Two Routes to Scalable Credit Assignment without Weight Symmetry

Daniel Kunin [* 1]   **Aran Nayebi** [* 2]   **Javier Sagastuy-Brena** [* 1]
Surya Ganguli [3]   **Jonathan M. Bloom** [4 5]   **Daniel L. K. Yamins** [6 7 8]

Abstract

The neural plausibility of backpropagation has long been disputed, primarily for its use of non-local weight transport — the biologically dubious requirement that one neuron instantaneously measure the synaptic weights of another. Until recently, attempts to create local learning rules that avoid weight transport have typically failed in the large-scale learning scenarios where backpropagation shines, e.g. ImageNet categorization with deep convolutional networks. Here, we investigate a recently proposed local learning rule that yields competitive performance with backpropagation and find that it is highly sensitive to metaparameter choices, requiring laborious tuning that does not transfer across network architecture. Our analysis indicates the underlying mathematical reason for this instability, allowing us to identify a more robust local learning rule that better transfers without metaparameter tuning. Nonetheless, we find a performance and stability gap between this local rule and backpropagation that widens with increasing model depth. We then investigate several non-local learning rules that relax the need for instantaneous weight transport into a more biologically-plausible "weight estimation" process, showing that these rules match state-of-the-art performance on deep networks and operate effectively in the presence of noisy updates. Taken together, our results suggest two routes towards the discovery of neural implementations

for credit assignment without weight symmetry: further improvement of local rules so that they perform consistently across architectures and the identification of biological implementations for non-local learning mechanisms.

1. Introduction

Backpropagation is the workhorse of modern deep learning and the only known learning algorithm that allows multi-layer networks to train on large-scale tasks. However, any exact implementation of backpropagation is inherently non-local, requiring instantaneous weight transport in which backward error-propagating weights are the transpose of the forward inference weights. This violation of locality is biologically suspect because there are no known neural mechanisms for instantaneously coupling distant synaptic weights. Recent approaches such as feedback alignment (Lillicrap et al., 2016) and weight mirror (Akrout et al., 2019) have identified circuit mechanisms that seek to approximate backpropagation while circumventing the weight transport problem. However, these mechanisms either fail to operate at large-scale (Bartunov et al., 2018) or, as we demonstrate, require complex and fragile metaparameter scheduling during learning. Here we present a unifying framework spanning a space of learning rules that allows for the systematic identification of robust and scalable alternatives to backpropagation.

To motivate these rules, we replace tied weights in backpropagation with a regularization loss on untied forward and backward weights. The forward weights parametrize the global cost function, the backward weights specify a descent direction, and the regularization constrains the relationship between forward and backward weights. As the system iterates, forward and backward weights dynamically align, giving rise to a pseudogradient. Different regularization terms are possible within this framework. Critically, these regularization terms decompose into geometrically natural primitives, which can be parametrically recombined to construct a diverse space of credit assignment strategies. This space encompasses existing approaches (including feedback alignment and weight mirror), but also elucidates novel learning rules. We show that several of these new strategies

are competitive with backpropagation on real-world tasks (unlike feedback alignment), without the need for complex metaparameter tuning (unlike weight mirror). These learning rules can thus be easily deployed across a variety of neural architectures and tasks. Our results demonstrate how high-dimensional error-driven learning can be robustly performed in a biologically motivated manner.

2. Related Work

Soon after Rumelhart et al. (1986) published the backpropagation algorithm for training neural networks, its plausibility as a learning mechanism in the brain was contended (Crick, 1989). The main criticism was that backpropagation requires exact transposes to propagate errors through the network and there is no known physical mechanism for such an "operation" in the brain. This is known as the *weight transport problem* (Grossberg, 1987). Since then many credit assignment strategies have proposed circumventing the problem by introducing a distinct set of feedback weights to propagate the error backwards. Broadly speaking, these proposals fall into two groups: those that encourage symmetry between the forward and backward weights (Lillicrap et al., 2016; Nøkland, 2016; Bartunov et al., 2018; Liao et al., 2016; Xiao et al., 2019; Moskovitz et al., 2018; Akrout et al., 2019), and those that encourage preservation of information between neighboring network layers (Bengio, 2014; Lee et al., 2015; Bartunov et al., 2018).

The latter approach, sometimes referred to as target propagation, encourages the backward weights to locally invert the forward computation (Bengio, 2014). Variants of this approach such as difference target propagation (Lee et al., 2015) and simplified difference target propagation (Bartunov et al., 2018) differ in how they define this inversion property. While some of these strategies perform well on shallow networks trained on MNIST and CIFAR10, they fail to scale to deep networks trained on ImageNet (Bartunov et al., 2018).

A different class of credit assignment strategies focuses on encouraging or enforcing symmetry between the weights, rather than preserving information. Lillicrap et al. (2016) introduced a strategy known as feedback alignment in which backward weights are chosen to be fixed random matrices. Empirically, during learning, the forward weights partially align themselves to their backward counterparts, so that the latter transmit a meaningful error signal. Nøkland (2016) introduced a variant of feedback alignment where the error signal could be transmitted across long range connections. However, for deeper networks and more complex tasks, the performance of feedback alignment and its variants break down (Bartunov et al., 2018).

Liao et al. (2016) and Xiao et al. (2019) took an alternative

route to relaxing the requirement for exact weight symmetry by transporting just the sign of the forward weights during learning. Moskovitz et al. (2018) combined sign-symmetry and feedback alignment with additional normalization mechanisms. These methods outperform feedback alignment on scalable tasks, but still perform far worse than backpropagation. It is also not clear that instantaneous sign transport is more biologically plausible than instantaneous weight transport.

More recently, Akrout et al. (2019) introduced weight mirror (WM), a learning rule that incorporates dynamics on the backward weights to improve alignment throughout the course of training. Unlike previous methods, weight mirror achieves backpropagation level performance on ResNet-18 and ResNet-50 trained on ImageNet.

Concurrently, Kunin et al. (2019) suggested training the forward and backward weights in each layer as an encoder-decoder pair, based on their proof that L_2-regularization induces symmetric weights for linear autoencoders. This approach incorporates ideas from both information preservation and weight symmetry.

A complementary line of research (Xie & Seung, 2003; Scellier & Bengio, 2017; Bengio et al., 2017; Guerguiev et al., 2017; Whittington & Bogacz, 2017; Sacramento et al., 2018; Guerguiev et al., 2019) investigates how learning rules, even those that involve weight transport, could be implemented in a biologically mechanistic manner, such as using spike-timing dependent plasticity rules and obviating the need for distinct phases of training. In particular, Guerguiev et al. (2019) show that key steps in the Kunin et al. (2019) regularization approach could be implemented by a spike-based mechanism for approximate weight transport.

In this work, we extend this regularization approach to formulate a more general framework of credit assignment strategies without weight symmetry, one that encompasses existing and novel learning rules. Our core result is that the best of these strategies are substantially more robust across architectures and metaparameters than previous proposals.

3. Regularization Inspired Learning Rule Framework

We consider the credit assignment problem for neural networks as a layer-wise regularization problem. We consider a network parameterized by forward weights θ_f and backward weights θ_b. Informally, the network is trained on the sum of a global task function $\mathcal{J}$ and a layer-wise regularization function[1] $\mathcal{R}$:

$$\mathcal{L}(\theta_f, \theta_b) = \mathcal{J}(\theta_f) + \mathcal{R}(\theta_b).$$

---

[1] $\mathcal{R}$ is not regularization in the traditional sense, as it does not directly penalize the forward weights θ_f from the cost function $\mathcal{J}$.

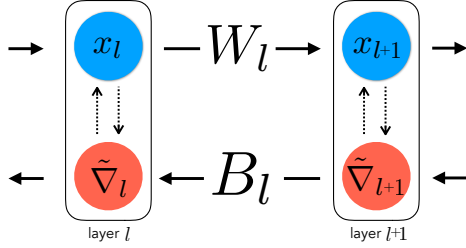| Local	$\mathcal{P}_l$	$\nabla\mathcal{P}_l$
decay	$\frac{1}{2}\|B_l\|^2$	B_l
amp	$-\mathrm{tr}(x_l^\mathsf{T} B_l x_{l+1})$	$-x_l x_{l+1}^\mathsf{T}$
null	$\frac{1}{2}\|B_l x_{l+1}\|^2$	$B_l x_{l+1} x_{l+1}^\mathsf{T}$
Non-local	$\mathcal{P}_l$	$\nabla\mathcal{P}_l$
sparse	$\frac{1}{2}\|x_l^\mathsf{T} B_l\|^2$	$x_l x_l^\mathsf{T} B_l$
self	$-\mathrm{tr}(B_l W_l)$	$-W_l^\mathsf{T}$

Figure 1. **Notational diagram.** The forward weight W_l propagates the input signal x_l downstream through the network. The backward weight B_l propagates the pseudogradient $\widetilde{\nabla}_{l+1}$ of $\mathcal{J}$ upstream through the network. The regularization function $\mathcal{R}$ is constructed layer-wise from x_l, x_{l+1}, W_l and B_l. Similar to Akrout et al. (2019), we assume lateral pathways granting the backward weights access to x and the forward weights access to $\widetilde{\nabla}$. Biases and non-linearities are omitted from the diagram.

Formally, every step of training consists of two updates, one for the forward weights and one for the backward weights. The forward weights are updated according to the error signal on $\mathcal{J}$ propagated through the network by the backward weights, as illustrated in Fig. 1. The backward weights are updated according to gradient descent on $\mathcal{R}$.

$$\Delta\theta_f \propto \widetilde{\nabla}J \qquad \Delta\theta_b \propto \nabla R$$

Thus, $\mathcal{R}$ is responsible for introducing dynamics on the backward weights, which in turn impacts the dynamics of the forward weights. The functional form of $\mathcal{R}$ gives rise to different learning rules and in particular the locality of a given learning rule depends solely on the locality of the computations involved in $\mathcal{R}$.

3.1. Regularization Primitives

In this work, the regularization function $\mathcal{R}$ is built from a set of simple *primitives* $\mathcal{P}$, which at any given layer l are functions of the forward weight $W_l \in \theta_f$, backward weight $B_l \in \theta_b$, layer input x_l, and layer output x_{l+1} as depicted in Fig. 1. These primitives are biologically motivated components with strong geometric interpretations, from which more complex learning rules may be algebraically constructed.

The primitives we use, displayed in Table 1, can be organized into two groups: those that involve purely local operations and those that involve at least one non-local operation. To classify the primitives, we use the criteria for locality described in Whittington & Bogacz (2017): (1) *Local computation.* Computations only involve synaptic weights acting on their associated inputs. (2) *Local plasticity.* Weight modifications only depend on pre-synaptic and post-synaptic activity. A primitive is local if it satisfies both of these constraints and non-local otherwise.

We introduce three **local primitives**: $\mathcal{P}^{\mathrm{decay}}$, $\mathcal{P}^{\mathrm{amp}}$, and

Table 1. **Regularization primitives.** Mathematical expressions for local and non-local primitives and their gradients with respect to the backward weight B_l. Note, both x_l and x_{l+1} are the post-nonlinearity rates of their respective layers.

$\mathcal{P}^{\mathrm{null}}$. The *decay* primitive can be understood as a form of energy efficiency penalizing the Euclidean norm of the backward weights. The *amp* primitive promotes alignment of the layer input x_l with the reconstruction $B_l x_{l+1}$. The *null* primitive imposes sparsity in the layer-wise activity through a Euclidean norm penalty on the reconstruction $B_l x_{l+1}$.

We consider two **non-local primitives**: $\mathcal{P}^{\mathrm{sparse}}$ and $\mathcal{P}^{\mathrm{self}}$. The *sparse* primitive promotes energy efficiency by penalizing the Euclidean norm of the activation $x_l^\mathsf{T} B_l$. This primitive fails to meet the *local computation* constraint, as B_l describes the synaptic connections from the $l+1$ layer to the l layer and therefore cannot operate on the input x_l.

The *self* primitive promotes alignment of the forward and backward weights by directly promoting their inner product. This primitive fails the *local plasticity* constraint, as its gradient necessitates that the backward weights can exactly measure the strengths of their forward counterparts.

3.2. Building Learning Rules from Primitives

These simple primitives can be linearly combined to encompass existing credit assignment strategies, while also elucidating natural new approaches.

Feedback alignment (FA) (Lillicrap et al., 2016) corresponds to no regularization, $\mathcal{R}_{\mathrm{FA}} \equiv 0$, effectively fixing the backward weights at their initial random values[2].

The **weight mirror (WM)** (Akrout et al., 2019) update, $\Delta B_l = \eta x_l x_{l+1}^\mathsf{T} - \lambda_{\mathrm{WM}} B_l$, where η is the learning rate and λ_{WM} is a weight decay constant, corresponds to gradient descent on the layer-wise regularization function

$$\mathcal{R}_{\mathrm{WM}} = \sum_{l \in \mathrm{layers}} \alpha \mathcal{P}_l^{\mathrm{amp}} + \beta \mathcal{P}_l^{\mathrm{decay}},$$

for $\alpha = 1$ and $\beta = \frac{\lambda_{\mathrm{WM}}}{\eta}$.

---

[2]We explore the consequences of this interpretation analytically in Appendix D.2.

	Alignment	$\mathcal{P}^{\text{decay}}$	$\mathcal{P}^{\text{amp}}$	$\mathcal{P}^{\text{null}}$	$\mathcal{P}^{\text{sparse}}$	$\mathcal{P}^{\text{self}}$
Local	Feedback Weight Mirror	✓	✓			
	Information	✓	✓	✓		
Non-Local	Symmetric	✓				✓
	Activation		✓		✓	

Table 2. **Taxonomy of learning rules** based on the locality and composition of their primitives.

If we consider primitives that are functions of the pseudogradients $\widetilde{\nabla}_l$ and $\widetilde{\nabla}_{l+1}$, then the **Kolen-Pollack (KP)** algorithm, originally proposed by Kolen & Pollack (1994) and modified by Akrout et al. (2019), can be understood in this framework as well. See Appendix D.3 for more details.

The range of primitives also allows for learning rules not yet investigated in the literature. In this work, we introduce several such novel learning rules, including Information Alignment (IA), Symmetric Alignment (SA), and Activation Alignment (AA). Each of these strategies is defined by a layer-wise regularization function composed from a linear combination of the primitives (Table 2). Information Alignment is a purely local rule, but unlike feedback alignment or weight mirror, contains the additional null primitive. In §4, we motivate this addition theoretically, and show empirically that it helps make IA a higher-performing and substantially more stable learning rule than previous local strategies. SA and AA are both non-local, but as shown in §5 perform even more robustly than any local strategy we or others have found, and may be implementable by a type of plausible biological mechanism we call "weight estimation."

3.3. Evaluating Learning Rules

For all the learning rules, we evaluate two desirable target metrics.

Task Performance. Performance-optimized CNNs on ImageNet provide the most effective quantitative description of neural responses of cortical neurons throughout the primate ventral visual pathway (Yamins et al., 2014; Cadena et al., 2019), indicating the biological relevance of task performance. Therefore, our first desired target will be ImageNet top-1 validation accuracy, in line with Bartunov et al. (2018).

Metaparameter Robustness. Extending the proposal of Bartunov et al. (2018), we also consider whether a proposed learning rule's metaparameters, such as learning rate and batch size, transfer across architectures. Specifically, when we optimize for metaparameters on a given architecture (e.g. ResNet-18), we will fix these metaparameters and use them to train both deeper (e.g. ResNet-50) and different variants (e.g. ResNet-v2). Therefore, our second desired target will be ImageNet top-1 validation accuracy *across* models for *fixed* metaparameters.

4. Local Learning Rules

Instability of Weight Mirror. Akrout et al. (2019) report that the weight mirror update rule matches the performance of backpropagation on ImageNet categorization. The procedure described in Akrout et al. (2019) involves not just the weight mirror rule, but a number of important additional training details, including alternating learning modes and using layer-wise Gaussian input noise. After reimplementing this procedure in detail, and using their prescribed metaparameters for the ResNet-18 architecture, the best top-1 validation accuracy we were able to obtain was 63.5% ($\mathcal{R}_{\text{WM}}$ in Table 3), substantially below the reported performance of 69.73%. To try to account for this discrepancy, we considered the possibility that the metaparameters were incorrectly set. We thus performed a large-scale metaparameter search over the continuous α, β, and the standard deviation σ of the Gaussian input noise, jointly optimizing these parameters for ImageNet validation set performance using a Bayesian Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011). After considering 824 distinct settings (see Appendix B.1 for further details), the optimal setting achieved a top-1 performance of 64.07% ($\mathcal{R}_{\text{WM}}^{\text{TPE}}$ in Table 3), still substantially below the reported performance in Akrout et al. (2019).

Considering the second metric of robustness, we found that the WM learning rule is very sensitive to metaparameter tuning. Specifically, when using either the metaparameters prescribed for ResNet-18 in Akrout et al. (2019) or those from our metaparameter search, directly attempting to train other network architectures failed entirely (Fig. 3, brown line).

Why is weight mirror under-performing backpropagation on both performance and robustness metrics? Intuition can be gained by simply considering the functional form of $\mathcal{R}_{\text{WM}}$, which can become *arbitrarily* negative even for fixed values of the forward weights. $\mathcal{R}_{\text{WM}}$ is a combination of a primitive which depends on the input ($\mathcal{P}^{\text{amp}}$) and a primitive which is independent of the input ($\mathcal{P}^{\text{decay}}$). Because of this, the primitives of weight mirror and their gradients may operate at different scales and careful metaparameter tuning must be done to balance their effects. This instability can be made precise by considering the dynamical system given by the symmetrized gradient flow on $\mathcal{R}_{\text{WM}}$ at a given layer l.

In the following analysis we ignore non-linearities, include weight decay on the forward weights, set $\alpha = \beta$, and consider the gradient with respect to both the forward and backward weights. When the weights, w_l and b_l, and input, x_l, are all scalar values, the gradient flow gives rise to the dynamical system

$$\frac{\partial}{\partial t}\begin{bmatrix} w_l \\ b_l \end{bmatrix} = -A \begin{bmatrix} w_l \\ b_l \end{bmatrix}, \tag{1}$$
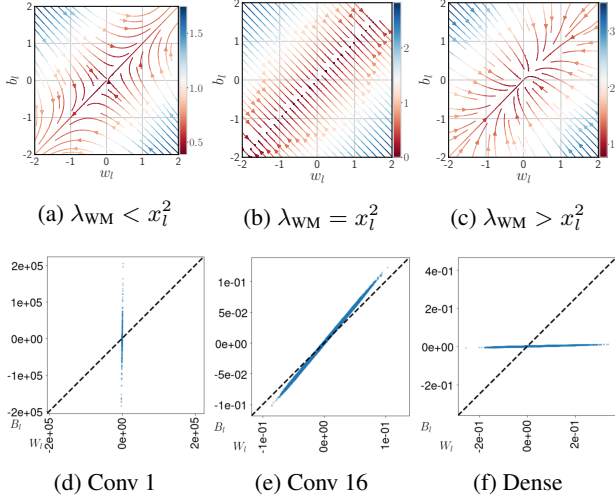
(a) $\lambda_{\mathrm{WM}} < x_l^2$ (b) $\lambda_{\mathrm{WM}} = x_l^2$ (c) $\lambda_{\mathrm{WM}} > x_l^2$

(d) Conv 1   (e) Conv 16   (f) Dense

Figure 2. **Unstable dynamics (a-c).** Symmetrized gradient flow on $\mathcal{R}_{\mathrm{WM}}$ at layer l with scalar weights and $x_l = 1$. The color and arrow indicate respectively the magnitude and direction of the flow. **Empirical instability (d-f).** Weight scatter plots for the first convolution, an intermediate convolution and the final dense layer of a ResNet-18 model trained with weight mirror for five epochs. Each dot represents an element in layer l's weight matrix and its (x, y) location corresponds to its forward and backward weight values, $(W_l^{(i,j)}, B_l^{(j,i)})$. The dotted diagonal line shows perfect weight symmetry, as is the case in backpropagation. Different layers demonstrate one of the the three dynamics outlined by the gradient flow analysis in §4: diverging, stable, and collapsing backward weights.

where A is an indefinite matrix (see Appendix D.1 for details.) A can be diagonally decomposed by the eigenbasis $\{u, v\}$, where u spans the symmetric component and v spans the skew-symmetric component of any realization of the weight vector $\begin{bmatrix} w_l & b_l \end{bmatrix}^{\mathsf{T}}$. Under this basis, the dynamical system decouples into a system of ODEs governed by the eigenvalues of A. The eigenvalue associated with the skew-symmetric eigenvector v is strictly positive, implying that this component decays exponentially to zero. However, for the symmetric eigenvector u, the sign of the corresponding eigenvalue depends on the relationship between λ_{WM} and x_l^2. When $\lambda_{\mathrm{WM}} > x_l^2$, the eigenvalue is positive and the symmetric component decays to zero (i.e. too much regularization). When $\lambda_{\mathrm{WM}} < x_l^2$, the eigenvalue is negative and the symmetric component exponentially grows (i.e. too little regularization). Only when $\lambda_{\mathrm{WM}} = x_l^2$ is the eigenvalue zero and the symmetric component stable. These various dynamics are shown in Fig. 2.

This analysis suggests that the sensitivity of weight mirror is not due to the misalignment of the forward and backward weights, but rather due to the stability of the symmetric component throughout training. Empirically, we find that this is true. In Fig. 2, we show a scatter plot of the backward and forward weights at three layers of a ResNet-18 model

trained with weight mirror. At each layer there exists a linear relationship between the weights, suggesting that the backward weights have aligned to the forward weights up to magnitude. Despite being initialized with similar magnitudes, at the first layer the backward weights have grown orders larger, at the last layer the backward weights have decayed orders smaller, and at only one intermediate layer were the backward weights comparable to the forward weights, implying symmetry.

This analysis also clarifies the stabilizing role of Gaussian noise, which was found to be essential to weight mirror's performance gains over feedback alignment (Akrout et al., 2019). Specifically, when the layer input $x_l \sim N\left(0, \sigma^2\right)$ and $\sigma^2 = \lambda_{\mathrm{WM}}$, then $x_l^2 \approx \lambda_{\mathrm{WM}}$, implying the dynamical system in equation (1) is stable.

Strategies for Reducing Instability. Given the above analysis, can we identify further strategies for reducing instability during learning beyond the use of Gaussian noise?

Adaptive Optimization. One option is to use an adaptive learning rule strategy, such as Adam (Kingma & Ba, 2014). An adaptive learning rate keeps an exponentially decaying moving average of past gradients, allowing for more effective optimization of the alignment regularizer even in the presence of exploding or vanishing gradients.

Local Stabilizing Operations. A second option to improve stability is to consider local layer-wise operations to the backward path such as choice of non-linear activation functions, batch centering, feature centering, and feature normalization. The use of these operations is largely inspired by Batch Normalization (Ioffe & Szegedy, 2015) and Layer Normalization (Ba et al., 2016) which have been observed to stabilize learning dynamics. The primary improvement that these normalizations allow for is the further conditioning of the covariance matrix at each layer's input, building on the benefits of using Gaussian noise. In order to keep the learning rule fully local, we use these normalizations, which unlike Batch and Layer Normalization, do not add any additional learnable parameters.

The Information Alignment (IA) Learning Rule. There is a third option for improving stability that involves modifying the local learning rule itself.

Without decay, the update given by weight mirror, $\Delta B_l = \eta x_l x_{l+1}^{\mathsf{T}}$, is Hebbian. This update, like all purely Hebbian learning rules, is unstable and can result in the norm of B_l diverging. This can be mitigated by weight decay, as is done in Akrout et al. (2019). However, an alternative strategy to dealing with the instability of a Hebbian update was given by Oja (1982) in his analysis of learning rules for linear neuron models. In the spirit of that analysis, assume that we can normalize the backward weight after each Hebbian

update such that

$$B_l^{(t+1)} = \frac{B_l^{(t)} + \eta x_l x_{l+1}^\mathsf{T}}{||B_l^{(t)} + \eta x_l x_{l+1}^\mathsf{T}||},$$

and in particular $||B_l^{(t)}|| = 1$ at all time t. Then, for small learning rates η, the right side can be expanded as a power series in η, such that

$$B_l^{(t+1)} = B_l^{(t)} + \eta \left(x_l x_{l+1}^\mathsf{T} - B_l^{(t)} x_l^\mathsf{T} B_l^{(t)} x_{l+1} \right) + O(\eta^2).$$

Ignoring the $O(\eta^2)$ term gives the non-linear update

$$\Delta B_l = \eta \left(x_l x_{l+1}^\mathsf{T} - B_l x_l^\mathsf{T} B_l x_{l+1} \right).$$

If we assume $x_l^\mathsf{T} B_l = x_{l+1}$ and B_l is a column vector rather than a matrix, then by Table 1, this is approximately the update given by the null primitive introduced in §3.1.

Thus motivated, we define **Information Alignment (IA)** as the local learning rule defined by adding a (weighted) null primitive to the other two local primitives already present in the weight mirror rule. That is, the layer-wise regularization function

$$\mathcal{R}_{\text{IA}} = \sum_{l \in \text{layers}} \alpha \mathcal{P}_l^{\text{amp}} + \beta \mathcal{P}_l^{\text{decay}} + \gamma \mathcal{P}_l^{\text{null}}.$$

In the specific setting when $x_{l+1} = W_l x_l$ and $\alpha = \gamma$, then the gradient of $\mathcal{R}_{\text{IA}}$ is proportional to the gradient with respect to B_l of $\frac{1}{2}||x_l - B_l W_l x_l||^2 + \frac{\beta}{2} \left(||W_l||^2 + ||B_l||^2 \right)$, a quadratically regularized linear autoencoder[3]. As shown in Kunin et al. (2019), all critical points of a quadratically regularized linear autoencoder attain symmetry of the encoder and decoder.

Empirical Results. To evaluate the three strategies for stabilizing local weight updates, we performed a neural architecture search implementing all three strategies, again using TPE. This search optimized for Top-1 ImageNet validation performance with the ResNet-18 architecture, comprising a total of 628 distinct settings. We found that validation performance increased significantly, with the optimal learning rule $\mathcal{R}_{\text{IA}}^{\text{TPE}}$, attaining 67.93% top-1 accuracy (Table 3). More importantly, we also found that the parameter robustness of $\mathcal{R}_{\text{IA}}^{\text{TPE}}$ is dramatically improved as compared to weight mirror (Fig. 3, orange line), nearly equaling the parameter robustness of backpropagation across a variety of deeper architectures. Critically, this improvement was achieved not by directly optimizing for robustness across architectures, but simply by finding a parameter setting that achieved high task performance on one architecture.

---

[3] In this setting, the resulting learning rule is a member of the target propagation framework introduced in §2.

Learning Rule	Top-1 Val Accuracy	Top-5 Val Accuracy
$\mathcal{R}_{\text{WM}}$	63.5%	85.16%
$\mathcal{R}_{\text{WM}}^{\text{TPE}}$	64.07%	85.47%
$\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$	64.40%	85.53%
$\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$	63.41%	84.83%
$\mathcal{R}_{\text{IA}}^{\text{TPE}}$	**67.93%**	**88.09%**
Backprop.	70.06%	89.14%

Table 3. **Performance of local learning rules with ResNet-18 on ImageNet.** $\mathcal{R}_{\text{WM}}$ is weight mirror as described in Akrout et al. (2019), $\mathcal{R}_{\text{WM}}^{\text{TPE}}$ is weight mirror with learning metaparameters chosen through an optimization procedure. $\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$ is weight mirror with an adaptive optimizer. $\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$ involves the addition of stabilizing operations to the network architecture. The best local learning rule, $\mathcal{R}_{\text{IA}}^{\text{TPE}}$, additionally involves the null primitive. For details on metaparameters for each local rule, see Appendix B.1.

To assess the importance of each strategy type in achieving this result, we also performed several ablation studies, involving neural architecture searches using only various subsets of the stabilization strategies (see Appendix B.1.3 for details). Using just the adaptive optimizer while otherwise optimizing the weight mirror metaparameters yielded the learning rule $\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$, while adding stabilizing layer-wise operations yielded the learning rule $\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$ (Table 3). We found that while the top-1 performance of these ablated learning rules was not better for the ResNet-18 architecture than the weight-mirror baseline, each of the strategies did individually contribute significantly to improved parameter robustness (Fig. 3, red and green lines).

Taken together, these results indicate that the regularization framework allows the formulation of local learning rules with substantially improved performance and, especially, metaparameter robustness characteristics. Moreover, these improvements are well-motivated by mathematical analysis that indicates how to target better circuit structure via improved learning stability.

5. Non-Local Learning Rules

While our best local learning rule is substantially improved as compared to previous alternatives, it still does not quite match backpropagation, either in terms of performance or metaparameter stability over widely different architectures (see the gap between blue and orange lines in Fig. 3). We next introduce two novel non-local learning rules that entirely eliminate this gap.

Symmetric Alignment (SA) is defined by the layer-wise regularization function

$$\mathcal{R}_{\text{SA}} = \sum_{l \in \text{layers}} \alpha \mathcal{P}_l^{\text{self}} + \beta \mathcal{P}_l^{\text{decay}}.$$
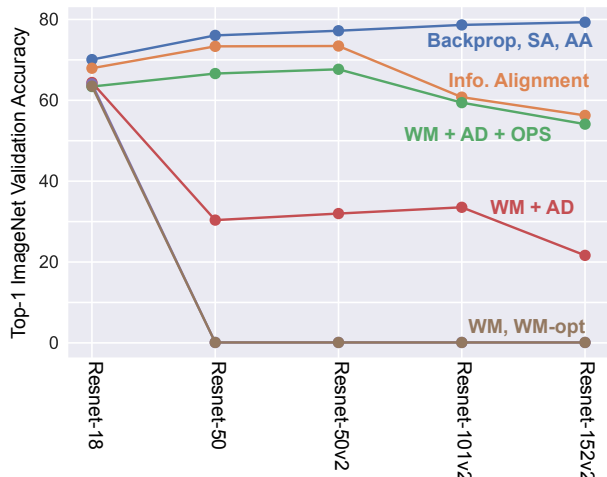
Figure 3. **Performance of local and non-local rules across architectures.** We fixed the categorical and continuous metaparameters for ResNet-18 and applied them directly to deeper and different ResNet variants (e.g. v2). A performance of 0.001 indicates the alignment loss became NaN within the first thousand steps of training. Our local rule Information Alignment (IA) consistently outperforms the other *local* alternatives across architectures, despite not being optimized for these architectures. The non-local rules, Symmetric Alignment (SA) and Activation Alignment (AA), consistently perform as well as backpropagation.

When $\alpha = \beta$, then the gradient of $\mathcal{R}_{\mathrm{SA}}$ is proportional to the gradient with respect to B_l of $\frac{1}{2}||W_l - B_l^{\mathsf{T}}||^2$, which encourages symmetry of the weights.

Activation Alignment (AA) is defined by the layer-wise regularization function

$$\mathcal{R}_{\mathrm{AA}} = \sum_{l \in \mathrm{layers}} \alpha \mathcal{P}_l^{\mathrm{amp}} + \beta \mathcal{P}_l^{\mathrm{sparse}}.$$

When $x_{l+1} = W_l x_l$ and $\alpha = \beta$, then the gradient of $\mathcal{R}_{\mathrm{AA}}$ is proportional to the gradient with respect to B_l of $\frac{1}{2}||W_l x_l - B_l^{\mathsf{T}} x_l||^2$, which encourages alignment of the activations.

Both SA and AA give rise to dynamics that encourage the backward weights to become transposes of their forward counterparts. When B_l is the transpose of W_l for all layers l then the updates generated by the backward pass are the exact gradients of $\mathcal{J}$. It follows intuitively that throughout training the pseudogradients given by these learning rules might converge to better approximations of the exact gradients of $\mathcal{J}$, leading to improved learning. Further, in the context of the analysis in equation (1), the matrix A associated with SA and AA is positive semi-definite, and unlike the case of weight mirror, the eigenvalue associated with the symmetric eigenvector u is zero, implying stability of the symmetric component.

While weight mirror and Information Alignment introduce dynamics that implicitly encourage symmetry of the forward

Model	Backprop.	Symmetric	Activation
ResNet-18	70.06%	69.84%	69.98%
ResNet-50	76.05%	76.29%	75.75%
ResNet-50v2	77.21%	77.18%	76.67%
ResNet-101v2	78.64%	78.74%	78.35%
ResNet-152v2	79.31%	79.15%	78.98%

Table 4. **Symmetric and Activation Alignment consistently match backpropagation.** Top-1 validation accuracies on ImageNet for each model class and non-local learning rule, compared to backpropagation.

and backward weights, the dynamics introduced by SA and AA encourage this property explicitly.

Despite not having the desirable locality property, we show that SA and AA perform well empirically in the weight-decoupled regularization framework — meaning that they *do* relieve the need for exact weight symmetry. As we will discuss, this may make it possible to find plausible biophysical mechanisms by which they might be implemented.

Parameter Robustness of Non-Local Learning Rules. To assess the robustness of SA and AA, we trained ResNet-18 models with standard 224-sized ImageNet images (training details can be found in Appendix B.2). Without any metaparameter tuning, SA and AA were able to match backpropagation in performance. Importantly, for SA we did not need to employ any specialized or adaptive learning schedule involving alternating modes of learning, as was required for all the local rules. However, for AA we did find it necessary to use an adaptive optimizer when minimizing $\mathcal{R}_{\mathrm{AA}}$, potentially due to the fact that it appears to align less exactly than SA (see Fig. S3). We trained deeper ResNet-50, 101, and 152 models (He et al., 2016) with larger 299-sized ImageNet images. As can be seen in Table 4, both SA and AA maintain consistent performance with backpropagation despite changes in image size and increasing depth of network architecture, demonstrating their robustness as a credit assignment strategies.

Weight Estimation, Neural Plausibility, and Noise Robustness. Though SA is non-local, it does avoid the need for instantaneous weight transport — as is shown simply by the fact that it optimizes effectively in the framework of decoupled forward-backward weight updates, where alignment can only arise over time due to the structure of the regularization circuit rather than instantaneously by *fiat* at each timepoint. Because of this key difference, it may be possible to find plausible biological implementations for SA, operating on a principle of iterative "weight estimation" in place of the implausible idea of instantaneous weight transport.

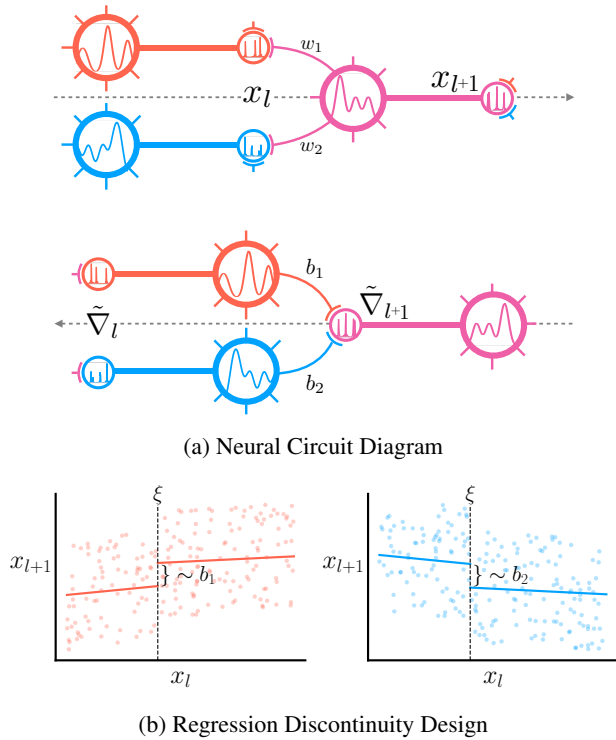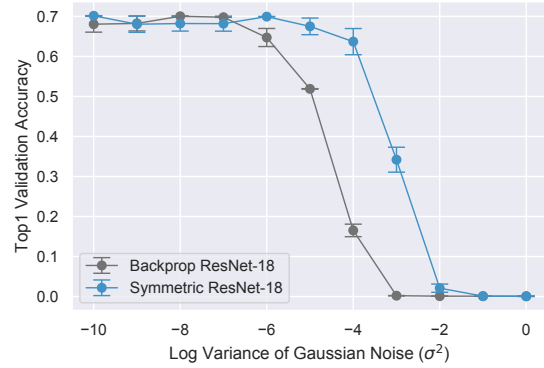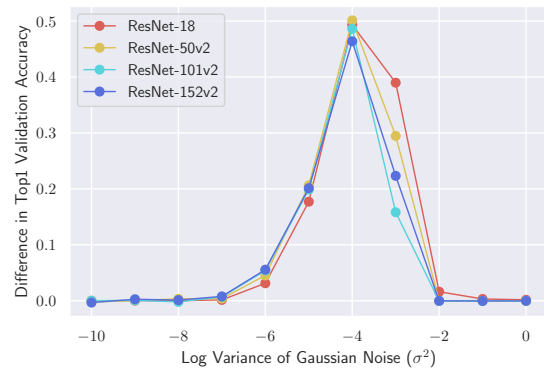By "weight estimation" we mean any process that can mea-

(a) Neural Circuit Diagram

(b) Regression Discontinuity Design

Figure 4. **Weight estimation.** (a) The notational diagram, as shown in Fig. 1, is mapped into a neural circuit. The top and bottom circuits represent the forward and backward paths respectively. Large circles represent the soma of a neuron, thick edges the axon, small circles the axon terminal, and thin edges the dendrites. Dendrites corresponding to the lateral pathways between the circuits are omitted. (b) A mechanism such as regression discontinuity design, as explained by Lansdell & Kording (2019) and Guerguiev et al. (2019), could be used independently at each neuron to do weight estimation by quantifying the causal effect of x_l on x_{l+1}.

sure changes in post-synaptic activity relative to varying synaptic input, thereby providing a temporal estimate of the synaptic strengths. Prior work has shown how noisy perturbations in the presence of spiking discontinuities (Lansdell & Kording, 2019) could provide neural mechanisms for weight estimation, as depicted in Fig. 4. In particular, Guerguiev et al. (2019) present a spiking-level mechanism for estimating forward weights from noisy dendritic measurements of the implied effect of those weights on activation changes. This idea, borrowed from the econometrics literature, is known as regression discontinuity design (Imbens & Lemieux, 2008). This is essentially a form of iterative weight estimation, and is used in Guerguiev et al. (2019) for minimizing a term that is mathematically equivalent to $\mathcal{P}^{\text{self}}$. Guerguiev et al. (2019) demonstrate that this weight estimation mechanism works empirically for small-scale networks.

Our performance and robustness results above for SA can

(a) ResNet-18

(b) Deeper Models

Figure 5. **Symmetric Alignment is more robust to noisy updates than backpropagation.** (a) Symmetric Alignment is more robust than backpropagation to increasing levels of Gaussian noise added to its updates for ResNet-18. (b) Symmetric Alignment maintains this robustness for deeper models. See Appendix B.3 for more details and similar experiments with Activation Alignment.

be interpreted as providing evidence that a rate-coded version of weight estimation scales effectively to training deep networks on large-scale tasks. However, there remains a gap between what we have shown at the rate-code level and the spike level, at which the weight estimation mechanism operates. Truly showing that weight estimation could work at scale would involve being able to train deep spiking neural networks, an unsolved problem that is beyond the scope of this work. One key difference between any weight estimation process at the rate-code and spike levels is that the latter will be inherently noisier due to statistical fluctuations in whatever local measurement process is invoked — e.g. in the Guerguiev et al. (2019) mechanism, the noise in computing the regression discontinuity.

As a proxy to better determine if our conclusions about the scalable robustness of rate-coded SA are likely to apply to spiking-level equivalents, we model this uncertainty by adding Gaussian noise to the backward updates during learning. To the extent that rate-coded SA is robust to such noise,
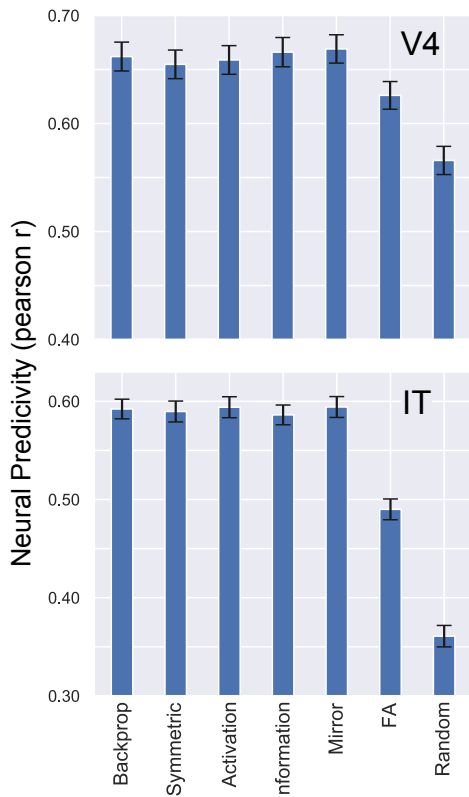
Figure 6. **Neural fits to temporally-averaged V4 and IT responses.** Neural fits to V4 (top) and IT (bottom) time-averaged responses (Majaj et al., 2015), using a 25 component PLS mapping on a ResNet-18. The median (across neurons) Pearson correlation over images, with standard error of mean (across neurons) denoting the error bars. "Random" refers to a ResNet-18 architecture at initialization. For details, see Appendix B.5.

the more likely it is that a spiking-based implementation will have the performance and parameter robustness characteristics of the rate-coded version. Specifically, we modify the update rule as follows:

$$\Delta\theta_b \propto \nabla\mathcal{R} + \mathcal{N}(0, \sigma^2), \qquad \Delta\theta_f \propto \widetilde{\nabla}\mathcal{J}.$$

As shown in Fig. 5, the performance of SA is very robust to noisy updates for training ResNet-18. In fact, for comparison we also train backpropagation with Gaussian noise added to its gradients, $\Delta\theta \propto \nabla\mathcal{J} + \mathcal{N}(0, \sigma^2)$, and find that SA is substantially *more* robust than backpropagation. For deeper models, SA maintains this robustness, implying that pseudogradients generated by backward weights with noisy updates leads to more robust learning than using equivalently noisy gradients directly.

6. Discussion

In this work, we present a unifying framework that allows for the systematic identification of robust and scalable alternatives to backpropagation. We obtain, through large-

scale searches, a local learning rule that transfers more robustly across architectures than previous local alternatives. Nonetheless, a performance and robustness gap persists with backpropagation. We formulate non-local learning rules that achieve competitive performance with backpropagation, requiring almost no metaparameter tuning and are robust to noisy updates. Taken together, our findings suggest that there are two routes towards the discovery of robust, scalable, and neurally plausible credit assignment without weight symmetry.

The first route involves further improving local rules. We found that the local operations and regularization primitives that allow for improved approximation of non-local rules perform better and are much more stable. If the analyses that inspired this improvement could be refined, perhaps further stability could be obtained. To aid in this exploration going forward, we have written an open-source TensorFlow library[4], allowing others to train arbitrary network architectures and learning rules at scale, distributed across multiple GPU or TPU workers. The second route involves the further refinement and characterization of scalable biological implementations of weight estimation mechanisms for Symmetric or Activation Alignment, as Guerguiev et al. (2019) initiate.

Given these two routes towards neurally-plausible credit assignment without weight symmetry, how would we use neuroscience data to adjudicate between them? It would be convenient if functional response data in a "fully trained" adult animal showed a signature of the underlying learning rule, without having to directly measure synaptic weights during learning. Such data have been very effective in identifying good models of the primate ventral visual pathway (Majaj et al., 2015; Yamins et al., 2014). As an initial investigation of this idea, we compared the activation patterns generated by networks trained with each local and non-local learning rule explored here, to neural response data from several macaque visual cortical areas, using a regression procedure similar to that in Yamins et al. (2014). As shown in Fig. 6, we found that, with the exception of the very poorly performing feedback alignment rule, all the reasonably effective learning rules achieve similar V4 and IT neural response predictivity, and in fact match that of the network learned via backpropagation. Such a result suggests the interesting possibility that the functional response signatures in an already well-learned neural representation may be relatively independent of which learning rule created them. Perhaps unsurprisingly, the question of identifying the operation of learning rules in an *in vivo* neural circuit will likely require the deployment of more sophisticated neuroscience techniques.

---
[4]https://github.com/neuroailab/neural-alignment

Acknowledgements

References

Akrout, M., Wilson, C., Humphreys, P. C., Lillicrap, T., and Tweed, D. Deep Learning without Weight Transport. *arXiv:1904.05391 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1904.05391. arXiv: 1904.05391.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baldi, P., Sadowski, P., and Lu, Z. Learning in the machine: Random backpropagation and the deep learning channel. *Artificial Intelligence*, 260:1–35, 2018. doi: 10.1016/j. artint.2018.03.003.

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. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9368–9378. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/ 8148-assessing-the-scalability-of- biologically-motivated-deep-learning- algorithms-and-architectures.pdf.

Bengio, Y. How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. *ArXiv*, abs/1407.7906, 2014.

Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., and Wu, Y. STDP-Compatible Approximation of Backpropagation in an Energy-Based Model. *Neural Computation*, 29(3):555–577, 2017. doi: 10.1162/NECO\ _a_00934. URL https://doi.org/10.1162/ NECO_a_00934. PMID: 28095200.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554, 2011.

Buchlovsky, P., Budden, D., Grewe, D., Jones, C., Aslanides, J., Besse, F., Brock, A., Clark, A., Colmenarejo, S. G., Pope, A., et al. Tf-replicator: Distributed machine learning for researchers. *arXiv preprint arXiv:1902.00465*, 2019.

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., and Ecker, A. S. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897, 2019.

Crick, F. The recent excitement about neural networks. *Nature*, 337:129132, 1989. doi: 10.1038/337129a0.

Grossberg, S. Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, 11: 23–63, 1987.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. Towards deep learning with segregated dendrites. *eLife*, 6, 2017. doi: 10.7554/eLife.22901.

Guerguiev, J., Kording, K. P., and Richards, B. A. Spike-based causal inference for weight alignment. *arXiv:1910.01689 [cs, q-bio]*, October 2019. URL http://arxiv.org/abs/1910.01689. arXiv: 1910.01689.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90. ISSN: 1063-6919.

Imbens, G. W. and Lemieux, T. Regression discontinuity designs: A guide to practice. *Journal of econometrics*, 142(2):615–635, 2008.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolen, J. and Pollack, J. Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 3, pp. 1375 – 1380 vol.3, 07 1994. ISBN 0-7803-1901-X. doi: 10.1109/ICNN.1994.374486.

Kunin, D., Bloom, J., Goeva, A., and Seed, C. Loss Landscapes of Regularized Linear Autoencoders. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3560–3569, Long Beach, California, USA, 09– 15 Jun 2019. PMLR. URL http://proceedings. mlr.press/v97/kunin19a.html.

Lansdell, B. J. and Kording, K. P. Spiking allows neurons to estimate their causal effect. *bioRxiv*, pp. 253351, 2019.

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.

Liao, Q., Leibo, J. Z., and Poggio, T. How Important is Weight Symmetry in Backpropagation? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1837–1844. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3016100.3016156.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, November 2016. ISSN 2041-1723. doi: 10.1038/ncomms13276. URL https://www.nature.com/articles/ncomms13276.

Majaj, N. J., Hong, H., Solomon, E. A., and DiCarlo, J. J. Simple Learned Weighted Sums of Inferior Temporal Neuronal Firing Rates Accurately Predict Human Core Object Recognition Performance. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 35:13402–18, 2015 Sep 30 2015. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.5181-14.2015.

Moskovitz, T. H., Litwin-Kumar, A., and Abbott, L. F. Feedback alignment in deep convolutional networks. *CoRR*, abs/1812.06488, 2018. URL http://arxiv.org/abs/1812.06488.

Nøkland, A. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *Advances in neural information processing systems*, pp. 1037–1045, 2016.

Oja, E. A simplified neuron model as a principal component analyzer. *E. J. Math. Biology*, 15(3):267 – 273, 1982. doi: 10.1007/BF00275687.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323:533536, October 1986. doi: 10.1038/323533a0.

Sacramento, J. a., 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. URL http://papers.nips.cc/paper/8089-dendritic-cortical-microcircuits-approximate-the-backpropagation-algorithm.pdf.

Saxe, A., Mcclelland, J., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 12 2013.

Scellier, B. and Bengio, Y. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00024. URL https://www.frontiersin.org/article/10.3389/fncom.2017.00024.

Whittington, J. C. R. 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.

Xiao, W., Chen, H., Liao, Q., and Poggio, T. Biologically-Plausible Learning Algorithms Can Scale to Large Datasets. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SygvZ209F7.

Xie, X. and Seung, H. Equivalence of Backpropagation and Contrastive Hebbian Learning in a Layered Network. *Neural computation*, 15:441–54, 03 2003. doi: 10.1162/089976603762552988.

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.

A. Code base

In this section we describe our implementation and highlight the technical details that allow its generality for use in any architecture. We used TensorFlow version 1.13.1 to conduct all experiments, and adhered to its interface. All code can be found at https://github.com/neuroailab/neural-alignment.

A.1. Layers

The essential idea of our code base is that by implementing custom layers that match the TensorFlow API, but use custom operations for matrix multiplication (`matmul`) and two-dimensional convolutions (`conv2d`), then we can efficiently implement arbitrary feedforward networks using any credit assignment strategies with untied forward and backward weights. Our custom `matmul` and `conv2d` operations take in a forward and backward kernel. They use the forward kernel for the forward pass, but use the backward kernel when propagating the gradient. To implement this, we leverage the `@tf.custom_gradient` decorator, which allows us to explicitly define the forward and backward passes for that op. Our `Layer` objects implement custom dense and convolutional layers which use the custom operations described above. Both layers take in the same arguments as the native TensorFlow layers and an additional argument for the learning rule.

A.2. Alignments

A learning rule is defined by the form of the layer-wise regularization $\mathcal{R}$ added to the model at each layer. The custom layers take an instance of an alignment class which when called will define its alignment specific regularization and add it to the computational graph.

The learning rule are specializations of a parent `Alignment` object which implements a `__call__` method that creates the regularization function. The regularization function uses tensors that prevent the gradients from flowing to previous layers via `tf.stop_gradient`, keeping the alignment loss localized to a single layer. Implementation of the `__call__` method is delegated to subclasses, such that they can define their alignment specific regularization as a weighted sum of primitives, each of which is defined as a function.

A.3. Optimizers

The total network loss is defined as the sum of the global cost function $\mathcal{J}$ and the local alignment regularization $\mathcal{R}$. The optimizer class provides a framework for specifying how to optimize each part of the total network loss as a function of the global step.

In the `Optimizers` folder you will find two important files:

- `rate_scheduler.py` defines a scheduler which is a function of the global step, that allows you to adapt the components of the alignment weighting based on where it is in training. If you do not pass in a scheduling function, it will by default return a constant rate.

- `optimizers.py` provides a class which takes in a list of optimizers, as well as a list of losses to optimize. Each loss element is optimized with the corresponding optimizer at each step in training, allowing you to have potentially different learning rate schedules for different components of the loss. Minibatching is also supported.

B. Experimental Details

In what follows we describe the metaparameters we used to run each of the experiments reported above, tabulated in Table 5. Any defaults from TensorFlow correspond to those in version 1.13.1.

B.1. TPE search spaces

We detail the search spaces for each of the searches performed in §4. For each search, we trained approximately 60 distinct settings at a time using the HyperOpt package (Bergstra et al., 2011) using the ResNet-18 architecture and L2 weight decay of $\lambda = 10^{-4}$ (He et al., 2016) for 45 epochs, corresponding to the point in training midway between the first and second learning rate drops. Each model was trained on its own Tensor Processing Unit (TPUv2-8 and TPUv3-8).

We employed a form of Bayesian optimization, a Tree-structured Parzen Estimator (TPE), to search the space of continuous and categorical metaparameters (Bergstra et al., 2011). This algorithm constructs a generative model of $P[score \mid configuration]$ by updating a prior from a maintained history H of metaparameter configuration-loss pairs. The fitness function that is optimized over models is the expected improvement, where a given configuration c is meant to optimize $EI(c) = \int_{x<t} P[x \mid c, H]$. This choice of Bayesian optimization algorithm models $P[c \mid x]$ via a Gaussian mixture, and restricts us to tree-structured configuration spaces.

B.1.1. $\mathcal{R}_{\mathrm{WM}}^{\mathrm{TPE}}$ SEARCH SPACE

Below is a description of the metaparameters and their ranges for the search that gave rise to $\mathcal{R}_{\mathrm{WM}}^{\mathrm{TPE}}$ in Table 3.

- Gaussian input noise standard deviation $\sigma \in [10^{-10}, 1]$ used in the backward pass, sampled uniformly.

	$\mathcal{R}_{\text{WM}}^{\text{TPE}}$	$\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$	$\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$	$\mathcal{R}_{\text{IA}}^{\text{TPE}}$
Alternating Minimization	True	True	True	True
Delay Epochs (de)	2	0	0	1
Train Batch Size ($|\mathcal{B}|$)	2048	256	256	256
SGDM Learning Rate	1.0	0.125	0.125	0.125
Alignment Optimizer	Vanilla GD	Adam	Adam	Adam
Alignment Learning Rate (η)	1.0	0.0053	0.0025	0.0098
σ	0.6905	0.9500	0.6402	0.8176
α/β	15.6607	13.9040	0.1344	129.1123
β	0.0283	2.8109×10^{-8}	232.7856	7.9990
γ	N/A	N/A	N/A	3.1610×10^{-6}
Forward Path Output (FO) Bias	True	True	True	True
FO ReLU	True	True	True	True
FO BWMC	True	True	True	True
FO FWMC	False	False	True	True
FO FWL2N	False	False	False	False
Backward Path Output (BO) Bias	False	False	False	True
BO ReLU	False	False	True	False
BO FWMC	False	False	False	True
BO FWL2N	False	False	True	True
Backward Path Input (BI) BWMC	True	True	True	False
BI FWMC	False	False	False	False
BI FWL2N	False	False	False	False

Table 5. Metaparameter settings (rows) for each of the learning rules obtained by large-scale searches (columns). Continuous values were rounded up to 4 decimal places.

- Ratio between the weighting of $\mathcal{P}^{\text{amp}}$ and $\mathcal{P}^{\text{decay}}$ given by $\alpha/\beta \in [0.1, 200]$, sampled uniformly.

- The weighting of $\mathcal{P}^{\text{decay}}$ given by $\beta \in [10^{-11}, 10^{7}]$, sampled log-uniformly.

We fix all other metaparameters as prescribed by Akrout et al. (2019), namely batch centering the backward path inputs and forward path outputs in the backward pass, as well as applying a ReLU activation function and bias to the forward path but not to the backward path in the backward pass. To keep the learning rule fully local, we do not allow for any transport during the mirroring phase of the batch normalization mean and standard deviation as Akrout et al. (2019) allow.

B.1.2. $\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$ SEARCH SPACE

Below is a description of the metaparameters and their ranges for the search that gave rise to $\mathcal{R}_{\text{WM+AD}}^{\text{TPE}}$ in Table 3.

- Train batch size $|\mathcal{B}| \in \{256, 1024, 2048, 4096\}$. This choice also determines the forward path Nesterov mo-

mentum learning rate on the *pseudogradient* of the categorization objective $\mathcal{J}$, as it is set to be $|\mathcal{B}|/2048$, and linearly warm it up to this value for 6 epochs followed by 90% decay at 30, 60, and 80 epochs, training for 100 epochs total, as prescribed by Buchlovsky et al. (2019).

- Alignment learning rate $\eta \in [10^{-6}, 10^{-2}]$, sampled log-uniformly. This parameter sets the adaptive learning rate on the Adam optimizer applied to the *gradient* of the alignment loss $\mathcal{R}$, and which will be dropped synchronously by 90% decay at 30, 60, and 80 epochs along with the Nesterov momentum learning rate on the *pseudogradient* of the categorization objective $\mathcal{J}$.

- Number of delay epochs $de \in \{0, 1, 2\}$ for which we delay optimization of the categorization objective $\mathcal{J}$ and solely optimize the alignment loss $\mathcal{R}$. If $de > 0$, we use the alignment learning rate η during this delay period and the learning rate drops are shifted by de epochs; otherwise, if $de = 0$, we linearly warmup η for 6 epochs as well.

- Whether or not to perform alternating minimization of $\mathcal{J}$ and $\mathcal{R}$ each step, or instead simultaneously optimize these objectives in a single training step.

The remaining metaparameters and their ranges were the same as those from Appendix B.1.1.

We fix the layer-wise operations as prescribed by Akrout et al. (2019), namely batch centering the backward path input and forward path outputs in the backward pass (**BI BWMC** and **FO BWMC**, respectively), as well as applying a ReLU activation function and bias to the forward path (**FO ReLU** and **FO Bias**, respectively) but not to the backward path in the backward pass (**BO ReLU** and **BO Bias**, respectively).

B.1.3. $\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$ SEARCH SPACE

Below is a description of the metaparameters and their ranges for the search that gave rise to $\mathcal{R}_{\text{WM+AD+OPS}}^{\text{TPE}}$ in Table 3. In this search, we expand the search space described in Appendix B.1.2 to include boolean choices over layer-wise operations performed in the *backward pass*, involving either the inputs, the forward path f_l (involving only the forward weights W_l), or the backward path b_l (involving only the backward weights B_l):

Use of biases in the forward and backward paths:

- **FO Bias:** Whether or not to use biases in the forward path.

- **BO Bias:** Whether or not to use biases in the backward path.

Use of nonlinearities in the forward and backward paths:

- **FO ReLU:** Whether or not to apply a ReLU to the forward path output.

- **BO ReLU:** Whether or not to apply a ReLU to the backward path output.

Centering and normalization operations in the forward and backward paths:

- **FO BWMC:** Whether or not to mean center (across the *batch* dimension) the forward path output $f_l = f_l - \bar{f}_l$.

- **BI BWMC:** Whether or not to mean center (across the *batch* dimension) the backward path *input*.

- **FO FWMC:** Whether or not to mean center (across the *feature* dimension) the forward path output $f_l = f_l - \hat{f}_l$.

- **BO FWMC:** Whether or not to mean center (across the *feature* dimension) the backward path output $b_l = b_l - \hat{b}_l$.

- **FO FWL2N:** Whether or not to L2 normalize (across the feature dimension) the forward path output $f_l = (f_l - \hat{f}_l)/\|f_l - \hat{f}_l\|_2$.

- **BO FWL2N:** Whether or not to L2 normalize (across the feature dimension) the backward path output $b_l = (b_l - \hat{b}_l)/\|b_l - \hat{b}_l\|_2$.

Centering and normalization operations applied to the inputs to the backward pass:

- **BI FWMC:** Whether or not to mean center (across the feature dimension) the backward pass input $x_l = x_l - \hat{x}_l$.

- **BI FWL2N:** Whether or not to L2 normalize (across the feature dimension) the backward pass input $x_l = (x_l - \hat{x}_l)/\|x_l - \hat{x}_l\|_2$.

The remaining metaparameters and their ranges were the same as those from Appendix B.1.2.

B.1.4. $\mathcal{R}_{\text{IA}}^{\text{TPE}}$ SEARCH SPACE

Below is a description of the metaparameters and their ranges for the search that gave rise to $\mathcal{R}_{\text{IA}}^{\text{TPE}}$ in Table 3. In this search, we expand the search space described in Appendix B.1.3, to now include the additional $\mathcal{P}^{\text{null}}$ primitive.

- The weighting of $\mathcal{P}^{\text{null}}$ given by $\gamma \in [10^{-11}, 10^7]$, sampled log-uniformly.

The remaining metaparameters and their ranges were the same as those from Appendix B.1.3.

B.2. Symmetric and Activation Alignment metaparameters

We now describe the metaparameters used to generate Table 4. We used a batch size of 256, forward path Nesterov with Momentum of 0.9 and a learning rate of 0.1 applied to the categorization objective $\mathcal{J}$, warmed up linearly for 5 epochs, with learning rate drops at 30, 60, and 80 epochs, trained for a total of 90 epochs, as prescribed by He et al. (2016).

For Symmetric and Activation Alignment ($\mathcal{R}_{\text{SA}}$ and $\mathcal{R}_{\text{AA}}$), we used Adam on the alignment loss $\mathcal{R}$ with a learning rate of 0.001, along with the following weightings for their primitives:

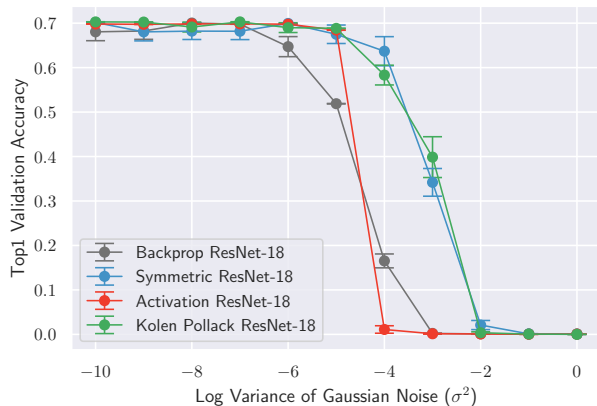- Symmetric Alignment: $\alpha = 10^{-3}, \beta = 2 \times 10^{-3}$

Figure S1. **Noisy updates.** Symmetric Alignment, Activation Alignment, and Kolen-Pollack are *more* robust to noisy updates than backpropagation for ResNet-18.

- Activation Alignment: $\alpha = 10^{-3}, \beta = 2 \times 10^{-3}$

We use biases in both the forward and backward paths of the backward pass, but do *not* employ a ReLU nonlinearity to either path.

B.3. Noisy updates

We describe the experimental setup and metaparameters used in §5 to generate Fig. 5.

Fig. 5a was generated by running 10 trials for each experiment configuration. The error bars show the standard error of the mean across trials.

For backpropagation we used a momentum optimizer with an initial learning rate of 0.1, standard batch size of 256, and learning rate drops at 30 and 60 epochs.

For Symmetric and Activation Alignment we used the same metaparameters as backpropagation for the categorization objective $\mathcal{J}$ and an Adam optimizer with an initial learning rate of 0.001 and learning rate drops at 30 and 60 epochs for the alignment loss $\mathcal{R}$. All other metaparameters were the same as described in Appendix B.2.

In all experiments we added the noise to the update given by the respective optimizers and scaled by the current learning rate, that way at learning rate drops the noise scaled appropriately. To account for the fact that the initial learning rate for the backpopagation experiments was 0.1, while for symmetric and activation experiments it was 0.001, we shifted the latter two curves by 10^4 to account for the effective difference in variance. See Fig. S1.
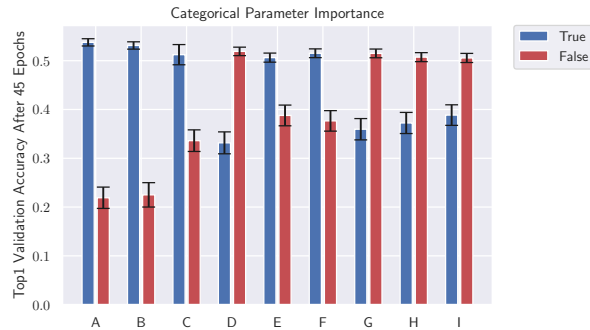

Figure S2. **Analysis of important categorical metaparameters of top performing local rule $\mathcal{R}_{\text{IA}}^{\text{TPE}}$.** Mean across models, and the error bars indicate SEM across models.

B.4. Metaparameter importance quantification

We include here the set of discrete metaparameters that mattered the most across hundreds of models in our large-scale search, sorted by most to least important, plotted in Fig. S2. Specifically, these amount to choices of activation, layer-wise normalization, input normalization, and Gaussian noise in the forward and backward paths of the backward pass. The detailed labeling is given as follows: **A:** Whether or not to L2 normalize (across the feature dimension) the backward path outputs in the backward pass. **B:** Whether to use Gaussian noise in the backward pass inputs. **C:** Whether to solely optimize the alignment loss in the first 1-2 epochs of training. **D, E:** Whether or not to apply a non-linearity in the backward or forward path outputs in the backward pass, respectively. **F:** Whether or not to apply a bias in the forward path outputs (pre-nonlinearity). **G, H:** Whether or not to mean center or L2 normalize (across the feature dimension) the inputs to the backward pass. **I:** Same as **A**, but instead applied to the forward path outputs in the backward pass.

B.5. Neural Fitting Procedure

We fit trained model features to multi-unit array responses from (Majaj et al., 2015). Briefly, we fit to 256 recorded sites from two monkeys. These came from three multi-unit arrays per monkey: one implanted in V4, one in posterior IT, and one in central and anterior IT. Each image was presented approximately 50 times, using rapid visual stimulus presentation (RSVP). Each stimulus was presented for 100 ms, followed by a mean gray background interleaved between images. Each trial lasted 250 ms. The image set consisted of 5120 images based on 64 object categories. Each image consisted of a 2D projection of a 3D model added to a random background. The pose, size, and x- and y-position of the object was varied across the image set, whereby 2 levels of variation were used (corresponding to medium and high variation from (Majaj et al., 2015).) Multi-unit responses

to these images were binned in 10ms windows, averaged across trials of the same image, and normalized to the average response to a blank image. They were then averaged 70-170 ms post-stimulus onset, producing a set of (5120 images x 256 units) responses, which were the targets for our model features to predict. The 5120 images were split 75-25 within each object category into a training set and a held-out testing set.

C. Visualizations

In this section we present some visualizations which deepen the understanding of the weight dynamics and stability during training, as presented in §4 and §5. By looking at the weights of the network at each validation point, we are able to compare corresponding forward and backward weights (see Fig. S3) as well as to measure the angle between the vectorized forward and backward weight matrices to quantify their degree of alignment (see Fig. S4). Their similarity in terms of scale can also be evaluated by looking at the ratio of the Frobenius norm of the backward weight matrix to the forward weight matrix, $\|B_l\|_F / \|W_l\|_F$. Separately plotting these metrics in terms of model depth sheds some insight into how different layers behave.

D. Further Analysis

D.1. Instability of Weight Mirror

As explained in §4, the instability of weight mirror can be understood by considering the dynamical system given by the symmetrized gradient flow on $\mathcal{R}_{\text{SA}}$, $\mathcal{R}_{\text{AA}}$, and $\mathcal{R}_{\text{WM}}$ at a given layer l. By symmetrized gradient flow we imply the gradient dynamics on the loss $\mathcal{R}$ modified such that it is symmetric in both the forward and backward weights. We ignore biases and non-linearities and set $\alpha = \beta$ for all three losses.

When the weights, w_l and b_l, and input, x_l, are all scalar values, the gradient flow for all three losses gives rise to the dynamical system,

$$\frac{\partial}{\partial t} \begin{bmatrix} w_l \\ b_l \end{bmatrix} = -A \begin{bmatrix} w_l \\ b_l \end{bmatrix},$$

For Symmetric Alignment and Activation Alignment, A is respectively the positive semidefinite matrix

$$A_{\text{SA}} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad A_{\text{AA}} = \begin{bmatrix} x_l^2 & -x_l^2 \\ -x_l^2 & x_l^2 \end{bmatrix}.$$

For weight mirror, A is the symmetric indefinite matrix

$$A_{\text{WM}} = \begin{bmatrix} \lambda_{\text{WM}} & -x_l^2 \\ -x_l^2 & \lambda_{\text{WM}} \end{bmatrix}.$$

In all three cases A can be diagonally decomposed by the

(a) SA Epoch 0   (b) SA Epoch 2   (c) SA Epoch 90

(d) AA Epoch 0   (e) AA Epoch 2   (f) AA Epoch 90

(g) WM Epoch 0   (h) WM Epoch 2   (i) WM Epoch 90

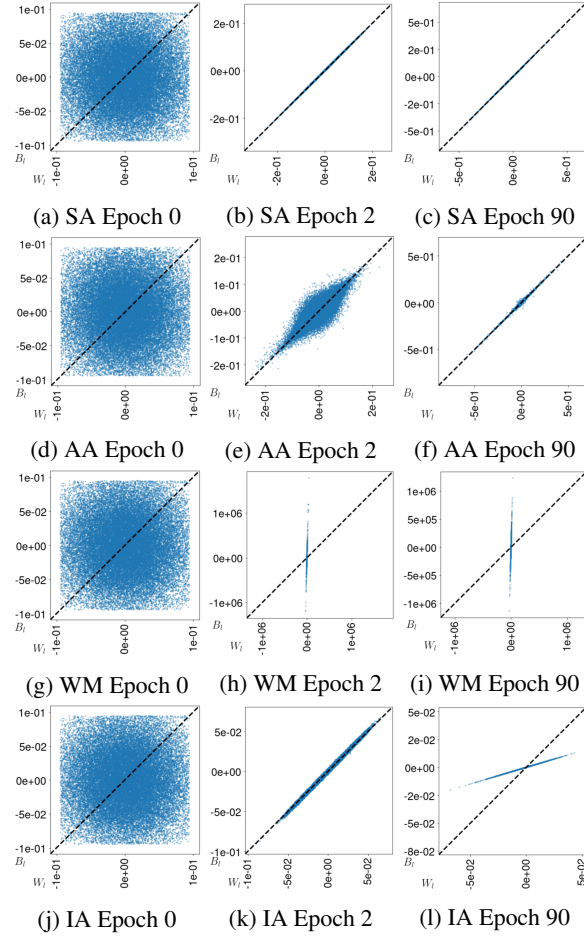(j) IA Epoch 0   (k) IA Epoch 2   (l) IA Epoch 90

Figure S3. **Learning symmetry.** Weight values of the third convolutional layer in ResNet-18 throughout training with various learning rules. Each dot represents an element in layer l's weight matrix and its (x, y) location corresponds to its forward and backward weight values, $(W_l^{(i,j)}, B_l^{(j,i)})$. The dotted diagonal line shows perfect weight symmetry, as is the case in backpropagation.

eigenbasis

$$\{u, v\} = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\},$$

where u spans the symmetric component and v spans the skew-symmetric component of any realization of the weight vector $\begin{bmatrix} w_l & b_l \end{bmatrix}^{\mathsf{T}}$.

As explained in §4, under this basis, the dynamical system decouples into a system of ODEs governed by the eigenvalues λ_u and λ_v associated with u and v. For all three learning rules, $\lambda_v > 0$ (λ_v is respectively 1, x^2, and $\lambda_{\text{WM}} + x_l^2$ for SA, AA, and weight mirror). For SA and AA, $\lambda_u = 0$, while for weight mirror $\lambda_u = \lambda_{\text{WM}} - x_l^2$.

(a) Symmetric Alignment

(b) Activation Alignment

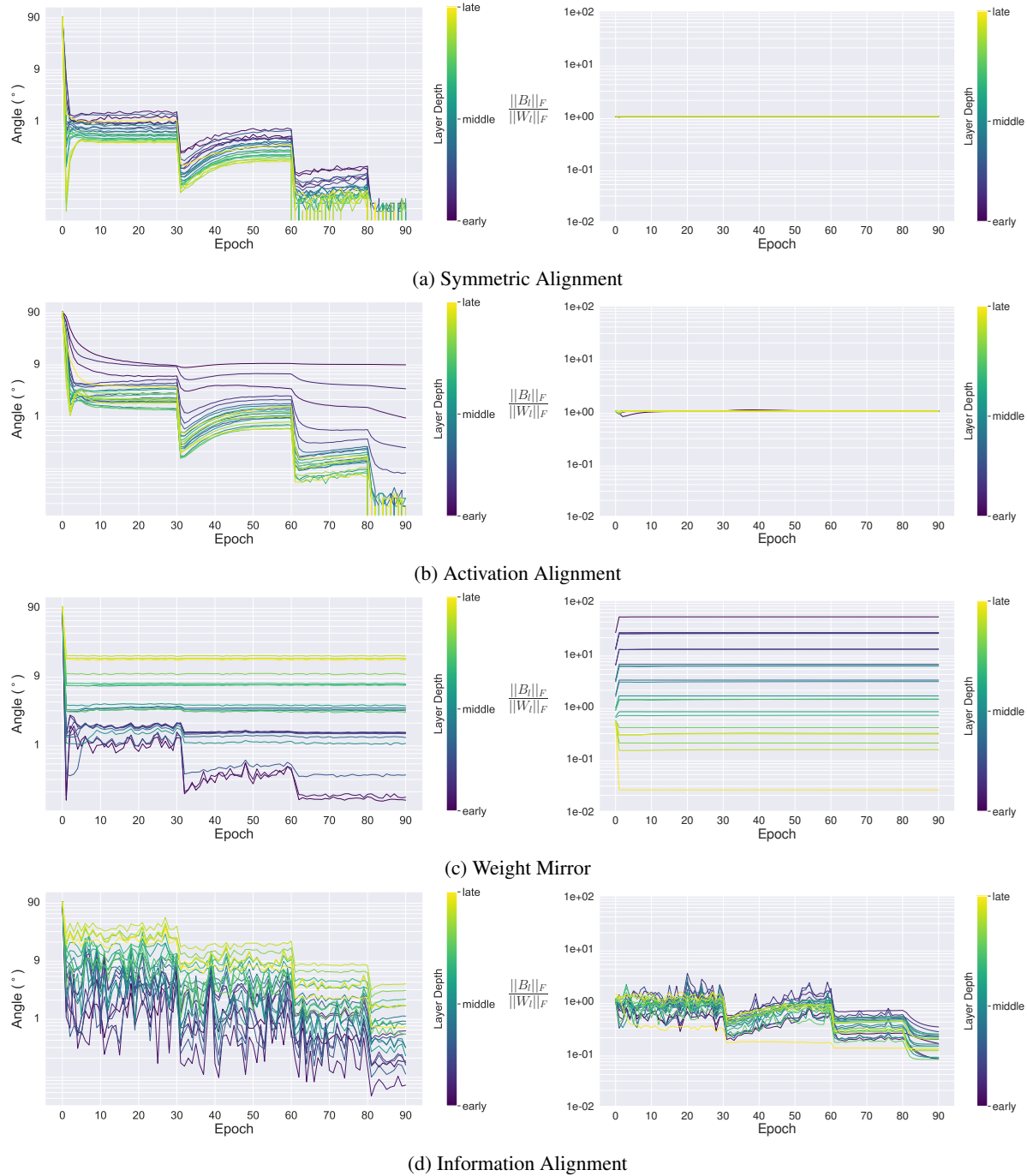(c) Weight Mirror

(d) Information Alignment

Figure S4. **Weight metrics during training.** Figures on the left column show the angle between the forward and the backward weights at each layer, depicting their degree of alignment. Figures on the right column show the ratio of the Frobenius norm of the backward weights to the forward weights during training. For Symmetric Alignment (a) we can clearly see how the weights align very early during training, with the learning rate drops allowing them to further decrease. Additionally, the sizes of forward and backwards weight also remain at the same scale during training. Activation Alignment (b) shows similar behavior to activation, though some of the earlier layers fail to align as closely as the Symmetric Alignment case. Weight Mirror (c) shows alignment happening within the first few epochs, though some of the later layers don't align as closely. Looking at the size of the weights during training, we can observe the unstable dynamics explained in §4 with exploding and collapsing weight values (Fig. 2) within the first few epochs of training. Information Alignment (d) shows a similar ordering in alignment as weight mirror, but overall alignment does improve throughout training, with all layers aligning within 5 degrees. Compared to weight mirror, the norms of the weights are more stable, with the backward weights becoming smaller than their forward counterparts towards the end of training.

D.2. Beyond Feedback Alignment

An underlying assumption of our work is that certain forms of layer-wise regularization, such as the regularization introduced by Symmetric Alignment, can actually improve the performance of feedback alignment by introducing dynamics on the backward weights. To understand these improvements, we build off of prior analyses of backpropagation (Saxe et al., 2013) and feedback alignment (Baldi et al., 2018).

Consider the simplest nontrivial architecture: a two layer scalar linear network with forward weights w_1, w_2, and backward weight b. The network is trained with scalar data $\{x_i, y_i\}_{i=1}^n$ on the mean squared error cost function

$$\mathcal{J} = \sum_{i=1}^n \frac{1}{2n}(y_i - w_2 w_1 x_i)^2.$$

The gradient flow of this network gives the coupled system of differential equations on (w_1, w_2, b)

$$\dot{w}_1 = b(\alpha - w_2 w_1 \beta) \tag{2}$$
$$\dot{w}_2 = w_1(\alpha - w_2 w_1 \beta) \tag{3}$$

where $\alpha = \sum_{i=1}^n \frac{y_i x_i}{n}$ and $\beta = \sum_{i=1}^n \frac{x_i^2}{n}$. For backpropagation the dynamics are constrained to the hyperplane $b = w_2$, while for feedback alignment the dynamics are contained on the hyperplane $b = b(0)$ given by the initialization. For Symmetric Alignment, an additional differential equation

$$\dot{b} = w_2 - b, \tag{4}$$

attracts all trajectories to the backpropagation hyperplane $b = w_2$.

To understand the properties of these alignment strategies, we explore the fixed points of their flow. From equation (2) and (3) we see that both equations are zero on the hyperbola

$$w_2 w_1 = \frac{\alpha}{\beta},$$

which is the set of minima of $\mathcal{J}$. From equation (4) we see that all fixed points of Symmetric Alignment satisfy $b = w_2$. Thus, all three alignment strategies have fixed points on the hyperbola of minima intersected with either the hyperplane $b = b(0)$ in the case of feedback alignment or $b = w_2$ in the case of backpropagation and Symmetric Alignment.

In addition to these non-zero fixed points, equation (2) and (3) are zero if b and w_1 are zero respectively. For backpropagation and Symmetric Alignment this also implies $w_2 = 0$, however for feedback alignment w_2 is free to be any value. Thus, all three alignment strategies have rank-deficient fixed points at the origin $(0, 0, 0)$ and in the case of feedback alignment more generally on the hyperplane $b = w_1 = 0$.

To understand the stability of these fixed points we consider the local linearization of the vector field by computing the Jacobian matrix[5]

$$J = \begin{bmatrix} \partial_{w_1} \dot{w}_1 & \partial_{w_1} \dot{w}_2 & \partial_{w_1} \dot{b} \\ \partial_{w_2} \dot{w}_1 & \partial_{w_2} \dot{w}_2 & \partial_{w_2} \dot{b} \\ \partial_b \dot{w}_1 & \partial_b \dot{w}_2 & \partial_b \dot{b} \end{bmatrix}.$$

A source of the gradient flow is characterized by non-positive eigenvalues of J, a sink by non-negative eigenvalues of J, and a saddle by both positive and negative eigenvalues of J.

On the hyperbola $w_2 w_1 = \frac{\alpha}{\beta}$ the Jacobian matrix for the three alignment strategies have the corresponding eigenvalues:

	λ_1	λ_2	λ_3
Backprop.	$-\left(w_1^2 + w_2^2\right)x^2$	0	
Feedback	$-\left(w_1^2 + bw_2\right)x^2$	0	0
Symmetric	$-\left(w_1^2 + w_2^2\right)x^2$	0	-1

Thus, for backpropagation and Symmetric Alignment, all minima of the the cost function $\mathcal{J}$ are sinks, while for feedback alignment the stability of the minima depends on the sign of $w_1^2 + bw_2$.

From this simple example there are two major takeaways:

1. All minima of the cost function $\mathcal{J}$ are sinks of the flow given by backpropagation and Symmetric Alignment, but only some minima are sinks of the flow given by feedback alignment.

2. Backpropagation and Symmetric Alignment have the exact same critical points, but feedback alignment has a much larger space of rank-deficient critical points.

Thus, even in this simple example it is clear that certain dynamics on the backward weights can have a stabilizing effect on feedback alignment.

D.3. Kolen-Pollack Learning Rule

If we consider primitives that are functions of the pseudo-gradients $\widetilde{\nabla}_l$ and $\widetilde{\nabla}_{l+1}$ in addition to the forward weight W_l, backward weight B_l, layer input x_l, and layer output x_{l+1}, then the Kolen-Pollack algorithm, originally proposed by Kolen & Pollack (1994) and modified by Akrout et al. (2019), can be understood in our framework.

The Kolen-Pollack algorithm circumvents the weight transport problem, by instead transporting the weight updates

---

[5]In the case that the vector field is the negative gradient of a loss, as in backpropagation, then this is the negative Hessian of the loss.
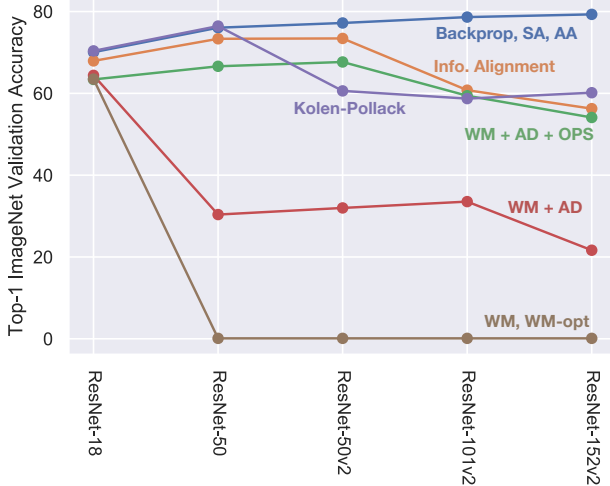
Figure S5. **Performance of Kolen-Pollack across architectures.** We fixed the categorical and continuous metaparameters for ResNet-18 and applied them directly to deeper and different ResNet variants (e.g. v2) as in Fig. 3. The Kolen-Pollack learning rule, matched backpropagation performance for ResNet-18 and ResNet-50, but a performance gap emerged for different (ResNet-50v2) and deeper (ResNet-101v2, ResNet-152v2) architectures.

and adding weight decay. Specifically, the forward and backward weights are updated respectively by

$$\Delta W_l = -\eta \widetilde{\nabla}_{l+1} x_l^\mathsf{T} - \lambda_{\text{KP}} W_l,$$
$$\Delta B_l = -\eta x_l \widetilde{\nabla}_{l+1}^\mathsf{T} - \lambda_{\text{KP}} B_l,$$

where η is the learning rate and λ_{KP} a weight decay constant. The forward weight update is the standard pseudogradient update with weight decay, while the backward weight update is equivalent to gradient descent on

$$\text{tr}(x_l^\mathsf{T} B_l \widetilde{\nabla}_{l+1}) + \frac{\lambda_{\text{KP}}}{2\eta} ||B_l||^2.$$

Thus, if we define the *angle* primitive

$$\mathcal{P}_l^{\text{angle}} = \text{tr}(x_l^\mathsf{T} B_l \widetilde{\nabla}_{l+1}) = \text{tr}(x_l^\mathsf{T} \widetilde{\nabla}_l),$$

then the **Kolen-Pollack (KP)** update is given by gradient descent on the layer-wise regularization function

$$\mathcal{R}_{\text{KP}} = \sum_{l \in \text{layers}} \alpha \mathcal{P}_l^{\text{angle}} + \beta \mathcal{P}_l^{\text{decay}},$$

for $\alpha = 1$ and $\beta = \frac{\lambda_{\text{KP}}}{\eta}$. The angle primitive encourages alignment of the forward activations with the backward pseudogradients and is local according to the criterion for locality defined in §3.1. Thus, the Kolen-Pollack learning rule only involves the use of local primitives, but it does necessitate that the backward weight update given by the angle primitive is the exact transpose to the forward weight

update at each step of training. This constraint is essential to showing theoretically how Kolen-Pollack leads to alignment of the forward and backward weights (Kolen & Pollack, 1994), but it is clearly as biologically suspect as exact weight symmetry. To determine empirically how robust Kolen-Pollack is when loosening this hard constraint, we add random Gaussian noise to each update. As shown in Fig. S1, even with certain levels of noise, the Kolen-Pollack learning rule can still lead to well performing models. This suggests that a noisy implementation of Kolen-Pollack that removes the constraint of exactness might be biologically feasible.

While Kolen-Pollack uses significantly fewer metaparameters than weight mirror or information alignment, the correct choice of these metaparameters is highly dependent on the architecture. As shown in Fig. S5, the Kolen-Pollack learning rule, with metaparameters specified by Akrout et al. (2019), matched backpropagation performance for ResNet-18 and ResNet-50. However, a considerable performance gap with backpropagation as well as our proposed learning rules (information alignment, SA, and AA) emerged for different (ResNet-50v2) and deeper (ResNet-101v2, ResNet-152v2) architectures, providing additional evidence for the necessity of the circuits we propose in maintaining robustness across architecture.