A method to automatically discover symbolic local learning rules

Andrea Perin[*] Aalto University

Fabio Anselmi[†] Università degli Studi di Trieste

Stéphane Deny[†] Aalto University

Abstract

We present a general method for discovering symbolic local learning rules that can replace backpropagation, applicable, in principle, to any deep learning architecture and task. The method automatically discovers local loss functions specific to each layer by (1) constructing a dictionary of monomials of local tensor variables, inspired by Penrose diagram notation, (2) regressing over coefficients weighting these monomials to build layerwise polynomial losses whose gradients together minimize a target global loss. We test the method on classification, reconstruction, and self-supervised learning tasks with small multilayer networks and minimal datasets. Across all tasks, the discovered rules induce non-trivial interactions between layers, which lead to significant reductions in the corresponding global losses. Furthermore, the method selects sparse combinations of monomials, finding local rules that are more interpretable than those generated by MLP-based alternatives. The potential implications of our method range from energy-efficient deep learning to the discovery of biologically plausible learning mechanisms.

1 Introduction

The plausibility of backpropagation as a model of biological learning is disputed [e.g., Crick, 1989, Lillicrap et al., 2016, 2020, Richards et al., 2019], mainly because it demands sharing gradient information across all layers of deep networks at each update step. A more plausible candidate would have to be local and require limited information sharing across the network [Bredenberg et al., 2023]. From a machine learning perspective, local learning rules—when supported by suitable hardware [Momeni et al., 2025]—may offer practical advantages over standard backpropagation: they reduce cross-layer information transfer, enable layers to be updated in parallel, and may reduce energy costs in hardware settings where communication is a bottleneck.

Different approaches have been proposed to replace backpropagation with local learning rules. For simple problems and shallow networks, closed-form learning rules have been derived analytically [e.g., Oja, 1982, 1989, Hyvärinen and Oja, 1998, Pehlevan et al., 2015]. For more complex problems and deeper networks, hand-engineered local learning rules have shown empirical success in specific settings [e.g., Löwe et al., 2019, Xiong et al., 2020, Siddiqui et al., 2024, Zihan et al., 2026]. Methods have also been proposed to *automatically discover* effective local learning rules [e.g., Metz et al., 2019, Gu et al., 2019, Gregor, 2020, Tyulmankov et al., 2022, Metz et al., 2022, Shervani-Tabar and Rosenbaum, 2023, Confavreux et al., 2023]. Typically, these methods train auxiliary multi-layer perceptrons (MLPs) to implement local weight updates. They, however, do not produce interpretable equations for the learning rules, as MLPs perform highly non-linear and complex transformations of their inputs.

Here, we start from the observation that many classical local learning rules can be expressed as low-order polynomial functions of locally available variables, such as weights and pre- and post-synaptic

[*]Corresponding author: andrea.perin@aalto.fi. [†]Co-last authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

activities. We postulate that, for more complex problems (deeper networks and complex tasks), approximate solutions also exist that are expressible as polynomials of accessible local variables. We design a method that turns the construction of local learning rules into a systematic, learnable procedure: for each layer, our method (1) automatically generates a dictionary of monomials of local tensor variables, leveraging a method known in physics as Penrose diagram notation, and (2) learns the coefficients weighting these elements to build polynomial local losses. The gradients of these losses induce local updates that together conspire to satisfy a target global objective. We demonstrate the feasibility of our approach on minimal problems of classification, reconstruction, and self-supervised learning.

2 Method

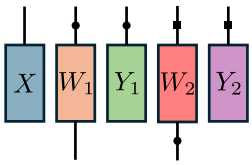
We start by choosing a *global loss function* $\mathcal{L} : \Theta \to \mathbb{R}$ that we wish to optimize, and a network architecture f_θ to train, where Θ is the space of the weights and $\theta \in \Theta$. For each layer in the network, we define its *neighborhood of degree* d as the set of variables including its inputs, weights, and outputs, as well as those of all the layers that are at most d steps away in the downstream computational graph (Fig. 2). We then construct scalar functions of elements in its neighborhood involving at most *order* K, possibly repeated, terms.

Specifically, we consider *trace monomials*, defined as those scalars that are computed by taking up to K, possibly repeated, neighborhood elements, and contracting all their indices two by two according to shape compatibility. For example, for a linear layer with weight matrix $W \in \mathbb{R}^{d\times m}$, input $x \in \mathbb{R}^m$ and output $y \in \mathbb{R}^d$, a few possible trace monomials, of orders 2, 3, and 4, respectively, are

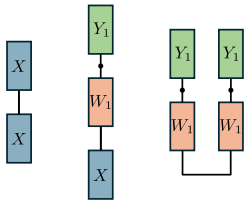
$$||x||^2 = \sum_i x_i x_i, \quad y^\top W x = \sum_{ij} W_{ij} x_j y_i, \quad ||y^\top W||^2 = \sum_{ijk} W_{ij} W_{kj} y_k y_i \tag{1}$$

where each contracted index appears *twice and only twice* in the expression.

We develop a method, inspired by Penrose's tensor diagram notation [Penrose et al., 1971], to generate all trace monomials up to order K. In this notation, a tensor of rank r is represented as a labeled box from which r different "legs" protrude, one for each index in the tensor, and each labeled (by equipping it with bullet points, squares, or any other graphical sign) to reflect the *size* of the respective index (Fig. 1a). Two tensors can be connected by joining them on a common leg; the corresponding index is understood to be *contracted*, i.e., summed over. For a rank-0 tensor (i.e., a scalar), the box has no protruding legs. Our method generates all combinations of up to K tensors in which all legs are paired according to their shapes, leading to scalars — the trace monomials (Fig. 1b). In all our experiments, we keep $K = 4$. In practice, we use the `einsum` function to compute the numerical values of the trace monomials (see Appendix A for a pseudo-code description of the method).



(a) Examples of tensor variables in Penrose diagram notation. Vectors have a single leg, matrices have two, in general differently sized, legs.



(b) Trace monomials in Eq. 1 written in Penrose diagram notation.

Figure 1: Examples illustrating the Penrose diagram notation.

Using these monomials as dictionary elements, for each layer ℓ in the network we construct a *local loss function*, defined as a weighted sum of its trace monomials:

$$\widehat{\mathcal{L}}^\ell = \sum_n \alpha_n^\ell \widehat{\mathcal{L}}_n^\ell. \tag{2}$$

We refer to the weights $\alpha_n^\ell \in \mathbb{R}$ as *rule coefficients* for layer ℓ, and initialize them to 0.

We now describe a *training episode*. We start by randomly initializing the network parameters. Each parameter is then recursively updated by using the gradients obtained from their respective local loss function, as in standard gradient descent, for a certain number T of time steps. For each layer ℓ in the network (dropping the ℓ superscript to simplify the notation):

$$\theta^{(t+1)}(\alpha) \leftarrow \theta^{(t)}(\alpha) + \sum_{n=1}^{N} \alpha_n \nabla_\theta \widehat{\mathcal{L}}_n(\alpha). \quad (3)$$

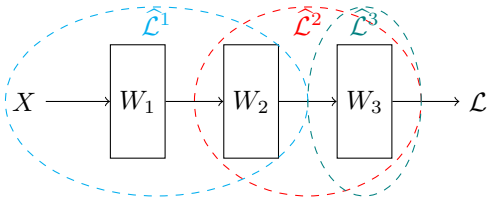


Figure 2: Schematic illustrating key notations used in the method. Here, the degree of the neighborhood is 2 (indicated by the ellipses).

We compute the global loss $\mathcal{L}$ on these resulting weights, and its gradients with respect to each layer's rule coefficients. We update each layer's rule coefficients, and conclude the training episode. In practice, a number N_{seed} of these episodes take place at once using the same rules, thus batching the process over different copies of the network parameters.

We repeat this process, each time running a new training episode with the freshly updated rule coefficients and a new set of initializations of the network parameters. This outer loop is executed until the value of $\mathcal{L}$ is below a chosen threshold.

2.1 Implementation details

The meta-optimization is numerically delicate: the outer-loop optimization is sensitive to the scale of the trace monomials, the length of the unrolled dynamics, and the stability of the induced weight updates. In our experiments, the following implementation choices were important for stable training.

Normalization of trace monomials: Different trace monomials can have wildly different magnitudes at initialization. To make all of them comparable at initialization, we measure the average value of trace monomials over S initializations of the network parameters, and scale initial α coefficients by setting them to be the inverse of these values.

Increasing time horizon: During training, we progressively increase the number of time steps T in a training episode. We also add a jitter J to the number of steps within an episode for a given realization, in order to stabilize training. We rely on *checkpointed backprop* [Wang et al., 2009, Stumm and Walther, 2010], as implemented in the `Equinox` library Kidger and Garcia [2021]. For memory reasons, this algorithm can only be run on a limited amount of data. Thus, for each training episode, we sample a subset B of the whole dataset; then, for each step in the episode, we subsample mini-batches of size b from this subset.

Early stopping: For a given initialization of the network parameters, a specific rule may quickly lead to divergent weights or, on the contrary, to vanishing weights. We stop a training episode as soon as the max-norm of *any* of the greedily updated weights becomes either larger or smaller than a chosen maximum/minimum safety value. This saves compute time, but most importantly ensures that the resulting gradients for the rule coefficients are not themselves exploding or vanishing. For all experiments, we use a maximum of 10^2 and a minimum of 10^{-2}.

Minimax training: At every initialization of the network parameters, we sample N_{seed} different copies in parallel. After following the current rule for the necessary number of time steps, we keep a subset $N_{\text{keep}} \leq N_{\text{seed}}$ of the *worst-performing* seeds. The gradients for the rule coefficients are the average over the N_{keep} worst-performing seeds.

Optimizers: For the inner loop, we use SGD with Nesterov momentum and unit learning rate. For the rule coefficients, we use AdamW with learning rate η.

Sparse prior on rule coefficients: Numerically, we form rule coefficients α as a Hadamard product between two vectors α_1 and α_2 on which gradient updates are applied. This procedure results in an implicit sparse prior [Aladrah et al., 2026] on rule coefficients, which we found to improve optimization results. As described above, α_1 is initialized to the inverse norm of the corresponding trace monomial, and α_2 is set to 0 at initialization.

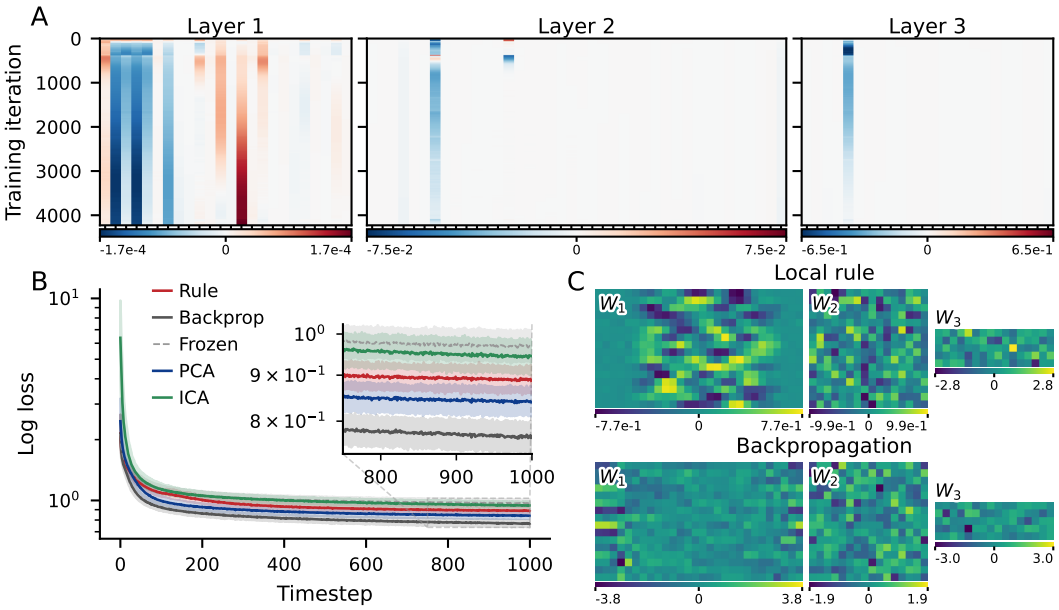


Figure 3: Classification task on the Sliced MNIST held-out split. (A) Evolution of α coefficients throughout training for a representative run. Each tick represents the α coefficient for one trace monomial of our dictionary. (B) Loss evolution throughout application of the learned rules (red), compared to backpropagation (grey), backpropagation with first layer frozen (dashed light grey), backpropagation with first layer frozen to the dataset's PCA (blue) and ICA (green). Results are aggregated over $N_{\text{eval}} = 128$ network initializations: the median loss value is reported as a solid line, and the shaded region corresponds to the 10th–90th percentile range. (C) Weights of the network after training with our learned rules (top) vs. backpropagation (bottom). Weights for PCA and ICA are shown in App. Fig. 11.

3 Results

3.1 Classification

Setup For classification, we consider a cross-entropy target loss with L2 regularization of weights ($\lambda_{\text{reg}} = 10^{-4}$), and use a simple, three-layer MLP with ReLU activations in the first and second layers, and log-softmax activation in the last layer. Thus, the classification portion of the loss function is the negative dot product between the network's output and the one-hot encoded label.

To keep computational load minimal, we used a low-dimensional version of MNIST, *Sliced MNIST*, in which we extract the middle (14th) row of pixels in the image, resulting in inputs of dimension 28. In order to make the classification problem more challenging, we randomly flip the sign of each sample vector: this last step makes the problem unsolvable by linear methods. We standardize the dataset to 0 mean and unit variance before using it. We then split the dataset into a training split consisting of classes (0,1,2,3,4), and a held-out split consisting of classes (5,6,7,8,9). We apply the same pipeline to Fashion-MNIST, of which we keep classes (5,6,7,8,9); the rules we find on the training split are then used both on the held-out split and on the Sliced Fashion-MNIST dataset, to test how the rules we find *transfer* across similar datasets. We choose the following weight sizes: $W_1 : (16, 28), W_2 : (16, 16), W_3 : (5, 16)$. We use $N_{\text{seed}} = 512$, $N_{\text{keep}} = 64$, $\eta = 3 \times 10^{-5}$, $B = 2048, b = 256$, $J = 0$, and T ranging from 1 to 500 over 4000 training episodes, and then staying at 500 for an additional 200 training episodes.

We consider neighborhoods of all possible degrees: for degree 1, the rules are *purely local*, as there are no cross-layer interactions. Conversely, at degree 3, all layers have access to the whole downstream computational graph, thus rules are *global*. At degree 2, the rules include nearest-neighbor interactions: layers 2 and 3 have access to the whole downstream computational graph, while layer 1 is limited to itself and layer 2. Thus, degree 2 is *local*, but not purely. By sweeping over the possible degrees, we show how the method can interpolate between purely local and global rules.

Results We test the learned rules by using them to train $N_{\text{eval}} = 128$ different copies of the same architecture we trained on, using the held-out splits of Sliced MNIST and Sliced Fashion-MNIST. The results for degree 2 on the held-out split of Sliced MNIST are reported in Fig. 3. Results for the other data splits, including Sliced Fashion-MNIST, are shown in App. Fig. 7. To meaningfully evaluate our method, we compare its performance to four alternatives: (1) standard backpropagation, (2) keeping the first layer frozen to its initialization value, and training only the last two layers (hereby called *frozen backprop*), (3) keeping the first layer frozen to the dataset's principal components (PCA)—a learner implementing a generalized Hebbian/Sanger rule will converge to this solution [Sanger, 1989, Oja, 1989], and (4) keeping the first layer frozen to the independent components (ICA)—a learner implementing a version of Oja's rule will converge to this solution Hyvärinen and Oja [1998]. Among these baselines, frozen backpropagation constitutes a lower threshold target: at degree 2, the last two layers have access to the target loss, which can be written as a dot product between the one-hot encoded labels and the net's outputs. Thus, we expect that a local update rule for layer 1 will *improve* on the results achievable by frozen backprop. The remaining baselines, instead, serve to provide a comparison to other local learning methods.

When local rules learned by our method are applied, the loss decreases more slowly than it does with backpropagation, but faster than frozen backpropagation—thus achieving the minimal result (see Fig. 3B). Our rule also outperforms the ICA baseline, but not the PCA baseline (rule: 0.89, frozen: 0.97, backprop: 0.77, PCA: 0.84, ICA: 0.94). In terms of accuracy, our rule achieves 67% (backprop: 71%, frozen: 63%, PCA: 68%, ICA: 65%). These results are remarkable, since classification tasks are known to be especially challenging to solve with local learning rules (see Baldi and Sadowski [2015] and Discussion).

A feature of our method is that it affords explicit, symbolic expressions for the local losses. In the following, we denote by $\hat{Y}$ the one-hot encoded labels. We restrict ourselves to a few of the top-magnitude terms for clarity[2]:

$$\begin{aligned}
\widehat{\mathcal{L}}^1 &= 10^{-5}\big(16.9\,\|W_1^\top Y_1\|^2 - 15.6\,Y_1 \cdot (Y_1 + W_1 X) - 8.94\,\|W_1 X\|^2 \\
&\qquad - 1.48\,Y_2 \cdot (Y_2 + W_2 Y_1) + 1.37\,\|W_1\|^2 - 1.20\,\|X\|^2 \|Y_1\|^2\big) \\
\widehat{\mathcal{L}}^2 &= 10^{-3}\big(-9.31\,\hat{Y}^\top W_3 Y_2 - 7.90\,\hat{Y} \cdot Y_3 + 1.04\,(\hat{Y} \cdot Y_3)^2\big) \\
\widehat{\mathcal{L}}^3 &= 10^{-2}\big(-2.50\,\hat{Y}^\top W_3 Y_2 - 1.88\,\hat{Y} \cdot Y_3\big)
\end{aligned} \tag{4}$$

Layers 2 and 3 are able to pick up the target loss (the term $\hat{Y} \cdot Y_3$), as well as a logit alignment monomial ($\hat{Y}^\top W_3 Y_2$). In the case of layer 2, we also recognize a squared cross-entropy term, whose gradient is aligned with the cross-entropy but with inverse sign. Layer 1 does not have access to the target loss terms, since the degree is only 2. In its loss, instead, we recognize terms that promote output magnitude ($Y_1 \cdot (Y_1 + W_1 X)$), together with others which moderate the dynamics ($\|W_1^\top Y_1\|^2$ and $\|W_1\|^2$, for instance), and forward-looking terms which contain layer 2's quantities, such as $Y_2 \cdot (Y_2 + W_2 Y_1)$. The latter terms are effectively a way for layer 1 to adapt to the features learned by layer 2, and thus may represent a surrogate of the usual backprop's backward channel.

3.2 Reconstruction

Setup We train a non-linear autoencoder on a reconstruction task on Sliced MNIST (see 3.1). We use a 3-layer MLP with sizes $W_1 : (16, 28), W_2 : (16, 16), W_3 : (28, 16)$, ReLU activations on layers 1 and 2, and no nonlinearity on the last layer. The target is the L2 loss between samples and their reconstructions.

As in the classification case, we train the rules on classes (0,1,2,3,4) of Sliced MNIST, and test them on the held-out classes (5,6,7,8,9), and on Sliced Fashion-MNIST. We use $N_{\text{seed}} = 512, N_{\text{keep}} = 256, B = 2048, b = 256, \eta = 10^{-5}$, $J = 5$, and T ranging from 1 to 300 over 2000 training episodes.

Results During training, α coefficients are selected by the method (Fig. 4A), leading to interpretable local loss equations. As local rules are subsequently applied, the loss decreases more slowly than it does with backpropagation. However, it decreases faster than it does with a degree-2 local version of backpropagation, where the first layer is frozen to initial weights and only the two last layers

[2]In the following, a norm operator applied to a matrix is understood to be a Frobenius norm.

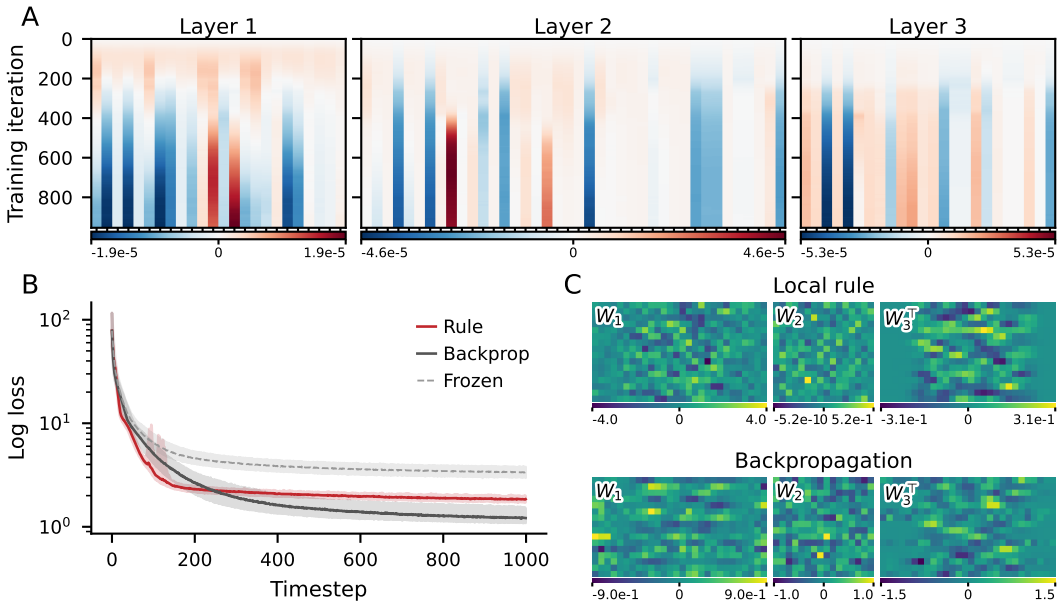


Figure 4: Reconstruction task. Same caption as Fig. 3.

are trained with backpropagation (Fig. 4B). The fact that our method beats this frozen backprop setup demonstrates that the learned losses cooperate in a non-trivial way across all layers, beyond what backpropagation limited to a neighborhood of degree-2 can accomplish (average loss values are, for rule: 1.90, frozen: 3.35, backprop: 1.27). The learned weights are structurally similar to those obtained with backpropagation (Fig. 4C). Results for the other data splits, including Sliced Fashion-MNIST, are shown in App. Fig. 8.

We report the largest terms in the rules (the cutoff being chosen for legibility):

$$\begin{aligned}
\widehat{\mathcal{L}}^1 &= 10^{-5}\big(-1.85\|W_1X\|^2 - 1.83Y_1\cdot(Y_1+W_1X) + 1.81\|Y_1^\top W_1\|^2 \\
&\qquad - 1.65\|W_2\|^2\|Y_1\|^2 - 1.31\|Y_1\|^2\|X\|^2 - 1.20\|W_2Y_1\|^2 + 1.20\|W_1^\top W_1\|^2\big) \\
\widehat{\mathcal{L}}^2 &= 10^{-5}\big(-4.30(W_3^\top X)\cdot(W_3^\top Y_3) + 4.18\|W_2Y_1\|^2 - 4.07\|W_2\|^2 X\cdot Y_3 - 3.78X\cdot(Y_3+W_3Y_2) \\
&\qquad + 2.55\|Y_2^\top W_2\|^2 - 2.50\|X\|^2 X\cdot Y_3 - 2.47(W_3^\top X)\cdot(W_2Y_1) - 2.45\|W_3\|^2 X\cdot Y_3\big) \\
\widehat{\mathcal{L}}^3 &= 10^{-5}\big(-5.23X\cdot(W_3Y_2+Y_3) - 2.82\|W_3\|^2 X\cdot Y_3 - 2.73\|X\|^2 X\cdot Y_3\big)
\end{aligned} \tag{5}$$

In this case, the target loss is not directly available as a dictionary element to *any* layer, being the squared difference $\|X - Y_3\|^2$. However, under assumptions of fixed norms $\|X\|^2$ and $\|Y_3\|^2$, it becomes proportional to $-X\cdot Y_3$. This latter term is found in layers 2 and 3, as well as terms which regularize the norm of Y_3. In layer 1, which does not have access to the final output Y_3, there are Hebbian-like terms (such as $\|W_1X\|^2$ and $\|Y_1\|^2$), reconstruction-promoting terms ($Y_1\cdot W_1X$ and $\|Y_1^\top W_1\|^2$), and forward-looking terms which contain W_2, such as $\|W_2Y_1\|^2$ and $\|W_2\|^2\|Y_1\|^2$. The latter terms are effectively a way for layer 1 to adapt to the features learned by layer 2, and thus may represent a surrogate of the usual backprop's backward channel.

3.3 Self-supervised learning

Setup We train a 3-layer MLP on a self-supervised learning task on full MNIST, flattening the digits into 784-dimensional input vectors. We consider random flips and shifts of up to 157 pixels (corresponding to a fifth of the total input length) as augmentations. The MLP has sizes $W_1 : (64, 784), W_2 : (64, 64), W_3 : (32, 64)$, and we use the Barlow Twins loss function [Zbontar et al., 2021] with tradeoff parameter $\lambda = 0.0625$, and with L2 regularization of weights ($\lambda_{\text{reg}} = 10^{-3}$). We use $N_{\text{seed}} = 128, N_{\text{keep}} = 128, B = 1024, b = 256, \eta = 5\times10^{-6}$, $J = 5$, and T ranging from 1 to 100 over 6000 training episodes. We keep T to a smaller value than in previous experiments due to stability issues in this setup.

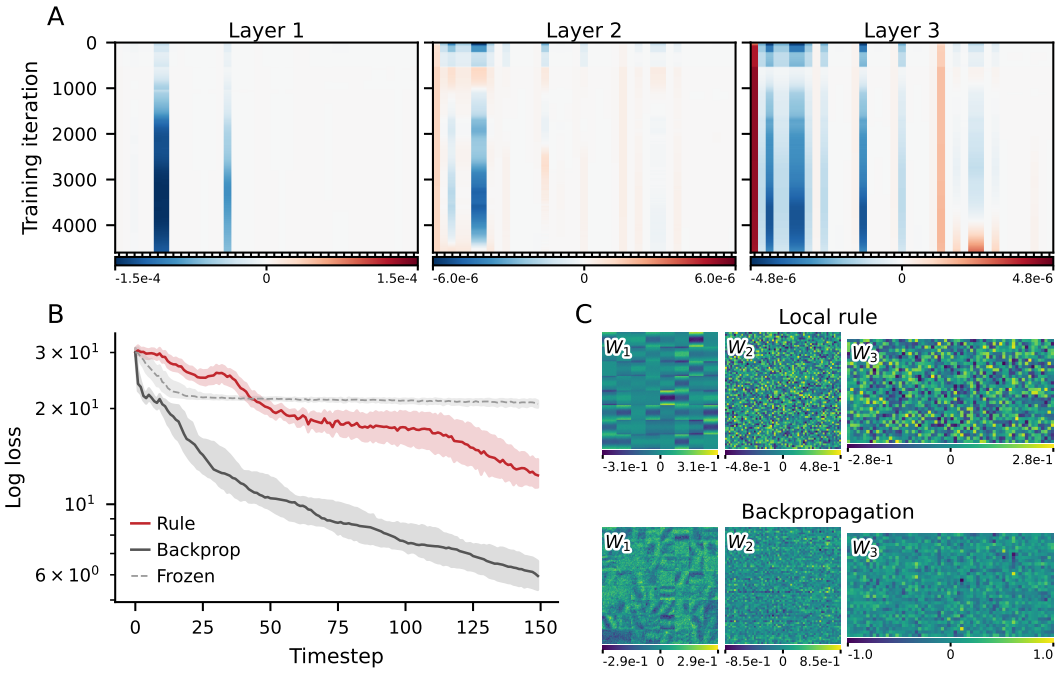


Figure 5: Self-supervised task on the MNIST held-out split. (A) Evolution of α coefficients throughout training for a representative run. Each tick represents the α coefficient for one trace monomial of our dictionary. (B) Loss evolution throughout application of the learned rules (red), compared to backpropagation (grey) and backpropagation with *first two layers* frozen (dashed light grey). Results are aggregated over $N_{\text{eval}} = 128$ network initializations: the median loss value is reported as a solid line, and the shaded region corresponds to the 10th–90th percentile range. We evaluate the rules for approximately the number of time steps they were trained for. (C) Weights of the network after training with our learned rules (top) vs. backpropagation (bottom). W_1 has been rearranged to show the weights of each neuron as 28×28 patches.

For this task, we choose to employ *layerwise local* learning rules, i.e., we choose a neighborhood of degree 1. Indeed, recent work [Zihan et al., 2026] suggests that backpropagation could be approximated with layerwise local learning rules for typical self-supervised learning objectives.

Results Local learning rules are learned on a training set of classes (0,1,2,3,4), and subsequently applied to a separate testing set of classes (5,6,7,8,9). Results on this held-out set are reported in Fig. 5. As before, when local rules learned by our method are applied, the loss decreases more slowly than it does with backpropagation, but faster than it does with a degree-1 local version of backpropagation where the first two layers are frozen to initial weights (average loss values are, for rule: 17.11, frozen: 22.47, backprop: 5.50). Results for the other data splits, including Fashion-MNIST, are shown in App. Fig. 9.

We report the largest terms in the rules (the cutoff being chosen for legibility). Terms with an apex denote one of the two augmentation pairs; we note how the method finds very similar coefficients for terms that are swap-invariant.

$$
\begin{aligned}
\widehat{\mathcal{L}}^1 &= 10^{-5}\big(-12.3(Y_1' \cdot (W_1 X) + Y_1 \cdot (W_1 X')) - 6.16(W_1 X) \cdot (W_1 X') + 1.56\|W_1\|^2\big) \\
\widehat{\mathcal{L}}^2 &= 10^{-7}\big(12.4\|W_2\|^2 + 9.22(Y_2 \cdot Y_2') + 5.94(Y_2 \cdot (W_2 Y_1 + Y_2)) + 5.92(Y_2' \cdot (W_2 Y_1' + Y_2')) \\
&\qquad + 4.60 Y_2 \cdot (W_2 Y_1') + 4.59 Y_2' \cdot (W_2 Y_1)\big) \\
\widehat{\mathcal{L}}^3 &= 10^{-6}\big(+ 5.72(W_3^\top Y_3) \cdot (W_3^\top Y_3') + 4.00\|W_3\|^2 \\
&\qquad - 2.82(Y_3 \cdot Y_3' + Y_3' \cdot (W_3 Y_2) + Y_3 \cdot (W_3 Y_2') + (W_3 Y_2) \cdot (W_3 Y_2'))\big)
\end{aligned}
\tag{6}
$$

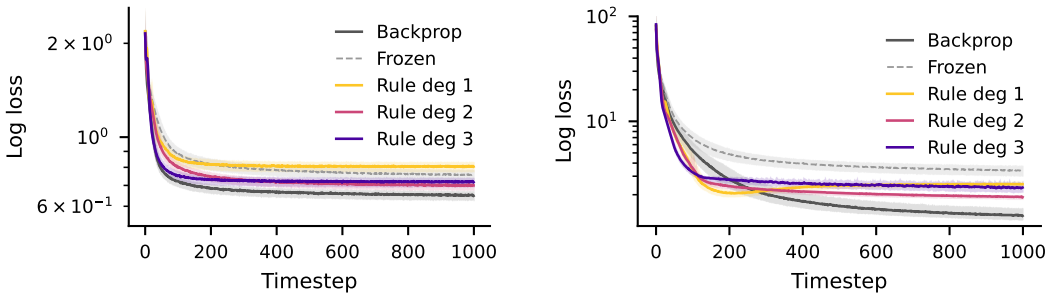


Figure 6: Effect of degree on the rules' performance as evaluated on the same-split of Sliced MNIST for classification (left) and reconstruction (right).

In these formulae, we recognize terms linked to the cross-augmentation pre- and post-nonlinearity (e.g., $Y_1' \cdot (W_1 X)$ and $Y_1 \cdot (W_1 X')$, note the symmetry between augmented pairs) and others that are linked to activation magnitudes (e.g., $||W_2 Y_1||^2$). Terms are also present that measure the cross-augmentation alignment, e.g., $(W_1 X) \cdot (W_1 X')$. We also note that many rule coefficients are matched in value. For these coefficients, we also observed that they typically evolve in a grouped manner during the optimization process, showing the same values throughout training. We take this "coefficient grouping" as a sign that our method is well-behaved. Indeed, in the self-supervised learning framework we use, inputs and outputs are always present in transformation pairs. To the model, there is no inherent difference between samples in an augmentation pair, as they share the same shape and statistics. Note that in layer 3 the grouped monomials are strictly equivalent due to the absence of nonlinearity in the final layer of our network.

3.4 Effects of degree and order

For both classification and reconstruction tasks, we perform *degree sweeps*: holding other hyperparameters fixed, we vary the degree, ranging from 1 (purely local rules) to 3 (global rules) (Fig. 6). Interestingly, we find that degree 2 can comfortably beat degree 1 on the classification task, but not on the reconstruction task, where degree 1 is competitive. This seems to corroborate previous theoretical work Baldi and Sadowski [2015] on the necessity of a backward communication channel for solving supervised tasks. Moreover, in both cases rules of degree 2 outperform global rules, but not full backpropagation. This shows that our global rules are not able to exactly recover backpropagation terms, even when they are explicitly available. Future work on the optimization side of the method could potentially address this limitation.

Another important hyperparameter in our method is the *order* of the monomials, which we fixed to $K = 4$ in all our experiments. Increasing the order results in larger dictionaries, and increases the expressive power of a ruleset (at the cost of increased computational burden). In App. Table 1 we provide the number of monomials as a function of degree and order for each of the layers in the MLP described in Sec. 3.1. We leave a detailed exploration of the effects of this hyperparameter for future work.

3.5 Shape invariance

The construction of the dictionary is shape-invariant: the number and the nature of the monomials do not depend on the specific shapes of the involved tensors. A natural question then is: are rules also shape-invariant? To test this, we run the degree-2 rules for the classification task on the held-out split of Sliced MNIST, on a network with different layer sizes than the network used for learning the rule (described in 3.1), specifically, $W_1 : (32, 28), W_2 : (16, 32), W_3 : (5, 16)$. We report the results in App. Fig. 10. The rules transfer partially to this differently sized architecture, but the performance deteriorates and stability seems to worsen. We leave further investigations on shape-invariance for future work.

4 Discussion

We presented a general method to learn equations for local learning rules. Other symbolic regression methods, such as SINDy, share a similar aim of inferring equations from data [Petersen and Landajuela, 2019, Udrescu et al., 2020, Cranmer et al., 2020, Biggio et al., 2021, Kamienny et al., 2022, Angelis et al., 2023, Makke and Chawla, 2022, Shojaee et al., 2025, Brunton et al., 2016]. They are usually employed to reverse-engineer dynamical systems from observations of their trajectories. Therein lies a crucial distinction: our method does not use trajectory data as a ground truth signal, and instead *relies solely on the signal given by the global loss*. A trajectory-based method, on the contrary, would pose both practical as well as theoretical problems: practical because of the data requirements (a whole trajectory, in our case, would consist of many copies of the weights of a model), and theoretical because what we want to approximate is not the backpropagation-induced dynamics, but instead the backpropagation-induced *performance*. In analogy with physics, we wish to apply constraints on the *macroscopic observable of the complex system* (the network's performance as measured by the global loss) rather than on its microscopic configurations.

Most previous attempts at meta-learning local rules have used MLPs [e.g., Metz et al., 2019, Gu et al., 2019, Gregor, 2020, Tyulmankov et al., 2022, Metz et al., 2022, Shervani-Tabar and Rosenbaum, 2023, Confavreux et al., 2023], trained to learn the gradients associated with specific layers in a pre-specified network architecture. Our method offers advantages over MLP-based ones. First, once the target architecture is specified, we find expressions which are more *interpretable* (as demonstrated in the Results section) than previous methods. Second, our method offers promises for transferring rules learned on small architectures and low-dimensional datasets to larger architectures and higher-dimensional datasets. Once written symbolically, the rules we find for the gradients carry no explicit dependence on the tensor shapes of data or network. Our preliminary results indicate that rules learned on one architecture can, at least partially, transfer to another architecture similar to the one used for training. Finally, our design choice of using a dictionary of low-order monomials in the local variables also distinguishes our method from the few other existing symbolic meta-learning methods, which restrict themselves either to variants of the Hebbian rule [e.g., Lindsey and Litwin-Kumar, 2020, Sandler et al., 2021], or to the estimation of optimal hyperparameters for normal backpropagation [e.g., Zheng et al., 2022].

Our attempts at finding polynomial local learning rules should not be interpreted as assuming that such rules are sufficient for arbitrary deep supervised learning, as limitations of local learning are known [Baldi and Sadowski, 2015]. In our framework, locality is a tunable parameter rather than a hard constraint: increasing the neighborhood degree gives each layer access to progressively larger portions of the downstream computational graph, up to the global case where the target loss itself can be represented at each layer. Indeed, if the target loss can be expressed as a linear combination of the trace monomials, our method can potentially recover backpropagation as one possible learning rule, where each layerwise loss is the target loss itself. Our framework is thus not limited to using purely local rules, but can be used to quantify how much nonlocal information is needed to obtain useful learning dynamics. Indeed, we have started exploring the effects of tuning the degree of our rules, and found corroborations to existing theoretical predictions. In addition to locality, our method allows us to tune the order of the trace monomials. Higher-order trace monomials can produce more complex local gradients, at the cost of increased computational burden and less stable dynamics.

Future work will also focus on scaling up the method to more realistic architectures and datasets. Improving the stability of the training process should be prioritized, as it is currently quite sensitive to hyperparameter choices, in part because the optimization landscape is fractal Sohl-Dickstein [2024]. Another promising direction is the analysis of the algebraic structure and interpretability of the rules learned by the method and, crucially, their *transferability* across datasets and architectures. If such transferability holds, the method could be used to discover local rules on simple datasets and then apply them to more complex ones. Lastly, we mention a possible alternative training method, known as the *adjoint* method [as used in, e.g., Bai et al., 2019], which relies on the implicit function theorem, and unlike the greedy procedure we presented, does not require to keep track of the whole trajectory. Instead, it requires reaching a fixed point of the dynamics, after which the gradients can be obtained by solving a linear system.

Acknowledgments

We thank Jaakko Lehtinen for useful discussions, and the Aalto Science-IT project for providing the computational resources. Funding: Research Council of Finland grant 3357590 ("Neuroscience-inspired Deep Learning") to S.D. The authors declare no competing interests.

References

N. Aladrah, E. Ballarin, M. Biagetti, A. Ansuini, A. d'Onofrio, and F. Anselmi. Understanding and inverse design of implicit bias in stochastic learning: a geometric perspective. *arXiv preprint arXiv:2601.06597*, 2026.

D. Angelis, F. Sofos, and T. Karakasidis. Artificial intelligence in physical sciences: Symbolic regression trends and perspectives. *Archives of Computational Methods in Engineering*, pages 1 – 21, 2023.

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 2019.

P. Baldi and P. Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks : The Official Journal of the International Neural Network Society*, 83:51–74, 2015.

L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo. Neural symbolic regression that scales. *arXiv*, abs/2106.06427, 2021.

C. Bredenberg, E. Williams, C. Savin, B. Richards, and G. Lajoie. Formalizing locality for normative synaptic plasticity models. *Advances in Neural Information Processing Systems*, 2023.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

B. Confavreux, P. Ramesh, P. J. Goncalves, J. H. Macke, and T. Vogels. Meta-learning families of plasticity rules in recurrent spiking networks using simulation-based inference. *Advances in Neural Information Processing Systems*, 2023.

M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. *arXiv*, 2020.

F. Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

K. Gregor. Finding online neural update rules by learning to remember. *arXiv*, 2020.

K. Gu, S. Greydanus, L. Metz, N. Maheswaranathan, and J. Sohl-Dickstein. Meta-learning biologically plausible semi-supervised update rules. *bioRxiv*, 2019.

A. Hyvärinen and E. Oja. Independent component analysis by general nonlinear Hebbian-like learning rules. *Signal Processing*, 64(3):301–313, 1998.

P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton. End-to-end symbolic regression with transformers. *arXiv*, 2022.

P. Kidger and C. Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems*, 2021.

T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7(1):13276, 2016.

T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

J. Lindsey and A. Litwin-Kumar. Learning to learn with feedback and local plasticity. *Advances in Neural Information Processing Systems*, 33, 2020.

S. Löwe, P. O'Connor, and B. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in Neural Information Processing Systems*, 2019.

N. Makke and S. Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57:1–38, 2022.

L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. *International Conference on Learning Representations*, 2019.

L. Metz, J. Harrison, C. D. Freeman, A. Merchant, L. Beyer, J. Bradbury, N. Agrawal, B. Poole, I. Mordatch, A. Roberts, et al. Velo: Training versatile learned optimizers by scaling up. *arXiv*, 2022.

A. Momeni, B. Rahmani, B. Scellier, L. G. Wright, P. L. McMahon, C. C. Wanjura, Y. Li, A. Skalli, N. G. Berloff, T. Onodera, et al. Training of physical neural networks. *Nature*, 645(8079):53–61, 2025.

E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(01):61–68, 1989.

C. Pehlevan, T. Hu, and D. B. Chklovskii. A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, 27 (7):1461–1495, 2015.

R. Penrose et al. Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications*, 1(221-244):3, 1971.

B. K. Petersen and M. Landajuela. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv*, 2019.

B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli, et al. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019.

M. Sandler, M. Vladymyrov, A. Zhmoginov, N. Miller, T. Madams, A. Jackson, and B. A. Y. Arcas. Meta-learning bidirectional update rules. In *International Conference on Machine Learning*. PMLR, 2021.

T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989. ISSN 0893-6080. URL `https://www.sciencedirect.com/science/article/pii/0893608089900440`.

N. Shervani-Tabar and R. Rosenbaum. Meta-learning biologically plausible plasticity rules with random feedback pathways. *Nature Communications*, 14(1):1805, 2023.

P. Shojaee, K. Meidani, S. Gupta, A. B. Farimani, and C. K. Reddy. LLM-SR: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*, 2025.

S. A. Siddiqui, D. Krueger, Y. LeCun, and S. Deny. Blockwise self-supervised learning at scale. *Transactions in Machine Learning Research*, 2024.

J. Sohl-Dickstein. The boundary of neural network trainability is fractal. *arXiv*, 2024.

P. Stumm and A. Walther. New algorithms for optimal online checkpointing. *SIAM Journal on Scientific Computing*, 32(2):836–854, 2010.

D. Tyulmankov, G. R. Yang, and L. Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557, 2022.

S. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *arXiv*, 2020.

Q. Wang, P. Moin, and G. Iaccarino. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567, 2009.

Y. Xiong, M. Ren, and R. Urtasun. LoCo: Local contrastive representation learning. *Advances in Neural Information Processing Systems*, 2020.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. *International Conference on Machine Learning*, 2021.

W. Zheng, T. Chen, T.-K. Hu, and Z. Wang. Symbolic learning to optimize: Towards interpretability and scalability. *arXiv preprint arXiv:2203.06578*, 2022.

W. S. Zihan, A. Delrocq, W. Gerstner, and G. Bellec. Can local learning match self-supervised backpropagation? *International Conference on Machine Learning*, 2026.

A Method to list trace monomials

The following code snippet shows the way we enumerate trace monomials.

```python
from collections import Counter
from typing import Tuple, Generator, Sequence, List
from sympy import Symbol
import itertools as its


# Type aliases
Shape = Tuple[Symbol, ...]
T = Tuple[str, Shape]
Signature = Tuple[Tuple[str, ...], Tuple[Tuple[int, ...], ...]]


def pairings(things: List[int]):
  """
  >>> list(pairings([1, 2, 3, 4]))
  [[(1, 2), (3, 4)], [(1, 3), (2, 4)], [(1, 4), (2, 3)]]
  """
  if len(things) == 2:
    yield [tuple(things)]
  else:
    for i in range(1, len(things)):
      yield from (
        [(things[0], things[i])] + p
        for p in pairings(things[1:i] + things[i + 1 :])
      )


def generate_traces(tlist: Sequence[T], n: int):
  assert n > 1
  combos = its.combinations_with_replacement(tlist, n)
  for tensors in combos:
    splitpoints = (0,) + tuple(its.accumulate([len(s) for n, s in tensors]))
    tnames = tuple(n for n, s in tensors)
    all_idxs = tuple(its.chain.from_iterable(t[1] for t in tensors))
    counts = Counter(all_idxs)
    if all(v % 2 == 0 for k, v in counts.items()):
      positions = [
        [p for p, sym in enumerate(all_idxs) if sym == s] for s in counts
      ]
      allpairings = its.product(*tuple(map(pairings, positions)))
```

```
      for p in allpairings:
        assignments = [-1 for t in all_idxs]
        p = its.chain.from_iterable(p)
        for v, (p1, p2) in enumerate(p):
          assignments[p1] = assignments[p2] = v
        assignments = [
          assignments[splitpoints[idx] : splitpoints[idx + 1]]
          for idx in range(len(tnames))
        ]
        assignments = tuple(tuple(a) for a in assignments)
        is_invalid = any(
          Counter(a).most_common(1)[0][1] > 1 for a in assignments
        )
        if not is_invalid:
          yield (tnames, assignments)
```

Listing 1: Function to generate trace monomials.

B Trace monomial counts

Table 1: Monomial counts in a 3-layer MLP as a function of degree and order.

Order	Degree 1	Degree 2	Degree 3
2	$[2, 2, 3]$	$[3, 4, 3]$	$[5, 4, 3]$
3	$[3, 3, 5]$	$[5, 7, 5]$	$[9, 7, 5]$
4	$[11, 11, 24]$	$[24, 40, 24]$	$[58, 40, 24]$
5	$[18, 18, 42]$	$[45, 78, 42]$	$[120, 78, 42]$
6	$[57, 57, 155]$	$[171, 328, 155]$	$[536, 328, 155]$

C Additional results

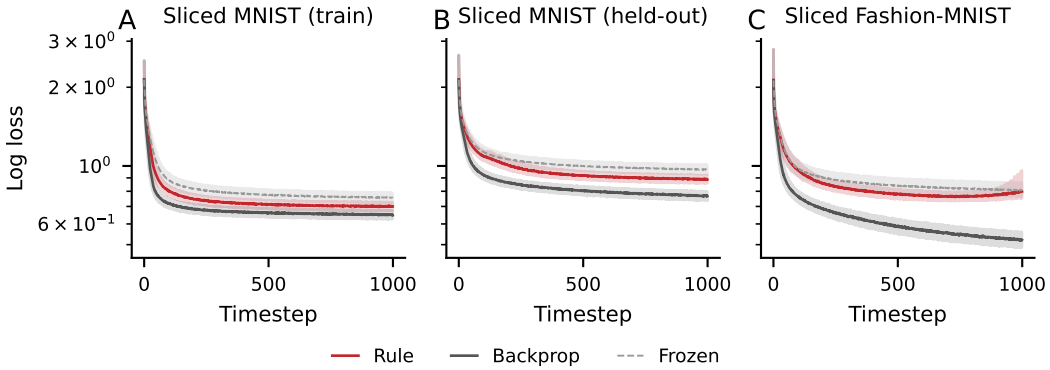


Figure 7: Comparison of the performance of the learned degree-2 rules for the classification task on (A) the same Sliced MNIST split that was used for training, (B) the held-out split of Sliced MNIST, and (C) Sliced Fashion-MNIST. Performance is best on the same and held-out splits, while it degrades slightly at the tail end of the training on the Fashion-MNIST split.

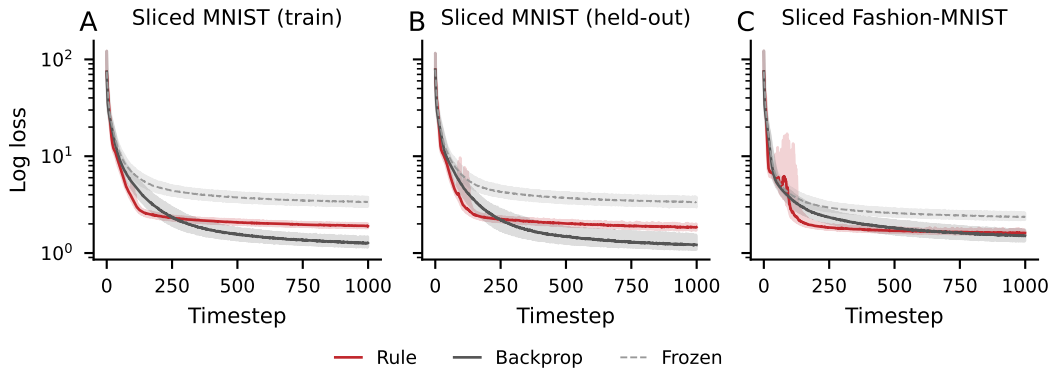


Figure 8: Comparison of the performance of the learned degree-2 rules for the reconstruction task on (A) the same Sliced MNIST split that was used for training, (B) the held-out split of Sliced MNIST, and (C) Sliced Fashion-MNIST. Performance is similar across tasks, with slightly higher instability on the Sliced Fashion-MNIST split.

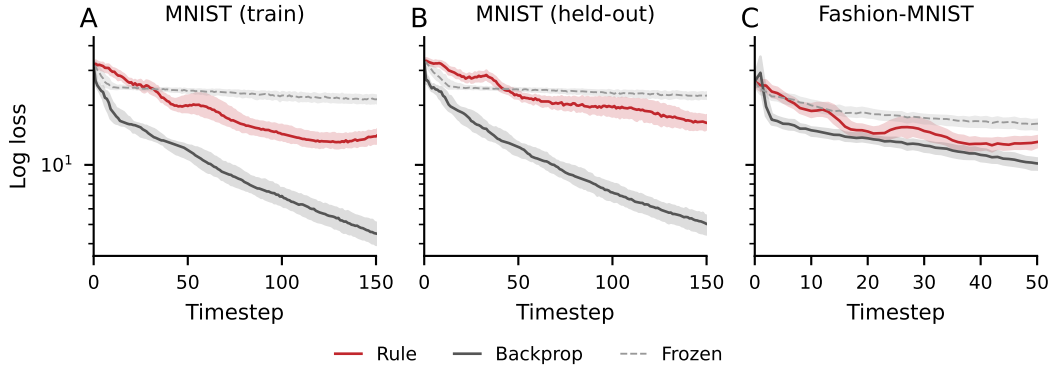


Figure 9: Comparison of the performance of the learned degree 1 rules for the self-supervised learning task on (A) the same MNIST split that was used for training, (B) the held-out split of MNIST, and (C) Fashion-MNIST. We stopped training early on the Fashion-MNIST split due to stability issues.

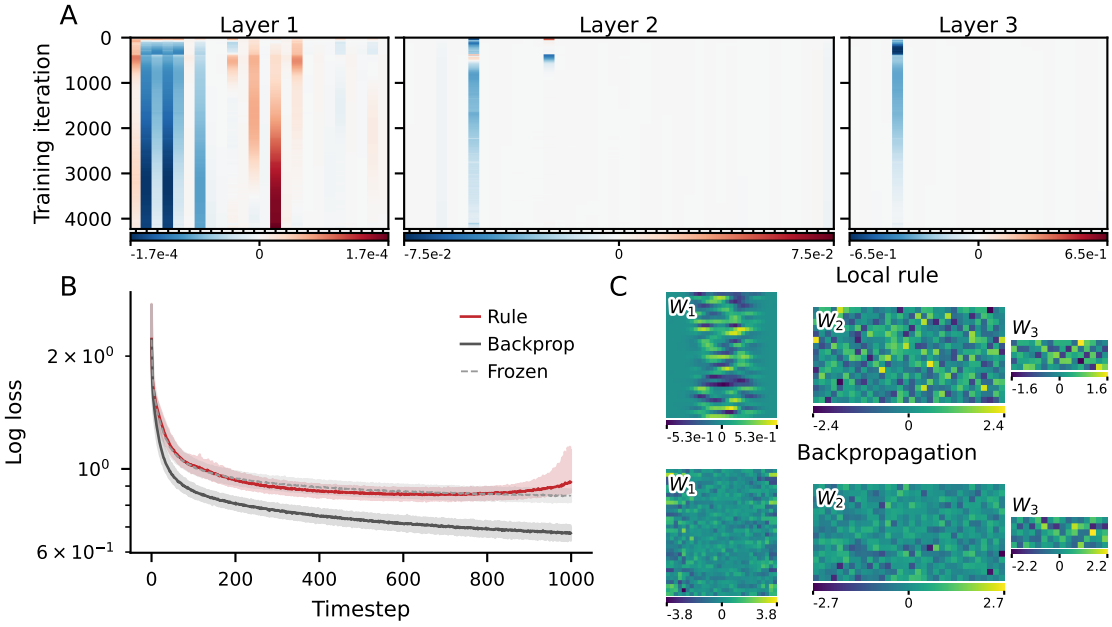


Figure 10: Testing the degree-2 classification rules on a differently sized network.

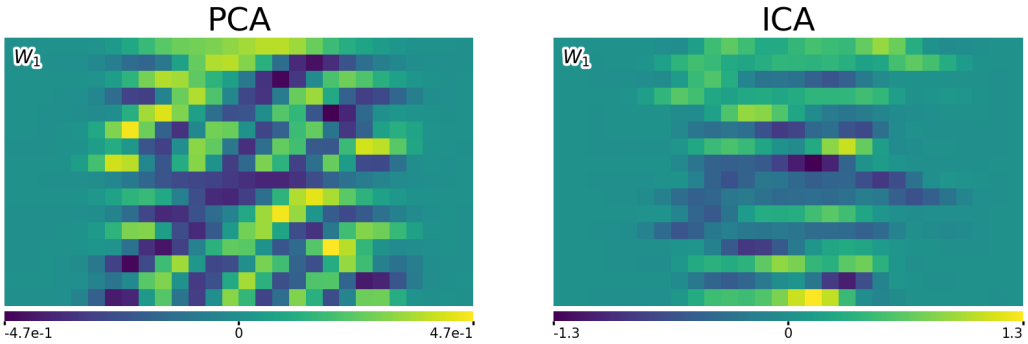


Figure 11: First frozen layer for the PCA (left) and ICA (right) cases in the held-out split of Sliced MNIST. In order to compose the first layer, we first run PCA/ICA with a number of components that is half the layer's output dimension; we then stack them with a copy multiplied by -1. We do so to address the presence of the ReLU nonlinearity.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract summarizes the main experimental findings described in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the results and discussion sections, we mention the limitations of the method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper is mostly empirical in nature, and no proofs are provided.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the method so that it should be reproducible. We also attach zipped code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The dataset is basic and widely available (MNIST), and we provide the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant experimental details are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports information regarding the significance of the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports key hardware/compute time details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We followed the NeurIPS Code of Ethics at the best of our effort.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper explicitly addresses the potential impact of the method therein described.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper is concerned with an early prototype of the method, for which there is no current risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators are credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing is involved in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not involved in any important, original or non-standard component of the project.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.