons with the same color share the same projection B^l . (B) and (C): Cosine similarity of the gradient update between BP-SSL and local-SSL across layers. For B, values are obtained from 4-layer linear convnets with optimal B^l_* . Kernel and stride have length 2 for simplicity. For C, values are obtained from VGG models trained on STL10. Purple: theoretical optimal update Δ_o obtained from training following the definition 3.4. (Details in Appendix B). The plotted measurements are performed after 100 epochs of training.

original CLAPP. First, we used the data augmentations in SimCLR (Chen et al., 2020) rather than non-overlapping image patches to create contrastive samples. Second, after self-supervised training, we read-out concatenated representations from multiple layers to train the linear classifier.

Spatial dependence Then, we add spatial dependence to the projection B^l , based on the analysis in the previous section. We call this new learning rule as CLAPP++, as it uses the same objective function as CLAPP. Consistent with our theory and simulation in linear networks, we find that the usage of spatial dependence makes the gradient of the local-SSL more similar to the BP-SSL learning algorithm during training of non-linear convnets (Figure 4, C). Ablation studies of CLAPP++ demonstrate that the most significant performance improvements on CIFAR10, STL10, and TinyImageNet originate from the introduction of spatial dependence (Table 2, Figure 5).

Direct feedback Next, we introduce direct top-down feedback and train a model CLAPP++DFB, which is implemented by defining reference input from the final layer $c^l = z^L$. Similar to our results on ReLU MLP, the alignment between CLAPP++DFB gradient and BP gradient further increases, and the gradient of CLAPP++DFB is also close to the ‘optimal’ gradient approximation obtained by explicitly training the feedback with the update in definition 3.4 (Figure 4, purple line). The improvements in gradient alignment is also consistent throughout different stages of training (Figure A2). Across the three datasets, CLAPP++DFB appears to provide a small but consistent improvement of performance (Figure 5, Table 2). We also train a model CLAPP++both, in which we optimize the sum of two losses $\mathcal{L}^l_{fb} + \mathcal{L}^l$ at each layer. This model achieves

the best performance across all datasets (Table 2).

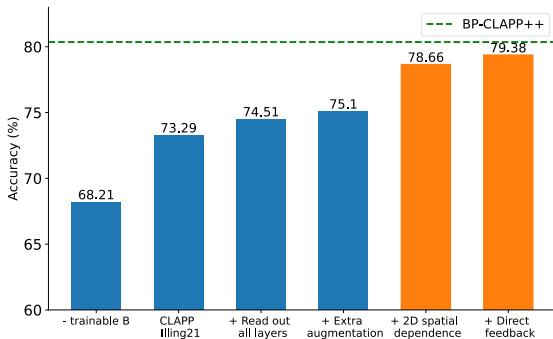


Figure 5. Comparison between CLAPP (Illing et al., 2021) and CLAPP++. Orange bars are theory guided algorithmic changes.

Local-SSL SOTA As summarized in Table 2, we compared our results to the layer-wise SSL algorithms that we could find in the literature. Previously, the best performing local-SSL algorithm was Self-Contrastive Forward Forward (SCFF). All our CLAPP++ variants with theoretically guided spatial dependence reach a higher accuracy on all three datasets. So it appears that CLAPP++ is therefore the current state-of-the-art of local-SSL algorithms.

Local-SSL reaches the performance of BP-SSL. Reaching the performance of backpropagation has been admirably hard with local supervised learning rules (Akrouf et al., 2019; Launay et al., 2020), and it was also never achieved in SSL setups. For our CLAPP++ variants with spatial dependence (CLAPP++, CLAPP++DFB, CLAPP++both), we find that local-SSL brings downstream classification accuracies close to the performance of BP-SSL. The most comparable

Table 2. Local-SSL achieves the performance of SSL-BP. Different local-SSL rules tested on various image classification benchmarks. The BP-SSL references are named BP-CLAPP++ when the same loss and network of CLAPP++ is used, but the gradients are propagated end-to-end by BP. In BP-InfoNCE, we replaced the CLAPP++ loss with the InfoNCE loss (van den Oord et al., 2019; Chen et al., 2020). Naive CLAPP DFA/Predictive Coding are the performance of using DFA (Nøkland, 2016) and Predictive Coding (Salvatori et al., 2024) to train the last layer objective L of CLAPP. When the citation is not provided, the results are produced from our own simulations on the same VGG convnet setup. Values after \pm are the 95% confidence interval computed from 4 simulations of different random seeds.

Method	Local update	CIFAR10 accuracy	STL-10 accuracy	Tiny-ImageNet accuracy
BP-CLAPP++	no	80.49 \pm 0.20	80.36 \pm 0.26	37.55 \pm 0.61
BP-InfoNCE	no	80.69 \pm 0.84	81.97 \pm 1.23	36.78 \pm 0.73
Naive CLAPP DFA	yes	-	52.30	-
Naive CLAPP Predictive Coding	yes	-	36.75	-
CLAPP (Illing et al., 2021)	yes	-	73.6	-
LPL (Halvagal & Zenke, 2023)	yes	59.4	63.2	-
Stochastic FF (Zhu et al., 2025)	yes	76.96	-	-
SCFF (Chen et al., 2025)	yes	80.60	77.14	35.67
CLAPP++ (no 2D spatial dependence)	yes	73.21	75.10	28.18
CLAPP++	yes	80.51 \pm 0.27	78.66 \pm 0.16	36.63 \pm 0.32
CLAPP++DFB	yes	80.65 \pm 0.18	79.38 \pm 0.17	36.70 \pm 0.24
CLAPP++both	yes	81.18 \pm 0.18	79.62 \pm 0.28	37.78 \pm 0.67

BP-SSL baseline is BP-CLAPP++, for which we optimized the CLAPP++ loss function of the last layer end-to-end with BP (BP-CLAPP++ in Figure 5, Table 2). To further show that this achievement is not trivially caused by the limitation of the SSL loss used by CLAPP++, we also train the InfoNCE loss, used in CPC and SimCLR (van den Oord et al., 2019; Chen et al., 2020). As shown in Table 2, the performance is also similar to our local CLAPP++ algorithm. Note that our BP-InfoNCE baseline is not however the state-of-the-art of global BP-SSL. It uses linear projection heads and the same VGG architecture as CLAPP++. Better performance is expected with non-linear projection heads and ResNet architecture (Chen et al., 2020).

5. Discussion

The central theoretical contribution is to show that some local-SSL algorithms can implement exactly BP-SSL in deep linear networks. This theoretical finding has guided our improvements of local-SSL algorithms to increase the similarity with BP-SSL and the algorithm performance.

So far, we have not exploited the resource efficiency of local-SSL. Besides the potential with custom hardware (Su & Indiveri, 2025; Momeni et al., 2023), CLAPP++DFB has a straightforward advantage over BP and CLAPP: instead of storing the activation or the context vector c^l at every layer. CLAPP++DFB requires a single context vector z^L for the whole network which reduces the memory footprint. We leave a deeper analysis to future work.

Biological plausibility of CLAPP++ Local learning rules like CLAPP (Illing et al., 2021) and LPL (Halvagal &

Zenke, 2023) were intended to model plasticity in the brain. For CLAPP, the bio-plausibility argument comes from the Hebbian-like weight update. For the matrix B^l the update is a classical hebbian rule $\Delta B_{jk}^l = \gamma^t \cdot z_j^l \cdot c_k^l$ where $\gamma^t \in \{-1, 0, 1\}$ is reminiscent of the effect of neuromodulators. The update of the feedforward weight W^l has a more important role with CLAPP: the plasticity is additionally gated by the contextual activity c^l and can be interpreted as the apical dendritic input that is thought to gate plasticity (Larkum et al., 1999; Bittner et al., 2017; Illing et al., 2021):

$$\Delta W_{ji}^l = \underbrace{\gamma^t}_{\text{neuromodulator}} \cdot \underbrace{(B^l c^l)_j}_{\text{dendritic prediction}} \cdot \underbrace{\rho'(a_j^l) z_i^{l-1}}_{\text{hebbian term}}. \quad (6)$$

In CLAPP++, this interpretation is not compromised, and it even elucidates a question that remained in CLAPP. Apical dendrites in the cortex are thought to integrate inputs from distant higher brain areas (Harris & Shepherd, 2015; Gilbert & Li, 2013; Marques et al., 2018; Keller et al., 2020; Nejad et al., 2025). Recent *in vitro* studies confirmed that apical dendrites modulate the synaptic plasticity in basal dendrites (Aceituno et al., 2024). As illustrated in Figure 1 C, this is better captured with CLAPP++DFB, where the learning signal comes from the top layer while the dendritic prediction $B^l c^l$ comes from nearby neurons in CLAPP. Similarly, our 2D spatial dependence model may explain some of the structured connectivity visible across the visual cortex (Harris et al., 2019).

More generally, our study shows that the gradient of global SSL principles can be well approximated by local biologically plausible learning rules. Thereby, it lifts the requirement for the contested machinery of backpropagation to implement efficient gradient based representation learning.

Impact Statement

This paper presents work whose goal is to advance the field of neuroscience and machine learning research. There are no potential societal consequences of our work that we feel must be specifically highlighted here.

Acknowledgement

This research is supported by the Swiss National Science Foundation grants (200020-207426 and 200021-236436) and Vienna Science and Technology Fund (VRG24-018). We thank Tommaso Salvatori, Zhe Su, and everyone from the laboratory of computational neuroscience (LCN) at EPFL for their support and discussion. We also especially want to thank Xing Chen and Bernd Illing for their GitHub code on previous work and for providing deep and insightful feedback on the paper.

References

- Aceituno, P. V., Farinha, M. T., Loidl, R., and Grewe, B. F. Learning cortical hierarchies with temporal hebbian updates. *Frontiers in Computational Neuroscience*, 17: 1136010, 2023.
- Aceituno, P. V., de Haan, S., Loidl, R., and Grewe, B. F. Target learning rather than backpropagation explains learning in the mammalian neocortex. *bioRxiv*, 2024. doi: 10.1101/2024.04.10.588837.
- Akroud, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.
- Bardes, A., Ponce, J., and Lecun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR 2022-International Conference on Learning Representations*, 2022.
- Barlow, H. B. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.
- Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.
- Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.
- Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.
- Bi, G. and Poo, M. Synaptic modification of correlated activity: Hebb’s postulate revisited. *Ann. Rev. Neurosci.*, 24:139–166, 2001.
- Bittner, K. C., Milstein, A. D., Grienberger, C., Romani, S., and Magee, J. C. Behavioral time scale synaptic plasticity underlies ca1 place fields. *Science*, 357(6355): 1033–1036, 2017.
- Bredenberg, C., Lyo, B., Simoncelli, E., and Savin, C. Impression learning: Online representation learning with synaptic plasticity. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11717–11729. Curran Associates, Inc., 2021.
- Bredenberg, C., Williams, E., Savin, C., Richards, B., and Lajoie, G. Formalizing locality for normative synaptic plasticity models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 5653–5684. Curran Associates, Inc., 2023.
- Caporale, N. and Dan, Y. Spike timing-dependent plasticity: A hebbian learning rule. *Ann. Rev. Neurosci.*, 31:25–46, 2008.
- Castellucci, V., Pinsker, H., Kupfermann, I., and Kandel, E. R. Neuronal mechanisms of habituation and dishabituation of the gill-withdrawal reflex in aplysia. *Science*, 167(3926):1745–1748, 1970.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.
- Chen, X., Liu, D., Laydevant, J., and Grollier, J. Self-contrastive forward-forward algorithm. *Nature Communications*, 16(1):5978, 2025.
- Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.
- Crick, F. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.
- Delrocq, A., Wu, Z., Bellec, G., and Gerstner, W. Critical periods support representation learning in a model of cortical processing. *bioRxiv*, 2024. doi: 10.1101/2024.12.20.629674.
- Ernault, M., Grollier, J., Querlioz, D., Bengio, Y., and Scellier, B. Updates of equilibrium prop match gradients of backprop through time in an rnn with static input.

- Advances in neural information processing systems*, 32, 2019.
- Ernault, M. M., Normandin, F., Moudgil, A., Spinney, S., Belilovsky, E., Rish, I., Richards, B., and Bengio, Y. Towards scaling difference target propagation by learning backprop targets. In *International Conference on Machine Learning*, pp. 5968–5987. PMLR, 2022.
- Földiák, P. Learning invariance from transformation sequences. *Neural computation*, 3(2):194–200, 1991.
- Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits*, 12:53, 2018.
- Gilbert, C. D. and Li, W. Top-down influences on visual processing. *Nature reviews neuroscience*, 14(5):350–363, 2013.
- Graf, L., Su, Z., and Indiveri, G. Echospike predictive plasticity: An online local learning rule for spiking neural networks. *arXiv preprint arXiv:2405.13976*, 2024.
- Halvagal, M. S. and Zenke, F. The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks. *Nature Neuroscience*, pp. 1–10, 2023.
- Harris, J. A., Mihalas, S., Hirokawa, K. E., Whitesell, J. D., Choi, H., Bernard, A., Bohn, P., Caldejon, S., Casal, L., Cho, A., et al. Hierarchical organization of cortical and thalamic connectivity. *Nature*, 575(7781):195–202, 2019.
- Harris, K. D. and Shepherd, G. M. The neocortical circuit: themes and variations. *Nature neuroscience*, 18(2):170–181, 2015.
- He, K., Huertas, M., Hong, S. Z., and Johannes W Hell, X. T., Shouval, H., and Kirkwood, A. Distinct eligibility traces for ltp and ltd in cortical synapses. *Neuron*, 88: 528–538, 2015.
- Hebb, D. O. The organization of behavior. *New York*, 1949.
- Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2(3):5, 2022.
- Homann, J., Koay, S. A., Chen, K. S., Tank, D. W., and Berry, M. J. Novel stimuli evoke excess activity in the mouse primary visual cortex. *Proceedings of the National Academy of Sciences*, 119(5):e2108882119, 2022.
- Hyvarinen, A. and Oja, E. Independent component analysis by general nonlinear hebbian-like learning rules. *Signal Processing*, 64:301–313, 1998.
- Illing, B., Ventura, J., Bellec, G., and Gerstner, W. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. *Advances in Neural Information Processing Systems*, 34:30365–30379, 2021.
- Kappel, D., Nazeer, K. K., Fokam, C. T., Mayr, C., and Subramoney, A. Block-local learning with probabilistic latent representations. *arXiv preprint arXiv:2305.14974*, 2023.
- Keller, A. J., Roth, M. M., and Scanziani, M. Feedback generates a second receptive field in neurons of the visual cortex. *Nature*, 582(7813):545–549, 2020.
- Kunin, D., Nayebi, A., Sagastuy-Brena, J., Ganguli, S., Bloom, J., and Yamins, D. Two routes to scalable credit assignment without weight symmetry. In *International Conference on Machine Learning*, pp. 5511–5521. PMLR, 2020.
- Laborieux, A., Ernault, M., Scellier, B., Bengio, Y., Grollier, J., and Querlioz, D. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:633674, 2021.
- Larkum, M. E., Zhu, J. J., and Sakmann, B. A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341, 1999.
- Launay, J., Poli, I., Boniface, F., and Krzakala, F. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.
- Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015.
- Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.
- Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.
- Lisman, J., Grace, A., and Duzel, E. A neoHebbian framework for episodic memory; role of dopamine-dependent late ltp. *Trends Neurosci.*, 34:536–547, 2011.
- Lomo, T. Frequency potentiation of excitatory synaptic activity in dentate area of hippocampal formation. In *Acta Physiologica Scandinavica*, pp. 128. Blackwell Science Ltd PO BOX 88, OSNEY MEAD, OXFORD OX2 ONE, OXON, ENGLAND, 1966.

- Löwe, S., O’Connor, P., and Veeling, B. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.
- Markram, H., Gerstner, W., and Sjöström, P. J. A history of spike-timing-dependent plasticity. *Frontiers in synaptic neuroscience*, 3:4, 2011.
- Marques, T., Nguyen, J., Fioreze, G., and Petreanu, L. The functional organization of cortical feedback inputs to primary visual cortex. *Nature neuroscience*, 21(5):757–764, 2018.
- Max, K., Kriener, L., Pineda García, G., Nowotny, T., Jaras, I., Senn, W., and Petrovici, M. A. Learning efficient backprojections across cortical hierarchies in real time. *Nature Machine Intelligence*, pp. 1–12, 2024.
- Meulemans, A., Farinha, M. T., Cervera, M. R., Sacramento, J., and Grewe, B. F. Minimizing control for credit assignment with strong feedback. In *International Conference on Machine Learning*, pp. 15458–15483. PMLR, 2022.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- Miller, K., Keller, J. B., and Stryker, M. P. Ocular dominance column development: analysis and simulation. *Science*, 245:605–615, 1989.
- Momeni, A., Rahmani, B., Malléjac, M., Del Hougne, P., and Fleury, R. Backpropagation-free training of deep physical neural networks. *Science*, 382(6676):1297–1303, 2023.
- Nejad, K. K., Anastasiades, P., Hertäg, L., and Costa, R. P. Self-supervised predictive learning accounts for cortical layer-specificity. *Nature Communications*, 16(1):6178, 2025.
- Nøkland, A. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.
- Oja, E. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.
- Papachristodoulou, A., Kyrkou, C., Timotheou, S., and Theocharides, T. Convolutional channel-wise competitive learning for the forward-forward algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14536–14544, 2024.
- Pinchetti, L., Qi, C., Lokshyn, O., Emde, C., M’Charrak, A., Tang, M., Frieder, S., Menzat, B., Oliviers, G., Bogacz, R., Lukasiewicz, T., and Salvatori, T. Benchmarking predictive coding networks – made simple. In *Proceedings of the 13th International Conference on Learning Representations, ICLR 2025, Singapore, 24–28 April 2025*, 2025.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.
- Rao, R. P. and Ballard, D. H. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.
- Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.
- Roelfsema, P. and van Ooyen, A. Attention-gated reinforcement learning of internal representations for classification. *Neural Computation*, 17:2176–2214, 2005.
- Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8721–8732. Curran Associates, Inc., 2018.
- Salvatori, T., Song, Y., Yordanov, Y., Millidge, B., Xu, Z., Sha, L., Emde, C., Bogacz, R., and Lukasiewicz, T. A stable, fast, and fully automatic learning algorithm for predictive coding networks. In *Proceedings of the 12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, 7–11 May 2024*, 2024.
- Scellier, B. and Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.
- Scellier, B., Ernoult, M., Kendall, J., and Kumar, S. Energy-based learning algorithms for analog computing: a comparative study. *Advances in Neural Information Processing Systems*, 36:52705–52731, 2023.
- Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., Kar, K., Bashivan, P., Prescott-Roy, J., Schmidt, K., Yamins, D. L. K., and DiCarlo, J. J. Brain-score: Which artificial neural network for object recognition is most brain-like? *bioRxiv*, 2018. doi:10.1101/407007.

- Schultz, W., Dayan, P., and Montague, P. R. A neural substrate of prediction and reward. *Science*, 275(5306): 1593–1599, 1997.
- Shatz, C. The developing brain. *Sci. Am.*, 267:60–67, 1992.
- Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., and Bogacz, R. Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nat. Neurosci.*, 27:348–358, 2024.
- Srinivasan, R. F., Mignacco, F., Sorbaro, M., Refinetti, M., Cooper, A., Kreiman, G., and Dellaferrera, G. Forward learning with top-down feedback: Empirical and analytical characterization. In *ICLR*, 2024.
- Su, Z. and Indiveri, G. Elfcore: A 28nm neural processor enabling dynamic structured sparse training and online self-supervised learning with activity-dependent weight update. In *2025 IEEE European Solid-State Electronics Research Conference (ESSERC)*, pp. 13–16. IEEE, 2025.
- van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding, 2019. URL <https://arxiv.org/abs/1807.03748>.
- Weilenmann, C., Ziogas, A. N., Zellweger, T., Portner, K., Mladenović, M., Kaniselvan, M., Moraitis, T., Luisier, M., and Emboras, A. Single neuromorphic memristor closely emulates multiple synaptic mechanisms for energy efficient neural networks. *Nature Communications*, 15(1):6898, 2024.
- Whittington, J. C. and Bogacz, R. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.
- Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.
- Wiskott, L. and Sejnowski, T. J. Slow feature analysis: Un-supervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.
- Yagishita, S., Hayashi-Takagi, A., Ellis-Davies, G. C., Urakubo, H., Ishii, S., and Kasai, H. A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620, 2014.
- Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.
- Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.
- Zhu, Z., QI, Y., Ma, H., Lu, W., and Feng, J. Stochastic forward-forward learning through representational dimensionality compression. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A. Proof

A.1. Proof of batched theorem 3.1 with batched loss function

Consider the loss of type 2 with batched input \mathcal{B} and L-2 regularization:

$$\mathcal{L}^l = \sum_{\mu \in \mathcal{B}} f\left(-z_{\text{neg}}^{\mu,l \top} B^l c^{\mu,l}\right) + f\left(z_{\text{pos}}^{\mu,l \top} B^l c^{\mu,l}\right) + \lambda \|B^l\|_F^2,$$

We show as in the Theorem 3.1 that $\frac{\partial \mathcal{L}^l}{\partial W_{ij}^l} = \frac{\partial \mathcal{L}^l}{\partial W_{ij}^l}$ but now each loss has a batch dimension. The core of the proof is identical to the batch one version described in the main text because we assume that there is no interdependencies across batch dimensions.

Proof. Deriving the back propagated gradient for the last layer loss we have: $\frac{\partial \mathbf{L}_*}{\partial z_{\text{pos}}^{\mu,l}} = f'((z_{\text{pos}}^{\mu,l})^\top B_*^L c^{\mu,L}) \cdot (W^L \dots W^{l+1})^\top B_*^L c^{\mu,L}$. For the local loss function we have: $\frac{\partial \mathcal{L}_*^l}{\partial z_{\text{pos}}^{\mu,l}} = f'((z_{\text{pos}}^{\mu,l})^\top B_*^l c^{\mu,l}) \cdot B_*^l c^{\mu,l}$. The gradients with respect to the parameter are the sum of the outer product of z^{l-1} and these gradients scaled by local derivatives, and the opposite contribution of the negative sample. Because $\frac{\partial \mathbf{L}_*}{\partial W_{ij}^l} = \sum_{\mu} \left(\frac{\partial \mathcal{L}_*^l}{\partial z_{\text{pos}}^{\mu,l}} \frac{\partial z_{\text{pos}}^{\mu,l}}{\partial W_{ij}^l} + \frac{\partial \mathcal{L}_*^l}{\partial z_{\text{neg}}^{\mu,l}} \frac{\partial z_{\text{neg}}^{\mu,l}}{\partial W_{ij}^l} \right)$, the difference between local-SSL gradients and BP gradients would only appears in the partial derivative of loss \mathbf{L}_* with respect to layer activations. Therefore, we can prove batched theorem 3.1 by showing that those two gradients are equal $\frac{\partial \mathbf{L}_*}{\partial z_{\text{pos}}^{\mu,l}} = \frac{\partial \mathcal{L}_*^l}{\partial z_{\text{pos}}^{\mu,l}}$, which is true if we have the equality $B_*^l = (W^L \dots W^{l+1})^\top B_*^L (W^L \dots W^{l+1})$. We could repeat the same method as the one in the proof for batch 1. In the case $B_*^l = I$, we simply observe that products of matrices are cancelling each other because the matrices are orthonormal. Otherwise, we use the strict convexity of \mathcal{L}^l with respect to B^l , which implies that B_*^l is uniquely defined. Then using the orthonormality of W^l , it means that the norm of $(W^L \dots W^{l+1})^\top B^L (W^L \dots W^{l+1})$ is the norm of B^L , therefore we have $\mathcal{L}^l(B^L) = \mathcal{L}^l((W^L \dots W^{l+1})^\top B^L (W^L \dots W^{l+1}))$. Therefore if we denote B_*^L the minimum of \mathcal{L}^L , this equality implied that the minimum of \mathcal{L}^l is reached for $B_*^l = (W^L \dots W^{l+1})^\top B_*^L (W^L \dots W^{l+1})$. \square

A.2. Proof of theorem 3.3

Our goal is to show that the local-SSL gradients are closer to the BP-SSL gradient when we use the direct feedback. To avoid ambiguity in the context of the following proof we use the matrix B^l to refer to matrix of the local-SSL $\mathcal{L}^l(B^l)$ and D^l to refer to the local-SSL DFB $\mathcal{L}^l(D^l)$. For all layers $l < L$, D^l is typically a rectangular matrix while B^l is a square matrix. Given our notation the loss at the top layer is identical in both, and it is equal to the loss used with BP-SSL, so we have by definition: $\mathbf{L}_* = \mathcal{L}_*^L = \mathcal{L}_{*,\text{fb}}^L$. And we seek to show that:

$$\left\| \frac{\partial \mathcal{L}_*^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} \right\|_F^2 \geq \left\| \frac{\partial \mathcal{L}_{*,\text{fb}}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} \right\|_F^2 \quad (7)$$

Proof. We denote $R^l = W^L W^{L-1} \dots W^{l+1}$. The dimensionality of each matrix W^l is $n^{l+1} \times n^l$ and $n^{l+1} < n^l$. The reduced dimensionality forces the last layer activity to compute scores in a low-dimensional space. We show that the null-space of R^l leads to difference between layerwise and end-to-end gradient.

We do singular value decomposition of $R^l = USV^T$. Here S is a n^L -by- n^{l+1} matrix with first n^L dimensions being identity matrix I_{n^L} and zeros elsewhere. Then, we could rewrite it also as $US_l V_l^T$ with S_l being I_{n^L} and V_l being the first n^L columns of V . So, we have $R^l = UV_l^T$. In addition, we denote V_\emptyset^T the remaining columns of V .

We do a change of variable $B^l = V \hat{B} V^T$ and examine the layerwise loss with a single input: (for simplicity, we denote $\Delta z^l = z_{\text{neg}}^l - z_{\text{pos}}^l$)

$$\mathcal{L}_*^l = \min_{\hat{B}} (\Delta z^l)^T V \hat{B} V^T c^l + \lambda \|\hat{B}\|_F^2$$

We could rewrite V as $[V_l \ V_\emptyset]$ and \hat{B} as $\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$ with B_1, B_2 having n^L rows and B_1, B_3 having n^L columns. Plug it

into the loss:

$$\begin{aligned}\mathcal{L}_*^l &= \min_{B_1..B_4} (\Delta z^l)^T [V_l \ V_\emptyset] \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \begin{bmatrix} V_l^T \\ V_\emptyset^T \end{bmatrix} c^l + \lambda \sum_{j=1}^4 \|B_j\|_F^2 \\ &= \min_{B_1..B_4} (\Delta z^l)^T (V_l B_1 V_l^T + V_l B_2 V_\emptyset^T + V_\emptyset B_3 V_l^T + V_\emptyset B_4 V_\emptyset^T) c^l + \lambda \sum_{j=1}^4 \|B_j\|_F^2\end{aligned}$$

If we isolate all the terms depending on B_1 , we show that this part is equivalent to BP loss \mathbf{L} . By using a change of variable $\hat{W} = U B_1 U^T$:

$$\begin{aligned}\mathcal{L}_{*,B_1}^l &= \min_{B_1} \sum_{\mu} (\Delta z^l)^T V_l B_1 V_l^T c^l + \lambda \|B_1\|_F^2 \\ &= \min_{\hat{W}} (\Delta z^l)^T V_l U^T \hat{W} U V_l^T c^l + \lambda \|\hat{W}\|_F^2 \\ &= \min_{\hat{W}} (\Delta z^l)^T R_l^T \hat{W} R_l c^l + \lambda \|\hat{W}\|_F^2 \\ &= \min_{\hat{W}} (\Delta z^L)^T \hat{W} c^L + \lambda \|\hat{W}\|_F^2 = \mathcal{L}_*^L = \mathbf{L}_*\end{aligned}$$

Therefore, the difference between \mathcal{L}_*^l and \mathbf{L}_* are the B_2, B_3, B_4 terms:

$$\mathcal{L}_*^l - \mathbf{L}_* = \min_{B_2..B_4} (\Delta z^l)^T (V_l B_2 V_\emptyset^T + V_\emptyset B_3 V_l^T + V_\emptyset B_4 V_\emptyset^T) c^l + \lambda \sum_{j=2}^4 \|B_j\|_F^2$$

Similarly, we apply the same methods to identify the difference between $\mathcal{L}_{*,fb}^l$ and \mathbf{L}_* . To better differentiate the B^l in \mathcal{L}^l and \mathcal{L}_{fb}^l , we use D^l to denote the rectangular matrix in \mathcal{L}_{fb}^l . The definition for $\mathcal{L}_{*,fb}^l$ is then:

$$\mathcal{L}_{*,fb}^l = \min_{\hat{D}} (\Delta z^l)^T D^l c^L + \lambda \|D^l\|_F^2$$

We again perform the change of variable $D^l = V \hat{D}$:

$$\begin{aligned}\mathcal{L}_{*,fb}^l &= \min_{\hat{D}} (\Delta z^l)^T V \hat{D} c^L + \lambda \|\hat{D}\|_F^2 \\ &= \min_{B_1, D_2} \sum_{\mu} (\Delta z^l)^T [V_l \ V_\emptyset] \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} c^L + \lambda \sum_{j=1}^2 \|D_j\|_F^2 \\ &= \min_{D_1, D_2} (\Delta z^l)^T (V_l D_1 + V_\emptyset D_2) c^L + \lambda \sum_{j=1}^2 \|D_j\|_F^2 \\ &= \mathbf{L}_* + \min_{D_2} (\Delta z^l)^T V_\emptyset D_2 c^L + \lambda \|D_2\|_F^2\end{aligned}$$

Here, the equality of $\mathbf{L}_* = \min_{D_1} (\Delta z^l)^T V_\emptyset D_1 c^L + \lambda \|D_1\|_F^2$ can be obtained from following the same proof of $\mathcal{L}_{*,B_1}^l = \mathbf{L}_*$ and a change of variable $D_1 = U^T \hat{W}$.

For the other term, we do change of variable $D_2 = \tilde{D}_3 U^T$, then:

$$\begin{aligned}\mathcal{L}_{*,fb}^l - \mathbf{L}_* &= \min_{D_3} (\Delta z^l)^T V_\emptyset D_3 U^T R^l c^l + \lambda \|D_3\|_F^2 \\ &= \min_{D_3} (\Delta z^l)^T V_\emptyset D_3 V_l^T c^l + \lambda \|D_3\|_F^2\end{aligned}$$

Notice these are the same as parts with B_3 in layerwise difference $\mathcal{L}_*^l - \mathbf{L}_*$. So there is the additional term with B_2 and B_4 :

$$\mathcal{L}_*^l - \mathbf{L}_* = \mathcal{L}_{*,fb}^l - \mathbf{L}_* + \min_{B_2, B_4} (\Delta z^l)^T (V_l B_2 V_\emptyset^T + V_\emptyset B_4 V_\emptyset^T) c^l + \lambda (\|B_2\|_F^2 + \|B_4\|_F^2)$$

This additional deviation in loss leads to additional deviation in gradient.

By definition of $B_{*,2}$ and $B_{*,4}$ of when computing the difference of gradient with respect to W^l , we obtain the two gradients deviations for \mathcal{L}_*^l and $\mathcal{L}_{*,fb}^l$ respectively: (for simplicity, we denote $\Delta z^{\mu,l-1} = z_{\text{neg}}^{\mu,l-1} - z_{\text{pos}}^{\mu,l-1}$)

$$\begin{aligned}\frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} &= V_\emptyset D_{*,3} V_l^T c^l (\Delta z^{l-1})^T \\ \frac{\partial \mathcal{L}_*^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} &= (V_l B_{*,2} V_\emptyset^T + V_\emptyset B_{*,4} V_\emptyset^T) c^l (\Delta z^{l-1})^T + \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l}\end{aligned}$$

In the following proof, we want to show that the two extra terms make the gradient deviate more instead of cancelling the deviation from $\mathcal{L}_{*,fb}^l$.

To begin with, we prove that the first extra term $V_l B_{*,2} V_\emptyset^T c^l (\Delta z^{l-1})^T$ is orthogonal to the rest. Note that $\frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l}$ is in the form of V_\emptyset multiplied with other matrices, so each column vector of $\frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l}$ is in the column space of V_\emptyset (i.e., the subspace spanned by the column vectors of V_\emptyset). The statement is also true for the second extra term: $V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T$. In contrast the first extra term $V_l B_{*,2} V_\emptyset^T c^l (\Delta z^{l-1})^T$ is in the column space of V_l , which is orthogonal to the column space of V_\emptyset . Then:

$$\begin{aligned}\left\| \frac{\partial \mathcal{L}_*^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} \right\|_F^2 &= \left\| \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} + V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T \right\|_F^2 + \left\| V_l B_{*,2} V_\emptyset^T c^l (\Delta z^{l-1})^T \right\|_F^2 \\ &\geq \left\| \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l} + V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T \right\|_F^2\end{aligned}$$

Next, we show that $V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T$ does not cancel the deviation from $\frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathbf{L}_*}{\partial W^l}$. To do so, we first compute analytically the expression for $B_{*,4}$. Since function $V_\emptyset B_4 V_\emptyset^T c^l (\Delta z^{l-1})^T + \lambda \|B_4\|_F^2$ is strictly convex, globally minimal $B_{*,4}$ is reached when the expression is at critical point (zero derivative):

$$\begin{aligned}\frac{\partial}{\partial B_4} (\Delta z^l)^T V_\emptyset B_4 V_\emptyset^T c^l + \lambda \|B_4\|_F^2 &= 0 \text{ when evaluated at } B_{*,4} \\ \Leftrightarrow 0 &= V_\emptyset^T \Delta z^l (c^l)^T V_\emptyset + 2\lambda B_{*,4} \\ \Leftrightarrow B_{*,4} &= -\frac{1}{2\lambda} V_\emptyset^T \Delta z^l (c^l)^T V_\emptyset\end{aligned}$$

Following the same process, we can obtain $D_{*,3} = -\frac{1}{2\lambda} V_\emptyset^T \Delta z^l (c^l)^T V_l$.

Substituting the analytical expression of $B_{*,4}$ and $D_{*,3}$ into deviations of gradients, we could derive:

$$\begin{aligned}\left\| \frac{\partial \mathcal{L}_*^l}{\partial W^l} - \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} \right\|_F^2 &\geq \left\| \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathcal{L}_*^l}{\partial W^l} + V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T \right\|_F^2 \\ &= \left\| V_\emptyset D_{*,3} V_l^T c^l (\Delta z^{l-1})^T + V_\emptyset B_{*,4} V_\emptyset^T c^l (\Delta z^{l-1})^T \right\|_F^2 \\ &= \left\| -\frac{1}{2\lambda} V_\emptyset V_\emptyset^T \Delta z^l (c^l)^T (V_\emptyset V_\emptyset^T + V_l V_l^T) c^l (\Delta z^{l-1})^T \right\|_F^2 \\ &= \left\| -\frac{1}{2\lambda} V_\emptyset V_\emptyset^T \Delta z^l (\|V_\emptyset^T c^l\|_2^2 + \|V_l^T c^l\|_2^2) (\Delta z^{l-1})^T \right\|_F^2 \\ &\geq \left\| -\frac{1}{2\lambda} V_\emptyset V_\emptyset^T \Delta z^l (\|V_l^T c^l\|_2^2) (\Delta z^{l-1})^T \right\|_F^2 \\ &= \left\| -\frac{1}{2\lambda} V_\emptyset V_\emptyset^T \Delta z^l (c^l)^T (V_l V_l^T) c^l (\Delta z^{l-1})^T \right\|_F^2 = \left\| \frac{\partial \mathcal{L}_{*,fb}^l}{\partial W^l} - \frac{\partial \mathcal{L}_*^l}{\partial W^l} \right\|_F^2\end{aligned}$$

B. Training details

Simulation for deep linear networks For Figure 2 and Figure 3 A, we simulate a 6 layer deep linear network. For Figure 2, each layer has 128 neurons for figure 3.1. For figure 3 A, layer widths reduces by half in each layer (from 128 to 4). We took 1024 MNIST samples and split them into 32 batches. For each batched input, images go through only simple data augmentations of cropping a 14x14 patch from the image. Patches from the same image are used as positive samples. We use type 2 loss with softplus $f(x) = \log(1 + \exp(-x))$ because it is smooth, but the result would hold for other decreasing convex f . The regularization coefficient is $\lambda = 0.01$. To compute B_*^l , we first use one forward pass to obtain z^l and c^l . Then, we use the LBFGS optimizer to find the minimum B_*^l . Afterwards, we compute the gradients using B_*^l .

For linear CNN (figure 3 B), we use 4 layers of convolutional networks, each layer with 32 channels. The kernel sizes and strides are 2. MNIST inputs are randomly cropped into 16x16 patches. Patches from the same image are used as positive samples. We use the same type 2 loss with softplus, with regularization $\lambda = 0.02$. When we train the model with spatial dependence, z^1, c^1 features in the local 2x2 patches share the same B^l in the first layer, but no features share B^l in subsequent layers (full 2D spatial dependence). This is effectively implemented by performing local 2×2 local average pooling for z^1 and c^1 , and flattening z^l, c^l in other layers.

Simulation for optimum similarity with BP update (Figure 3, B) We simulate a 6 layer ReLU MLP with 512 neurons in each layer. The training dataset in MNIST is used to train the network. For each batched input, images go through only simple data augmentations of cropping a 14x14 patch from the image. We use again type 2 loss with softplus $f(x) = \log(1 + \exp(-x))$. The batch size is 32, and we use Adam optimizer with a learning rate is 5×10^{-5} . Each model is trained for 20 epochs with different versions of local-SSL algorithms. Following training, each model use the test dataset of MNIST to evaluate cosine similarity of its gradient updates to the BP updates. For the theoretical optimum, we just change the update rule of B_o^l to the gradient descent of loss $\left\| \frac{\partial \mathcal{L}_{fb}^l}{\partial z^l}(B_o^l) - \frac{\partial \mathcal{L}_*^l}{\partial z^l} \right\|$.

For Table 2, on CIFAR10, STL10, and TinyImageNet We follow the training setup and procedure of Illing et al. (2021). We use the STL-10 dataset (Coates et al., 2011), which is a dataset designed for unsupervised learning. It contains 100,000 unlabeled images from 10 classes. It also contains 500 labeled training images and 800 test images for each class. We first train the neural network model (Table A1) on the unlabeled images using the self-supervised losses. Then, we freeze the pretrained neural network, and then train a linear classifier using the representations of labeled training images. The decoding accuracy performance is computed by evaluating the decoder on representations of the test images.

Table A1. Network Architecture

# of trainable layer	layer type
1	3x3 conv128, ReLU
2	3x3 conv256, ReLU 2x2 MaxPool
3	3x3 conv256, ReLU
4	3x3 conv512, ReLU 2x2 MaxPool
5	3x3 conv1024, ReLU 2x2 MaxPool
6	3x3 conv1024, ReLU 2x2 MaxPool

To create positive or negative samples, each image undergoes the data augmentations as in SimCLR (Chen et al., 2020). Specifically, we use the torchvision.transforms in PyTorch to perform the following augmentations: RandomResizeCrop, RandomHorizontalFlip, random ColorJitter, RandomGrayScale, random GaussianBlur.

For introducing 2D spatial dependence, we let features within the local $n \times n$ (size in Table A2) patches to share B^l . This is effectively implemented by performing local average pooling depending on the layer and the image dataset.

Table A2. Size of patches sharing B^l at different layers (L1-L6) for computing the loss

Dataset	Image Size	L1	L2	L3	L4	L5	L6
CIFAR10	32	4	4	4	2	2	1
STL10	96	12	12	12	6	6	3
Tiny-ImageNet	64	8	8	8	4	4	2

For BP-CLAPP++ training, we just take the CLAPP++ objective at the last layer, and then perform backpropagation to train earlier layers. For BP-InfoNCE, we use a linear projection head with output dimension 1024 to project z^L and c^L . Then, we use the InfoNCE loss as in Chen et al. (2020) and backpropagation to train the whole network. To examine the influence of spatial dependence in BP-SSL training, we simulate both the case with spatial dependence and without for \mathbf{L} . We found no effect of spatial dependence on performance, so we just report the number without spatial dependence.

The model receives 32 images at each batch during training. For each patch of each image, one positive contrastive sample and one negative contrastive sample are selected to compute the loss \mathcal{L}^l and to update the network parameters. We use Adam optimizer with a learning rate $2 \cdot 10^{-4}$ to train each model for 300 epochs. For STL10, each experiment takes approximately 5 hours on 4 NVIDIA A100 GPUs.

For training the linear classifier, the representation is computed by averaging the representations of all patches associated with an image. We use Adam optimizer with a learning rate 10^{-3} .

Simulation for Figure 4 C : We repeat the above simulation for STL10, except that we remove spatial dependence in the last layer for ‘CLAPP++’ and ‘CLAPP++DFB’. In this way, they use the same last layer objective (\mathbf{L}) as model ‘CLAPP++ (no spatial dependence)’, so the comparison with BP-SSL gradients are fair across these learning rules. The classification performance on STL10 does not change if we remove the spatial dependence of the last layer. CLAPP++ reaches 78.63% accuracy, and CLAPP++DFB reaches 79.01% accuracy. After the simulation, we freeze the encoder at 100 epochs (or other number of epochs in figure A2), and evaluate the gradient alignment to BP-CLAPP++ on the held out test dataset.

C. Training self-supervised objectives with BP-approximation methods

We compare our proposed learning rules with feedback to two other existing biologically plausible BP-approximation frameworks: Directed Feedback Alignment (DFA) (Nøkland, 2016) and Predictive Coding (PC) (Whittington & Bogacz, 2017). More specifically, we use these methods as approximations of backpropagation in the End-to-End training of the CLAPP objective.

Self-supervised Directed Feedback Alignment In DFA, the partial derivative of the loss with respect to the final layer activity $\delta^L = \partial \mathbf{L} / \partial z^L$ is directly projected to each intermediate layer through a random fixed matrix: $\delta^l = F^l \delta^L$. The feedforward weight is then updated with: $\Delta W_{ji}^l = \delta_j^l \rho'(a_j^l) z_i^{l-1}$. When a shallow network is trained using DFA on a simple task like MNIST, the network evolves such that δ^l in intermediate layers would approximate the true partial derivative $\partial \mathbf{L} / \partial z^l$.

Applying this method to optimize the final layer contrastive predictive loss \mathbf{L} of CLAPP++, we get the update rule for feedforward weights in the intermediate layers:

$$\Delta W_{ji}^l = \gamma^L \cdot (F^l B^L z^L)_j \cdot \rho'(a_j^l) z_i^{l-1} \quad (8)$$

Such a method based on feedback alignment is known to be unscalable to deeper networks and more complex image classification tasks (Bartunov et al., 2018). In Table 2, we also show that it performs much inferior to the CLAPP model.

The method is closely related to our proposed feedback scheme. In the CLAPP++DFB learning rule, B^l is used to replace the combined term $F^l B^L$ in the DFA method. However, the top-down B^l in CLAPP++DFB is trained with a Hebbian learning

Table A3. Image classification performance using backpropagation (BP) and predictive coding (PC) to optimize different losses.

Loss	CIFAR-10		STL-10	
	BP	PC	BP	PC
Cross-entropy (supervised)	72.41	71.92	-	-
CLAPP objective (self-supervised)	65.68	45.62	78.36	36.75
LPL objective (self-supervised)	73.65	47.68	70.29	51.21

rule, whereas the top-down connection F^l in DFA is a random fixed matrix. The broadcasting factor γ^l in CLAPP++DFB also has dependence on layer activities.

Self-supervised Predictive Coding In short, we use the JAX package for predictive coding implemented in Pinchetti et al. (2025) (<https://github.com/liukidar/pcx>) to optimize the self-supervised objective.

This method takes a different approach by modeling each layer’s activity h^l as a Gaussian distribution conditioned on the previous layer’s activity and layer parameters: $P(h^l) \sim \mathcal{N}(\mu^l, \Sigma^l)$ where $\mu^l = f(\theta^l, h^{l-1}) = W^l \rho(h^{l-1})$ for a feedforward layer, and Σ^l is usually set as identity matrix. In predictive coding, activity h^l and parameters θ^l are optimized to minimize the following energy function:

$$\mathcal{F}(h, \theta) = \sum_{l=1}^{L-1} \frac{1}{2} (h^l - \mu^l)^2 + \mathcal{L}(\mu^L, y) \quad (9)$$

Here \mathcal{L} could be the loss function defined at the final layer L . The classic predictive coding method (Whittington & Bogacz, 2017) optimizes the energy \mathcal{F} by first updating the activity h^l to an equilibrium and then updating the parameters θ^l . The feedback information is propagated down from the top layer when the activities h^l are updated using the gradient descent of \mathcal{F} :

$$\Delta h^l = \eta_h [-\epsilon^l + \rho'(h^l) \odot (W^{l+1T} \epsilon^{l+1})] \quad (10)$$

where $\epsilon^l = h^l - \mu^l$ for $l < L$ and $\epsilon^L = -\partial \mathcal{L} / \partial \mu^L$. The transpose of the feedforward weight W^{l+1T} propagates down the information ϵ^L from the final layer. After updating the activity to equilibrium h^l_* , the parameters are updated using the gradient descent, which has the Hebbian format:

$$\Delta W^l = \eta_\theta \cdot \rho(h^{l-1}) \epsilon^l \quad (11)$$

We apply this method to optimize the self-supervised CLAPP objective at the final layer by replacing $\mathcal{L}(\mu^L, y)$ with $\mathbf{L}(\rho(\mu^L))$. To optimize the energy during training, we follow the incremental predictive coding (iPC) method used in Salvatori et al. (2024) to update the activity $\{h^l\}_{l=1}^{L-1}$ and parameters $\{W^l\}_{l=1}^L$ iteratively. This method has been shown to achieve better performance and more stable training. It is also more biologically plausible than the classic predictive coding (Whittington & Bogacz, 2017) or equilibrium propagation (Scellier & Bengio, 2017) because it does not require additional signals to control separate phases of updating activities and updating parameters.

After performing a hyperparameter search and selecting the best model, we found that predictive coding fails to learn useful representations. To further confirm our findings of the inferior performance of predictive coding in Table 2, we apply the same hyperparameter search to train different losses on different datasets. We find that predictive coding only performs close to backpropagation on supervised loss, whereas the method fails to optimize the self-supervised contrastive objectives used by CLAPP (Table A3). To further show that the result is not specific to contrastive objectives like CLAPP, we also implement predictive coding to optimize the non-contrastive objectives used by LPL (Halvagal & Zenke, 2023), where we found a similar result (Table A3). Therefore, the predictive coding method could not be easily extended to self-supervised loss.

To update h^l , we follow the code in Salvatori et al. (2024), and use the SGD optimizer with a momentum of 0.5. The learning rate η_h and optimization steps T are searched according to Table A4. In supervised training, we update W^l using

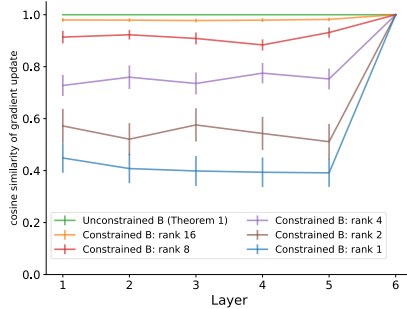


Figure A1. Cosine similarity of weight update of local-SSL and BP-SSL for B^l with different rank. Simulation is done in 6-layer deep linear network with 128 neurons in each layer.

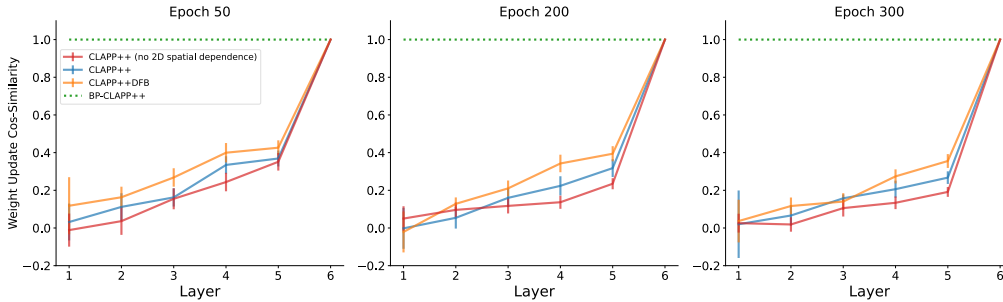


Figure A2. Cosine similarity of weight update of local-SSL and BP-SSL at different stages of the training process on STL-10.

the Adam optimizer with a learning rate $5 \cdot 10^{-5}$ and weight decay 10^{-4} . We use a batch size of 128 and train the model for 50 epochs. The test dataset is evaluated at the end of each epoch, and the best accuracy is reported. In self-supervised training, we use the same batch size, epoch number, learning rate, and optimizer for W^l as in CLAPP (Illing et al., 2021) and LPL (Halvagal & Zenke, 2023). Detailed choices are specified in Table A4:

Table A4. Hyperparameters for different losses in predictive coding. T is the number of updates for h and w for each input. η_h is the learning rate for updating h . These two hyperparameters are specific to predictive coding and are thereby tuned. Other hyperparameters are kept the same as the ones used in backpropagation training.

Loss	PC-Supervised	PC-CLAPP	PC-LPL
T	{1, 9, 17}		
η_h	{0.01, 0.1, 0.5}		
Batch size	128	32	1024
η_w	$5 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	10^{-3}
Epochs	50	300	800
Weight decay	10^{-4}	0	10^{-4}
Optimizer	Adam		

D. Additional Comparison with BP-SSL on Local-SSL

We analyze the effect of imposing constraints on B^l for theorem 3.1. We constrain the ranks of B^l by parametrizing $B^l = (F_1^l)^T F_2^l$ where F_1^l and F_2^l are trainable weight matrices, both with n^l columns and r rows (n^l is number of neurons in layer l , and $r < n^l$). Then, we repeat the same simulation for theorem 3.1 (green line in figure 2). As the rank decreases, B^l is more constrained, and the alignment between local-SSL and BP-SSL decreases.

We analyze the gradient alignment at different stages of training(Figure A2). In all stages, introducing top-down feedback improves the gradient alignment.