One-Time Initial Alignment Enables Resilient Learning without Weight Transport

Jeonghwan Cheon Jaehyuk Bae Se-Bum Paik
Department of Brain and Cognitive Sciences
Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
{jeonghwan518, jaehyukbae, sbpaik}@kaist.ac.kr

Abstract

Backpropagation is the cornerstone of deep learning, but its reliance on repeated synchronization between forward and backward pathways makes it computationally expensive. Feedback alignment offers a promising alternative by approximating error gradients through fixed random feedback, thereby avoiding symmetric weight transport. However, this approach often struggles with poor learning performance and instability, especially in deep networks. Here, we investigate the practical effectiveness of aligning forward and feedback weights once at initialization while keeping feedback fixed throughout training. In large language model fine-tuning tasks, standard feedback alignment diverged, whereas feedback alignment with initial alignment enabled stable learning and performance comparable to backpropagation. Controlled network experiments further show that this simple initialization condition guides stable error minimization in the loss landscape, improving network trainability. Spectral analyses reveal that initial alignment promotes smoother gradient flow and convergence to flatter minima, resulting in better generalization and robustness. Together, our results suggest that one-time initial alignment provides a simple strategy for extending fixed-feedback learning to practical learning tasks without repeated forward-to-feedback weight transport.

1 Introduction

Recent breakthroughs in artificial intelligence have been driven by a powerful learning rule, backpropagation [1, 2]. Leveraging this method, modern large-scale deep networks have achieved superhuman performance in domains such as image recognition [3, 4] and language modeling [5, 6]. Despite its effectiveness, backpropagation faces significant challenges in terms of computational and resource efficiency [7–9]. For instance, training a single large language model can consume over a thousand megawatt-hours of electricity [10] and emit tens of tons of carbon dioxide [11], raising concerns about its sustainability. This heavy energy consumption largely stems from a core limitation of backpropagation: it requires dynamic access to separate memory units during learning [7, 12, 13].

These limitations motivate alternative learning rules that do not require repeated synchronization between separate forward and feedback pathways [14–22]. This requirement is particularly relevant to neuromorphic architectures that colocate memory and computation, where maintaining matched forward and feedback weights can impose additional communication and excessive programming costs [23–25]. Feedback alignment (FA) offers a simple approach by replacing the transposed forward weights with fixed random feedback matrices [26]. It demonstrates that neural networks can learn even without synchronized forward and backward weights during training. However, standard FA often exhibits a substantial performance gap relative to backpropagation in deeper networks and more complex tasks [27].

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

Previous work has shown that the learning dynamics of FA can be altered by controlling the correlation between the initial forward weights and the fixed feedback weights. Bordelon and Pehlevan [28] theoretically analyzed this relationship in infinite-width networks, characterizing how initial correlation affects the contribution of earlier layers to learning in the lazy regime and representation evolution in the feature-learning regime. Furthermore, Cheon et al. [29] showed that initial correlation between forward and feedback weights can emerge through random noise pretraining inspired by developmental neuroscience and demonstrated its benefits for subsequent learning.

Here, we revisit this notion of initial alignment to extend the applicability of feedback alignment to practical learning tasks. We show that one-time initial alignment can substantially improve fixed-feedback learning without requiring further forward-to-feedback weight transport. In the evaluated language-model fine-tuning tasks, initial feedback alignment achieves performance close to backpropagation while keeping the feedback weights fixed throughout training. Controlled network experiments further characterize how initial alignment shapes learning dynamics, including trainability, loss-landscape geometry, generalization, and robustness, and identify limitations associated with network depth and initial alignment. Overall, our results suggest that one-time initial alignment can extend the practical scope of feedback alignment while preserving fixed feedback pathways, making it a promising approach for neuromorphic systems where repeated weight synchronization is costly.

2 Preliminaries

2.1 Credit assignment and backpropagation

To optimize a network to perform a given task — to enable learning — each neuron must adjust its synaptic weights in a way that reduces the discrepancy between its actual and expected outputs. This, in turn, requires assigning an appropriate portion of the overall error to each weight, a challenge known as the credit assignment problem. In machine learning, backpropagation [1] offers an effective solution to this problem and has become the standard method for training artificial neural networks.

Concretely, consider a typical multi-layer feedforward network for pattern classification, denoted as $f_\theta : \mathbb{R}^m \to \mathbb{R}^d$, parameterized by $\theta = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=0}^{L-1}$. Given an input $\mathbf{x} \in \mathbb{R}^m$, the network computes its output through the forward pass:

$$\mathbf{o}_{l+1} = \mathbf{W}_l \mathbf{h}_l + \mathbf{b}_l, \quad \mathbf{h}_{l+1} = \phi(\mathbf{o}_{l+1}), \tag{1}$$

where $\mathbf{W}_l$ denotes the forward weight matrix at layer l, $\mathbf{b}_l$ is the corresponding bias vector. The signal is propagated through a nonlinear activation function ϕ. Once the forward pass is complete, the loss is computed as the discrepancy between the network's output and the ground-truth labels. Backpropagation then calculates the gradient of this loss with respect to each weight and bias, and these gradients are subsequently used to update the network parameters.

$$\delta_l = \frac{\partial \mathcal{L}}{\partial \mathbf{o}_l} = (\mathbf{W}_l^T \delta_{l+1}) \odot \phi'(\mathbf{o}_l), \tag{2}$$

where δ_l denotes the error signal at layer l, ϕ' represents the derivative of the activation function, and $\odot$ indicates the element-wise product. Based on this error signal, the weights are updated as

$$\Delta \mathbf{W}_l = -\eta \delta_{l+1} \mathbf{h}_l^T, \tag{3}$$

where η denotes the learning rate.

2.2 Weight transport problem and feedback alignment

Although backpropagation endows neural networks with powerful learning capabilities, it relies on exact knowledge of the forward weights in the next layer to compute the backward error signal — known as the weight transport problem [14, 15] — which underlies both its energy inefficiency and its lack of biological plausibility.

Feedback alignment [26] offers a more biologically plausible alternative by replacing the backward-pathway weights with fixed, random matrices $\mathbf{B}_l$, thereby eliminating the need for weight transport:

$$\delta_l = \frac{\partial \mathcal{L}}{\partial \mathbf{o}_l} = (\mathbf{B}_l \delta_{l+1}) \odot \phi'(\mathbf{o}_l). \tag{4}$$

Since feedback alignment substitutes the core backpropagation operation — the use of the transpose of the forward weights $\mathbf{W}_l$ — with a random, fixed matrix $\mathbf{B}_l$, the resulting error signal is initially uninformative. However, during training, the forward weights $\mathbf{W}_l$ gradually adapt to "align" with the fixed $\mathbf{B}_l^\top$. This emergent alignment causes the error signals (4) to approximate those computed by backpropagation (2).

3 One-time initial feedback alignment

Algorithm 1 Feedback alignment with initial weight alignment

procedure INITIALIZE FROM SCRATCH($f_\theta, \theta_{\text{init}}$)
  for $l = 0, \ldots, L-1$ **do**
    $\mathbf{W}_l \sim \mathcal{N}(0, a^2/fan_{\text{in}})$ ▷ initialize forward weights
    $\mathbf{R}_l \sim \mathcal{N}(0, a^2/fan_{\text{in}})$ ▷ independent feedback noise
    $\mathbf{B}_l \leftarrow \mathbf{W}_l^\top \cos(\theta_{\text{init}}) + \mathbf{R}_l \sin(\theta_{\text{init}})$
procedure INITIALIZE FOR FINE-TUNING($f_{\theta^{\text{pre}}}$)
  for $l = 0, \ldots, L-1$ **do**
    $\mathbf{W}_l \leftarrow \mathbf{W}_l^{\text{pre}}$ ▷ retain pretrained forward weights
    $\mathbf{B}_l \leftarrow (\mathbf{W}_l^{\text{pre}})^\top$ ▷ initialize fixed feedback weights
procedure TRAIN WITH FIXED FEEDBACK(f_θ, data)
  Keep all $\mathbf{B}_l$ fixed throughout training
  for each epoch **do**
    for each $(\mathbf{x}, \mathbf{y}) \in$ data **do**
      $\mathcal{L}(\theta) \leftarrow \text{Loss}(f_\theta(\mathbf{x}), \mathbf{y})$ ▷ forward pass, Eq. 1
      $\boldsymbol{\delta}_L \leftarrow \partial\mathcal{L}/\partial\mathbf{o}_L$
      for $l = L-1, \ldots, 1$ **do**
        $\boldsymbol{\delta}_l \leftarrow (\mathbf{B}_l\boldsymbol{\delta}_{l+1}) \odot \phi'(\mathbf{o}_l)$ ▷ Eq. 4
      for $l = 0, \ldots, L-1$ **do**
        $\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta\,\boldsymbol{\delta}_{l+1}\mathbf{h}_l^\top$ ▷ Eq. 3
        $\mathbf{b}_l \leftarrow \mathbf{b}_l - \eta\,\boldsymbol{\delta}_{l+1}$

Even though feedback alignment eliminates ongoing weight transport, its performance often remains below that of standard backpropagation. We therefore examined whether soft alignment at initialization improves learning with a fixed feedback pathway. Our method requires alignment only at initialization; thereafter, only the forward parameters evolve, while the feedback weights remain fixed. For scratch training, we first sample the forward weights and construct the feedback weights as a linear combination of their transpose and independent Gaussian noise. For pretrained fine-tuning, we retain the pretrained forward weights and copy their transpose once into the feedback pathway (Algorithm 1). Both procedures establish an initially aligned pair without subsequent synchronization.

4 Results

4.1 Initial alignment improves learning with fixed feedback

To compare the effect of initial alignment, we investigated neural network learning under three different feedback dynamics while keeping all other conditions constant. Backpropagation (BP) computes backward errors using the current forward weights and their transposes (Figure 1a). Feedback alignment (FA) instead uses a separate pathway with fixed, random feedback weights (Figure 1b). Initial feedback alignment (IFA) uses the same fixed-feedback rule but begins with aligned forward and feedback weights (Figure 1b,c). After initialization, the forward parameters change while the feedback weights remain fixed. We compared their computational flow assuming physically separate forward and feedback pathways (Figure 1c) and counted forward-to-feedback weight-transport operations (Figure 1d). A transport pass copies the complete set of corresponding forward weights into the feedback pathway. Over N training steps, BP requires N passes to provide the current forward transpose, whereas IFA requires only one initial pass. FA requires no forward-to-feedback transport when its random feedback weights are generated locally. These counts include initialization and describe weight transport between pathways.

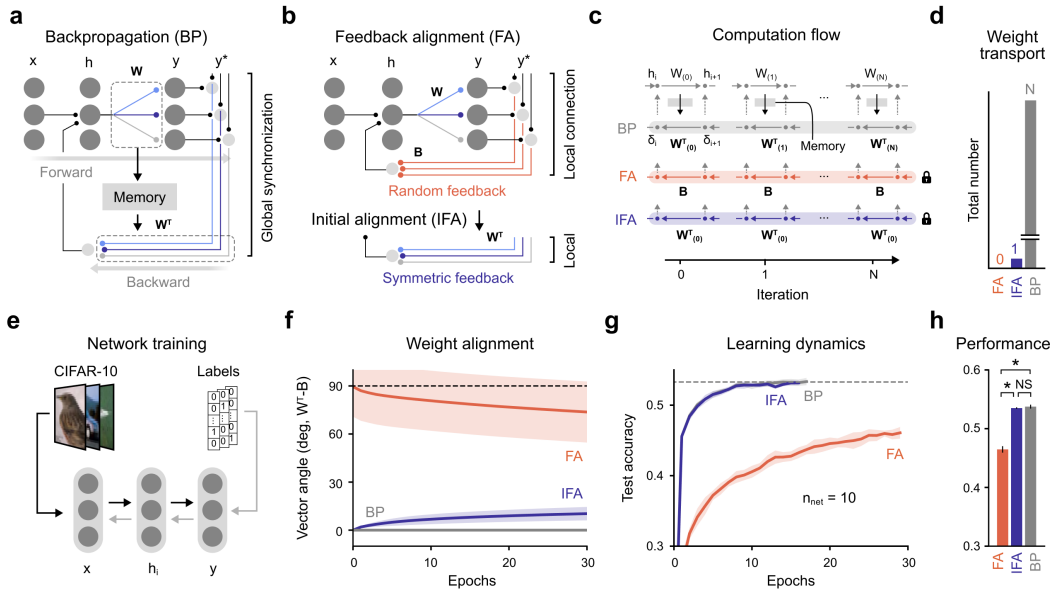


Figure 1: **Effect of initial weight alignment on learning without weight transport. a,** Schematic of the BP algorithm, where forward and backward weights are kept synchronized during training through external memory access. **b,** Schematic of fixed-feedback pathways. FA uses random feedback weights, whereas IFA initializes feedback in alignment with the forward weights and keeps it fixed during training. **c,** Computation flow for implementing BP, FA and IFA on-chip. **d,** Number of forward-to-feedback weight-transport operations over N training steps. BP requires N transport passes, FA requires none, and IFA requires one initial pass. **e,** Multi-layer feedforward networks trained from scratch on CIFAR-10 using different feedback rules. **f,** Alignment angle between forward and backward weights in the final layer of the network. **g,** Learning curves showing test accuracy over the course of training. **h,** Final test accuracy.

To compare these learning algorithms, we trained multi-layer feedforward networks on a natural image classification task using three distinct feedback schemes (Figure 1e). To characterize their learning dynamics, we first measured the alignment between forward and backward weights (Figure 1f) by computing the cosine similarity of corresponding weight vectors. As expected, BP maintains exact symmetry: forward and backward weights are identical because each forward weight is stored and retrieved during the backward pass. In FA, fixed random feedback weights gradually induce alignment in the forward weights during training. In contrast, IFA begins with $\mathbf{B}_l = \mathbf{W}_l(0)^\top$, and this alignment gradually relaxes as the forward weights change during learning. We next examined the learning curves for each feedback scheme (Figure 1g). Remarkably, IFA's convergence trajectory closely mirrors that of BP: it learns substantially faster than baseline FA and ultimately achieves performance comparable to BP. Indeed, final accuracy under IFA is significantly higher than under FA (Figure 1h; FA vs. IFA, $n_{\text{net}} = 10$, two-sided rank-sum test, $P < 10^{-3}$) and not significantly different from BP (BP vs. IFA, $n_{\text{net}} = 10$, two-sided rank-sum test, NS, $P = 0.082$). These findings suggest that a one-time alignment of forward and feedback weights at initialization is sufficient to enable learning performance comparable to BP, without requiring ongoing synchronization throughout training. The performance advantage of IFA over FA persisted over a longer training horizon, suggesting that initial alignment does more than accelerate early learning (Supplementary Results A.1.5).

One prominent application of IFA is downstream fine-tuning after deployment, where pretrained models must adapt to new tasks under the memory and energy constraints of edge devices. In this setting, a model can be pretrained with conventional BP and subsequently fine-tuned using a fixed feedback pathway to avoid repeated forward-to-feedback synchronization (Figure 2a). We tested this scenario by fine-tuning a pretrained Qwen3-1.7B model on the GSM8K mathematical-reasoning benchmark. Baseline FA completely failed in this large pretrained language model: the training loss rapidly diverged, showing that random feedback catastrophically destabilized fine-tuning rather than merely slowing convergence (Figure 2b). This collapse led to unreliable reasoning outputs, extremely high perplexity, and a near-zero correct-answer ratio (Figure 2c–f). By contrast, IFA remained stable,

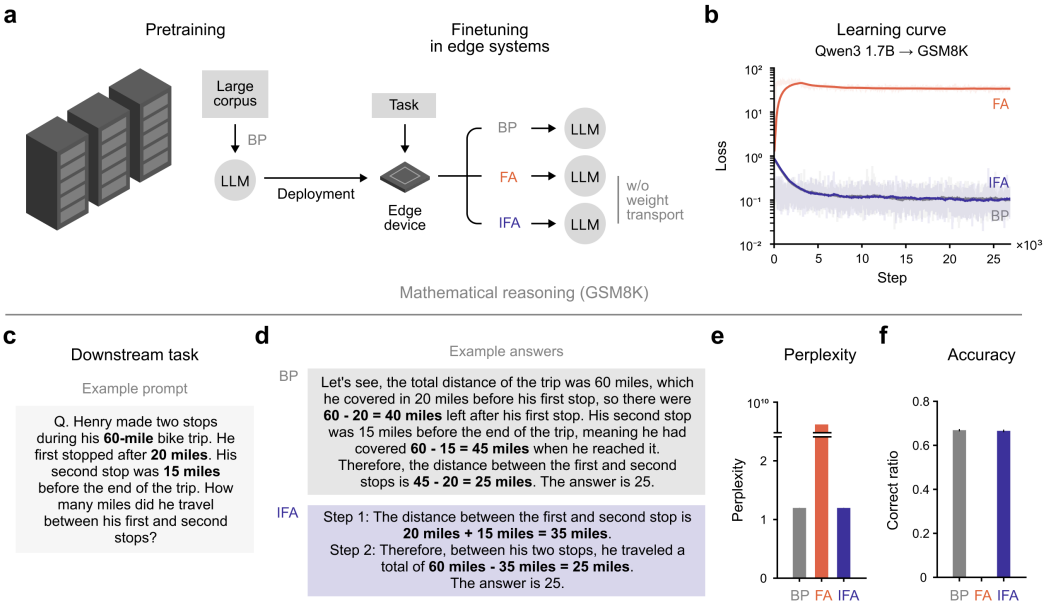


Figure 2: **Downstream fine-tuning of a pretrained language model without weight transport.** **a,** Schematic of the edge fine-tuning scenario. A large language model (LLM) is first pretrained with conventional BP on a large corpus and then deployed to an edge device, where task-specific fine-tuning is performed with BP, FA, or IFA. FA and IFA avoid dynamic weight transport during fine-tuning, while IFA initializes the feedback pathway from the pretrained forward weights only once. **b,** Learning curves for fine-tuning a Qwen3-1.7B model on the GSM8K mathematical-reasoning task. **c,** Example downstream prompt from GSM8K. **d,** Representative answers generated by BP- and IFA-fine-tuned models. **e,** Perplexity after fine-tuning. **f,** Correct-answer ratio on GSM8K.

closely followed the BP learning trajectory, and recovered BP-level perplexity and correct-answer ratio. Thus, one-time initialization of the feedback pathway from pretrained forward weights prevents catastrophic divergence and enables effective downstream adaptation without weight transport.

To assess whether this benefit generalizes across tasks and model sizes, we extended full fine-tuning to five benchmarks with Qwen3-1.7B and to ARC-Challenge and RTE with Qwen3-8B (Table 1). Across these seven settings, IFA achieves comparable accuracy to that of BP, whereas FA fails to converge. Thus, one-time alignment supported effective fine-tuning across the evaluated tasks and model sizes while retaining a fixed feedback pathway. Feedback transport decreased under the separate-pathway model, whereas storing fixed feedback increased GPU memory usage (Supplementary Results A.2.3–A.2.4).

Table 1: **Task accuracy after full fine-tuning of pretrained Qwen3 models.** BP, FA, and IFA values (%, higher is better) are reported as the mean ± standard deviation across three independent runs. Base denotes the pretrained model evaluated once before fine-tuning. All fine-tuning runs use three epochs; IFA initializes the feedback pathway from the pretrained forward weights once and then keeps it fixed.

			Fine-tuned		
Model	Task	Base	BP	FA	IFA
Qwen3-1.7B	StrategyQA	19.07	40.27 ± 4.62	0.00 ± 0.00	38.72 ± 1.39
Qwen3-1.7B	BoolQ	77.58	85.47 ± 0.03	0.00 ± 0.00	85.19 ± 0.18
Qwen3-1.7B	ARC-Challenge	71.76	76.93 ± 0.30	0.37 ± 0.36	77.19 ± 0.44
Qwen3-1.7B	RTE	39.35	86.64 ± 1.30	0.00 ± 0.00	86.16 ± 0.75
Qwen3-1.7B	CommonsenseQA	60.52	73.16 ± 0.71	0.00 ± 0.00	73.22 ± 0.43
Qwen3-8B	ARC-Challenge	90.87	92.24 ± 0.31	0.03 ± 0.05	92.18 ± 0.18
Qwen3-8B	RTE	86.28	90.61 ± 0.00	0.00 ± 0.00	90.85 ± 0.21

4.2 Stabilizing learning through initial weight alignment

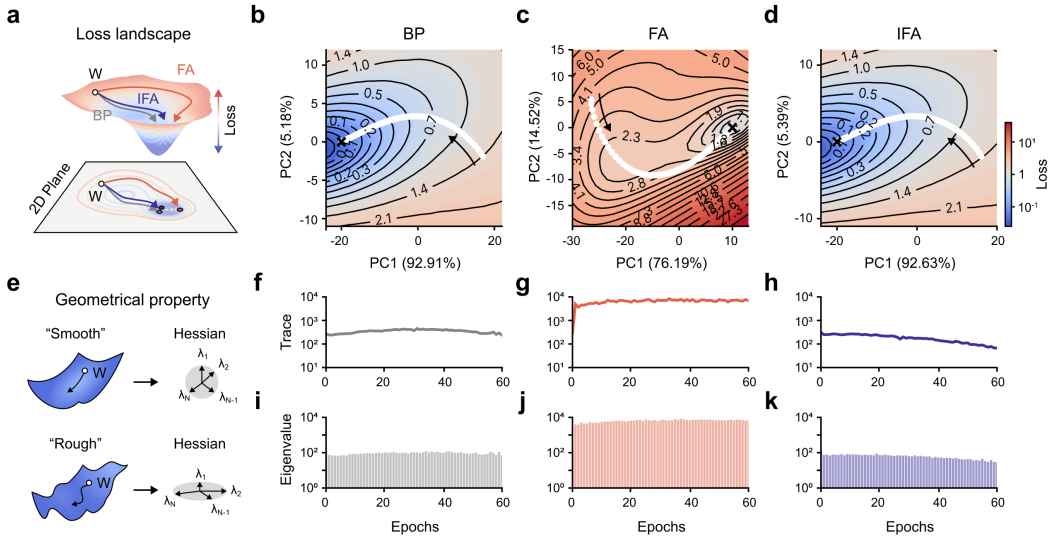


Figure 3: **Loss landscape analysis of learning trajectories in MLPs trained from scratch on CIFAR-10 under different feedback rules. a,** Illustration of neural networks traversing the loss landscape during training, with each trajectory corresponding to a different learning algorithm. **b–d,** Training trajectories projected onto a two-dimensional subspace defined by the first two principal components (PCA): **b,** BP, **c,** FA, **d,** IFA. **e,** Geometric properties of the loss landscape, quantified using the Hessian's spectrum; smoother landscapes exhibit narrower spectra (i.e., smaller spectral radius). **f–h,** Trace of the Hessian over the course of training. **i–k,** Maximum eigenvalue of the Hessian during training for BP, FA, and IFA, respectively.

Next, we examined how initial alignment influences learning dynamics within the loss landscape, using principal component analysis [30] (Figure 3a) to visualize training trajectories. Specifically, we compared the trajectories produced under three feedback dynamics: BP (Figure 3b), FA (Figure 3c), and IFA (Figure 3d). BP exhibits stable and directed trajectories toward low-loss solutions. In contrast, FA results in more erratic trajectories — longer and less direct — indicating inefficient navigation of the loss landscape. Notably, IFA produces stable trajectories and a loss landscape structure that closely resemble those of BP, suggesting that initial alignment facilitates more efficient and stable learning. IFA also remained aligned with the current BP gradient despite parameter and representation changes (Supplementary Results A.3.5–A.3.6).

To quantitatively compare the learning trajectories under different feedback dynamics, we performed a spectral analysis of the loss Hessian [31]. Specifically, we assessed the smoothness of the loss landscape (Figure 3e) by measuring the trace of the Hessian (Figure 3f–h) and its maximum eigenvalue (Figure 3i–k). Consistent with our trajectory-based observations, BP maintains low Hessian trace and top eigenvalue throughout training, indicating a smooth loss surface. In contrast, FA exhibits substantially higher trace and top eigenvalue, suggesting a much rougher landscape. Notably, IFA preserves low Hessian trace and top eigenvalues across the entire learning process. These results suggest that providing initial guidance for error minimization is sufficient to promote stable learning dynamics and a smooth optimization trajectory, even in the absence of ongoing weight transport.

Building on the observation that stable learning induced by initial alignment between forward and backward weights governs the entire training process, we hypothesized that this alignment enhances neural network trainability under various conditions. First, we investigated networks initialized with different combinations of forward and backward weight variances (Figure 4a). Next, we examined how initial alignment influences the maximum trainable depth (Figure 4b). In baseline FA, trainability rapidly degrades as depth increases, whereas IFA enables training at greater depths, comparable to those achievable with BP. Finally, we considered a limited-data regime (Figure 4c). Because FA must learn to approximate BP from data, it fails when the dataset is too small. In contrast, IFA enables successful training even under severe data constraints, achieving performance comparable to BP.

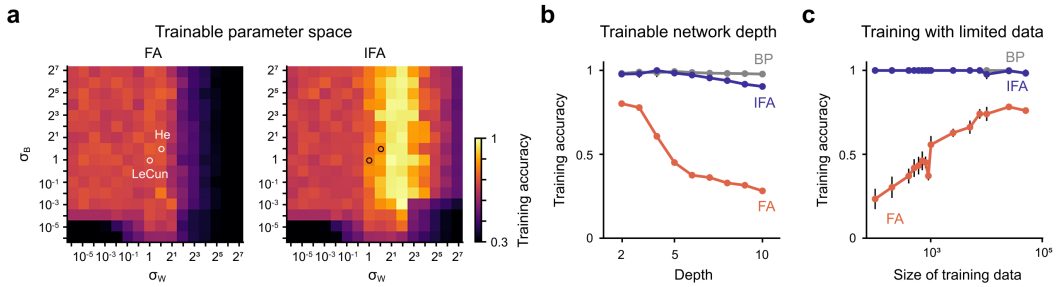


Figure 4: **Trainability of MLPs trained from scratch on CIFAR-10 under various conditions. a,** Trainable parameter space: the x- and y-axes represent the variances of the forward and backward weights, respectively. Color indicates the final training accuracy. The point $(1, 1)$ corresponds to LeCun initialization; $(\sqrt{2}, \sqrt{2})$ corresponds to He initialization. **b,** Trainability versus network depth: final training accuracy for feedforward networks with depths ranging from 2 to 10 layers. **c,** Trainability under limited data: final training accuracy as a function of training set sizes, ranging from 100 to 50,000 samples.

4.3 Initial alignment enhances robust generalization

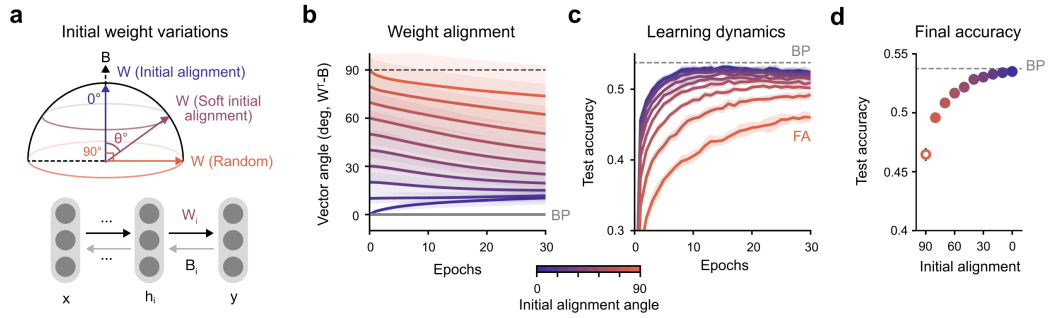


Figure 5: **Effect of the degree of initial alignment on network performance. a,** Schematic of initial alignment between forward and feedback weights. Exact alignment gives $\mathbf{B}_l = \mathbf{W}_l(0)^\top$, whereas FA uses independent forward and feedback weights. Soft alignment combines aligned and independent Gaussian components. **b,** Angle between forward and backward weights over the course of training. **c,** Learning curves showing test accuracy during training. **d,** Relationship between the initial alignment angle and final accuracy.

To evaluate the impact of initial alignment, we systematically varied the alignment angle between forward and backward weights and analyzed the resulting learning dynamics (Figure 5a). An initialization parameter of $0°$ gives exact alignment between $\mathbf{W}_l(0)$ and $\mathbf{B}_l^\top$, while $90°$ gives independent random weights, as in baseline FA. Intermediate settings were generated by mixing aligned and independent Gaussian components, with θ_{init} controlling their relative contributions. We found that the learning dynamics depend strongly on the initial alignment (Figure 5b). When the initial alignment is strong, learning proceeds by gradually relaxing this alignment; when the initial alignment is weak, the forward weights progressively align with the fixed feedback weights. The learning curves reflect this behavior (Figure 5c): stronger initial alignment leads to faster convergence and higher final test accuracy. Plotting the final test accuracy as a function of the initial alignment angle (Figure 5d) reveals a trend — performance improves with stronger alignment but eventually saturates. These results suggest that while a good initial alignment is critical for effective learning, perfect symmetry between forward and backward weights is not necessary.

Next, we performed a spectral analysis of the converged optima obtained under varying initial alignment angles. Specifically, we visualized the loss landscape by perturbing the model parameters along two principal directions (Figure 6a), defined by the eigenvectors corresponding to the largest and

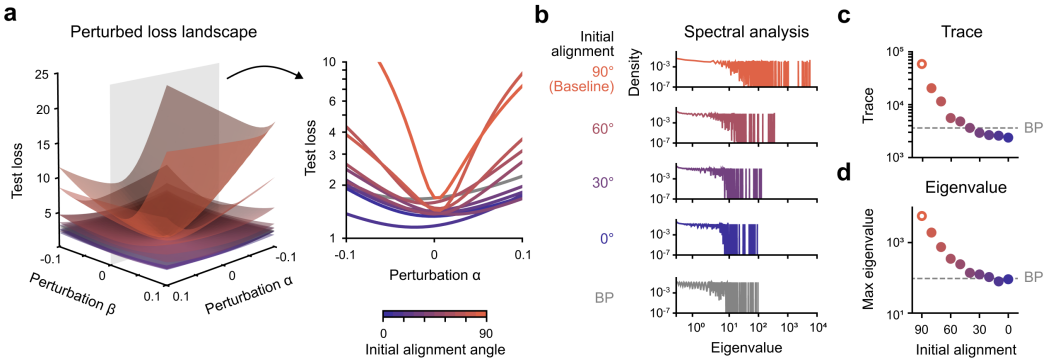


Figure 6: **Spectral analysis across initial alignment angles in MLPs trained from scratch on CIFAR-10. a,** Loss landscape visualized by perturbing the trained model parameters within a two-dimensional subspace defined by the top two eigenvectors of the Hessian. The axes, denoted by α and β, represent scaling factors for perturbations along the first and second eigenvector directions, respectively. The landscapes are color-coded according to each network's initial alignment angle. **b,** Spectral density of the Hessian eigenvalues, illustrating the distribution of curvature across the parameter space. **c,** Relationship between the initial alignment angle and the trace of the Hessian. **d,** Relationship between the initial alignment angle and the largest Hessian eigenvalue.

second-largest eigenvalues of the Hessian. The baseline FA method exhibits a rough loss landscape, where even small perturbations in parameter space lead to significant increases in loss. In contrast, applying a soft initial alignment dramatically smooths the loss landscape of the converged optima, making it comparable to that observed with BP. Furthermore, increasing the degree of initial weight alignment leads to progressively smoother loss landscapes.

Additionally, we validated this trend by analyzing the spectral density of Hessian eigenvalues (Figure 6b). Under baseline FA, the spectrum is shifted toward higher eigenvalues and exhibits greater dispersion compared to BP. Introducing initial alignment compresses the spectrum, resulting in a distribution that closely matches that of BP. To quantify the smoothness of the loss landscape, we computed both the Hessian trace (Figure 6c) and its largest eigenvalue (Figure 6d). Both metrics decrease as the initial alignment angle decreases, and then plateau as alignment becomes stronger. These geometric insights align with our earlier observation that test accuracy improves and then saturates with stronger initial alignment (Figure 5d). Collectively, these findings indicate that initial weight alignment plays a critical role in shaping a smooth loss landscape — a known factor in promoting robust generalization [32–36]. Notably, only a moderate degree of alignment is necessary to obtain these benefits.

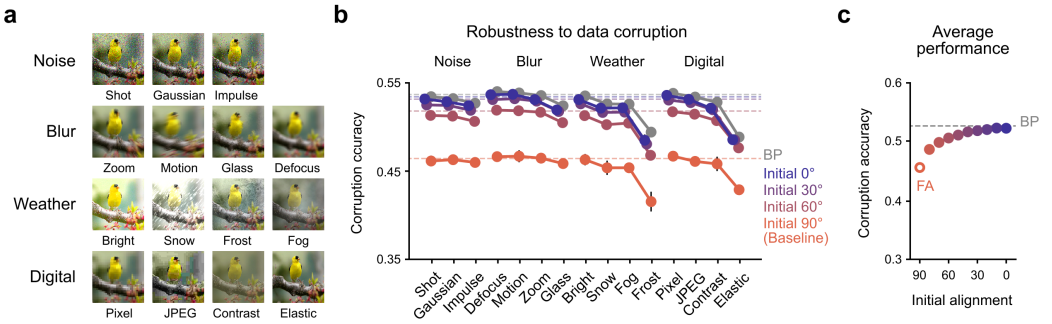


Figure 7: **Effect of initial weight alignment on corruption robustness. a,** Fifteen image corruption types, grouped into four categories — noise, blur, weather, and digital — were applied to images to evaluate robustness. **b,** Classification accuracy on corrupted inputs (solid lines with markers) compared to accuracy on clean, uncorrupted images (dotted lines). **c,** Relationship between the initial alignment angle and robustness, measured as the average accuracy across all corruption types.

The spectral analysis revealed that initial weight alignment promotes convergence to a smoother loss landscape. Prior work has suggested that smoother loss landscape reduce a network's sensitivity to input distortions or perturbations [36], implying that initial weight alignment may enhance robustness. To test this hypothesis, we employed a benchmark image dataset featuring fifteen corruption types, categorized into noise, blur, weather, and digital distortions [37] (Figure 7a). We trained networks using feedback dynamics across varying initial alignment angles and evaluated their classification accuracy on the corrupted inputs (Figure 7b). Notably, increasing the degree of initial alignment consistently improved test accuracy across all corruption types (Figure 7c). Furthermore, when the severity of each corruption was gradually increased, networks with stronger initial alignment exhibited greater resistance to these distortions. These results suggest that initial weight alignment, by promoting a smoother loss landscape, ultimately enhances the robustness of the network.

4.4 The role of weight misalignment in promoting adversarial robustness

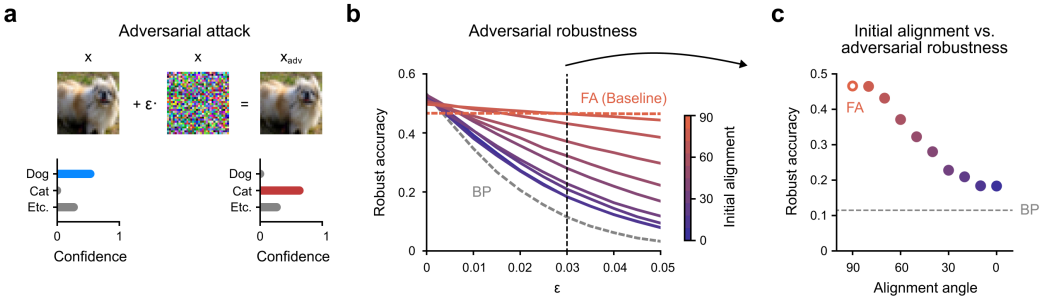


Figure 8: **Effect of weight misalignment on adversarial robustness. a,** Illustration of an adversarial attack: small, targeted perturbations are added to input images to induce misclassification. **b,** Classification accuracy as a function of increasing adversarial perturbation strength. **c,** Relationship between the initial alignment angle and adversarial robustness, measured as classification accuracy under attack with perturbation magnitude $\epsilon = 0.03$.

Thus far, we have emphasized that stronger initial weight alignment enhances standard training performance. Conversely, we also demonstrate that weight misalignment can confer benefits in the context of adversarial robustness. To evaluate this, we conducted adversarial attacks using the Fast Gradient Sign Method (FGSM) (Figure 8a), which applies gradient-based perturbations to input images to induce misclassification [38]. Under standard BP, model accuracy rapidly deteriorates as the perturbation strength increases (Figure 8b, BP). In contrast, models trained with IFA maintain significantly higher accuracy, even under stronger attacks (Figure 8b, IFA). This robustness appears to emerge from the gradual "loosening" of weight alignment that naturally occurs during fixed-feedback learning, which reduces the impact of adversarial perturbations. Furthermore, we observed that larger initial misalignment further improves adversarial robustness (Figure 8c). Taken together, these findings highlight a functional advantage of weight misalignment and underscore the practical benefits of employing "soft" alignment at initialization.

5 Discussion

We demonstrated that one-time initial alignment substantially improves learning with fixed feedback pathways, particularly in pretrained-model fine-tuning. Specifically, standard FA diverged in language-model fine-tuning, whereas simply initializing the feedback weights to match the pretrained forward weights enabled stable learning and performance comparable to backpropagation. After this single initialization step, the feedback weights remained fixed throughout training. These results show that one-time initial alignment can have lasting effects on subsequent learning and support effective fine-tuning in large pretrained networks. Controlled network experiments further examined trainability, loss-landscape geometry, generalization, and robustness, characterizing the benefits and limitations of this initialization.

Previous work has established initial forward–feedback correlation as an important factor shaping feedback alignment dynamics. Bordelon and Pehlevan [28] theoretically examined how this corre-

lation influences learning and representation evolution by studying fixed feedback correlated with the initial forward weights. Our work builds on this perspective by evaluating initially aligned fixed feedback across practical learning tasks. Rather than introducing initial correlation as a new principle, we demonstrate its empirical effectiveness and investigate the conditions under which its benefits are maintained, particularly when fine-tuning pretrained models.

Our analyses further clarify how the benefits of initial alignment can persist during learning. In the controlled networks, forward weights and representations changed substantially while the update direction remained aligned with the current backpropagation gradient. These observations show that effective fixed-feedback learning need not be restricted to networks whose parameters and representations remain close to initialization. However, they do not establish the same learning regime in language-model fine-tuning, where proximity to pretrained parameters may contribute to the effectiveness of fixed feedback. Performance gaps also persisted in deeper networks, indicating that initial alignment does not universally recover backpropagation performance.

Reducing memory requirements and data movement is an important direction for efficient neural-network training. Existing approaches reduce activation storage through recomputation [39] or avoid storing activations for backpropagation through forward-only optimization [40]. Our approach targets repeated weight transport between physically separate forward and feedback pathways. One-time initial alignment removes subsequent synchronization between these pathways while retaining backward error propagation. This reduction does not itself imply lower GPU memory usage or faster computation, since fixed feedback must still be stored and used during the backward pass. Its potential hardware benefits depend on architectures where feedback synchronization incurs substantial communication and programming costs. Our results therefore motivate further evaluation of initially aligned fixed feedback in neuromorphic systems where these costs are a bottleneck [23, 41].

6 Broader impacts

Reducing the resource demands of neural-network training is important for the sustainability of artificial intelligence. By showing that pretrained models can be adapted with fixed feedback, this work may inform the design of learning systems that avoid repeated synchronization between separate forward and feedback pathways. In neuromorphic architectures where such synchronization is costly, this approach could contribute to lower training energy requirements. However, the present experiments do not establish reductions in energy consumption or carbon emissions.

7 Limitations

Our method targets architectures with physically separate forward and feedback pathways, where repeatedly synchronizing the feedback weights with the current forward weights can impose transport and programming costs. One-time alignment preserves learning performance close to BP in the evaluated fine-tuning tasks while eliminating further forward-to-feedback weight transport. However, this reduction in transport does not by itself imply computational or memory-efficiency gains on conventional GPUs. IFA retains backward error propagation and, in our dense GPU implementation, stores additional fixed feedback matrices, resulting in higher peak GPU memory usage than BP. Potential hardware benefits require validation on systems where feedback synchronization is a bottleneck.

8 Code availability

Python 3.12 (Python software foundation) with PyTorch 2.12 is used to perform the simulation. The code used in this work is available at `https://github.com/cogilab/IFA`.

Acknowledgments and Disclosure of Funding

This work was supported by grants from the National Research Foundation of Korea (NRF) (RS-2022-NR070602 and RS-2025-02304581 to S.-B.P.) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) (RS-2026-25518317 and RS-2026-25522672 to S.-B.P.), funded by the Korean government (MSIT).

References

[1] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).

[2] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

[3] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* **25** (2012).

[4] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

[5] Achiam, J. *et al.* Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[6] Team, G. *et al.* Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[7] Strubell, E., Ganesh, A. & McCallum, A. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 13693–13696 (2020).

[8] Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE* **105**, 2295–2329 (2017).

[9] Schwartz, R., Dodge, J., Smith, N. A. & Etzioni, O. Green ai. *Communications of the ACM* **63**, 54–63 (2020).

[10] Patterson, D. *et al.* Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350* (2021).

[11] Luccioni, A. S., Viguier, S. & Ligozat, A.-L. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research* **24**, 1–15 (2023).

[12] Horowitz, M. Computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, 10–14 (IEEE, 2014).

[13] Chen, Y.-H., Emer, J. & Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH computer architecture news* **44**, 367–379 (2016).

[14] Grossberg, S. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science* **11**, 23–63 (1987).

[15] Crick, F. The recent excitement about neural networks. *Nature* **337**, 129–132 (1989).

[16] Liao, Q., Leibo, J. & Poggio, T. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30 (2016).

[17] Xiao, W., Chen, H., Liao, Q. & Poggio, T. Biologically-plausible learning algorithms can scale to large datasets. In *International Conference on Learning Representations* (2019).

[18] Kolen, J. F. & Pollack, J. B. Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 3, 1375–1380 (IEEE, 1994).

[19] Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T. & Tweed, D. B. Deep learning without weight transport. *Advances in Neural Information Processing Systems* **32** (2019).

[20] Whittington, J. C. & Bogacz, R. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation* **29**, 1229–1262 (2017).

[21] Song, Y. *et al.* Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature Neuroscience* **27**, 348–358 (2024).

[22] Scellier, B. & Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience* **11**, 24 (2017).

[23] Indiveri, G. & Liu, S.-C. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE* **103**, 1379–1397 (2015).

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

[25] Richards, B. A. & Lillicrap, T. P. Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology* **54**, 28–36 (2019).

[26] Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications* **7**, 13276 (2016).

[27] Bartunov, S. *et al.* Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in Neural Information Processing Systems* **31** (2018).

[28] Bordelon, B. & Pehlevan, C. The influence of learning rule on representation dynamics in wide neural networks. In *International Conference on Learning Representations* (2023).

[29] Cheon, J., Lee, S. W. & Paik, S.-B. Pretraining with random noise for fast and robust learning without weight transport. In Globerson, A. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 37, 13748–13768 (Curran Associates, Inc., 2024).

[30] Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., 2018).

[31] Yao, Z., Gholami, A., Keutzer, K. & Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, 581–590 (IEEE, 2020).

[32] Hochreiter, S. & Schmidhuber, J. Flat minima. *Neural computation* **9**, 1–42 (1997).

[33] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (2016).

[34] Chaudhari, P. *et al.* Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment* **2019**, 124018 (2019).

[35] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. & Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).

[36] Foret, P., Kleiner, A., Mobahi, H. & Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations* (2021).

[37] Hendrycks, D. & Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations* (2019).

[38] Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[39] Chen, T., Xu, B., Zhang, C. & Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174* (2016).

[40] Malladi, S. *et al.* Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems*, vol. 36, 53038–53075 (2023).

[41] Merolla, P. A. *et al.* A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).

[42] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278–2324 (1998).

[43] Netzer, Y. *et al.* Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2 (Granada, 2011).

[44] Krizhevsky, A. Learning multiple layers of features from tiny images. Tech. Rep., University of Toronto (2009).

[45] Coates, A., Ng, A. & Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223 (JMLR Workshop and Conference Proceedings, 2011).

[46] Kurakin, A., Goodfellow, I. J. & Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, 99–112 (Chapman and Hall/CRC, 2018).

[47] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[48] Yao, Z., Gholami, A., Lei, Q., Keutzer, K. & Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., 2018).

[49] Avron, H. & Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)* **58**, 1–34 (2011).

[50] Golub, G. H. & Welsch, J. H. Calculation of gauss quadrature rules. *Mathematics of computation* **23**, 221–230 (1969).

[51] Ghorbani, B., Krishnan, S. & Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 2232–2241 (PMLR, 2019).

A Supplementary results

A.1 One-time initial alignment enables learning without weight transport

A.1.1 Layer-wise weight alignment dynamics

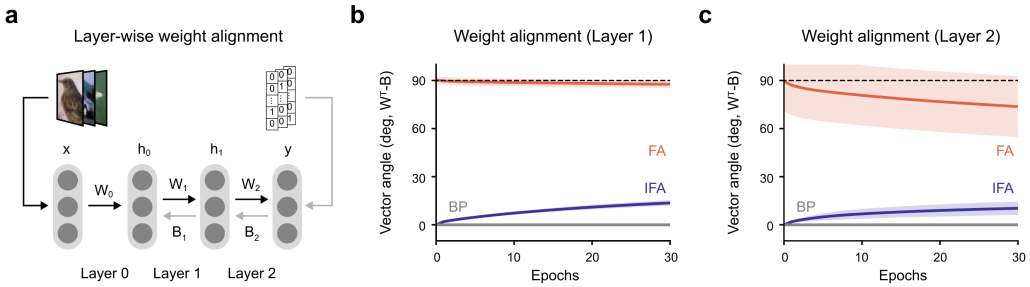


Figure S1: **Weight alignment between forward and backward weights during learning. a,** Schematic of the three-layer feedforward neural network and its learning process. Forward and backward weights are analyzed in all layers except the first (Layer 0). **b–c,** Alignment angle between forward and backward weights in each layer of the network. **b,** Weight alignment in Layer 1. The shaded area represents the standard deviation of alignment angles across 512 neurons. **c,** Weight alignment in Layer 2, the final layer of the network. The shaded area represents the standard deviation of alignment angles across 512 neurons.

To generalize the trend of alignment dynamics observed in the penultimate layer (Figure 1f), we extended the analysis to all layers (Figure S1a). With the exception of the first layer, which lacks corresponding backward weights, we examined the vector angles between forward and backward weights in the second (Figure S1b) and final layers (Figure S1c). Consistent with the main results, feedback alignment (FA) [26] learns by aligning forward weights with backward weights over the course of training, whereas standard backpropagation (BP) [1] enforces identical forward and backward weights. Our model, initial feedback alignment (IFA), learns by gradually relaxing this alignment during training.

A.1.2 Training details for different learning models

To investigate the impact of initial alignment on neural network learning, we visualized learning curves under three different learning rules: (1) BP, (2) FA, and (3) IFA, while keeping all other experimental conditions constant. We plotted training accuracy (Figure S2a), training loss (Figure S2b), test accuracy (Figure S2c), and test loss (Figure S2d) over the course of training. Baseline FA exhibited slower convergence and lower performance compared to standard BP. In contrast, IFA achieved comparable learning speed and accuracy to BP, with nearly overlapping learning curves for both the training and test sets.

A.1.3 Validation across neural network architectures

Table S1: **Benchmarked performance of the convolutional neural networks.** Each performance value (%) is reported as the mean ± standard deviation across ten independent trials.

Method	Initial angle	2 Conv + 3 FC	5 Conv + 3 FC
BP	—	62.13 ± 0.90	73.10 ± 0.29
FA	90°	45.67 ± 1.71	50.41 ± 1.11
IFA	0°	56.37 ± 1.24	65.55 ± 0.62

Our main results, obtained using multi-layer feedforward neural networks, demonstrated that initial alignment enhances the scalability of feedback alignment algorithms. To assess whether this effect generalizes to other architectures and deeper models, we further evaluated convolutional neural networks (CNNs) (Figure S3). We first tested a LeNet-variant CNN [42] with two convolutional layers and three fully connected layers (Figure S3a). We found that both the training and test accuracy

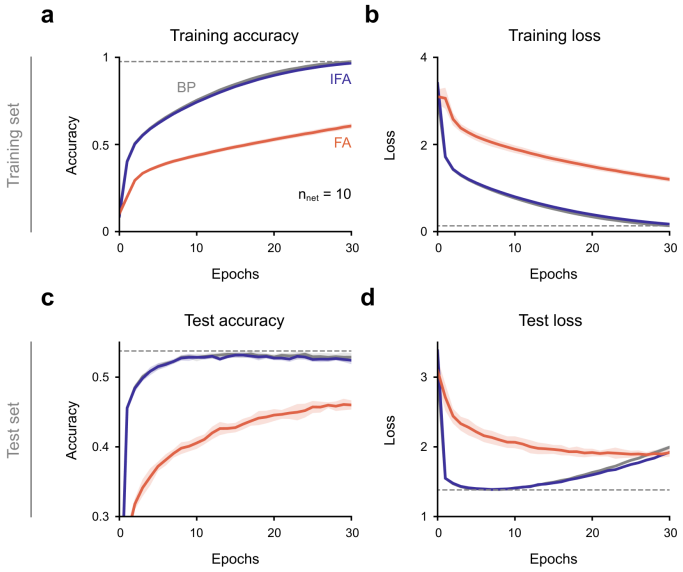


Figure S2: **Learning curves across different feedback dynamics. a–d,** Learning curves over the course of training. **a,** Training accuracy. **b,** Training loss. **c,** Test accuracy. **d,** Test loss. The dotted gray horizontal line represents the final performance achieved by BP. The shaded area denotes the standard deviation across ten independent trials.

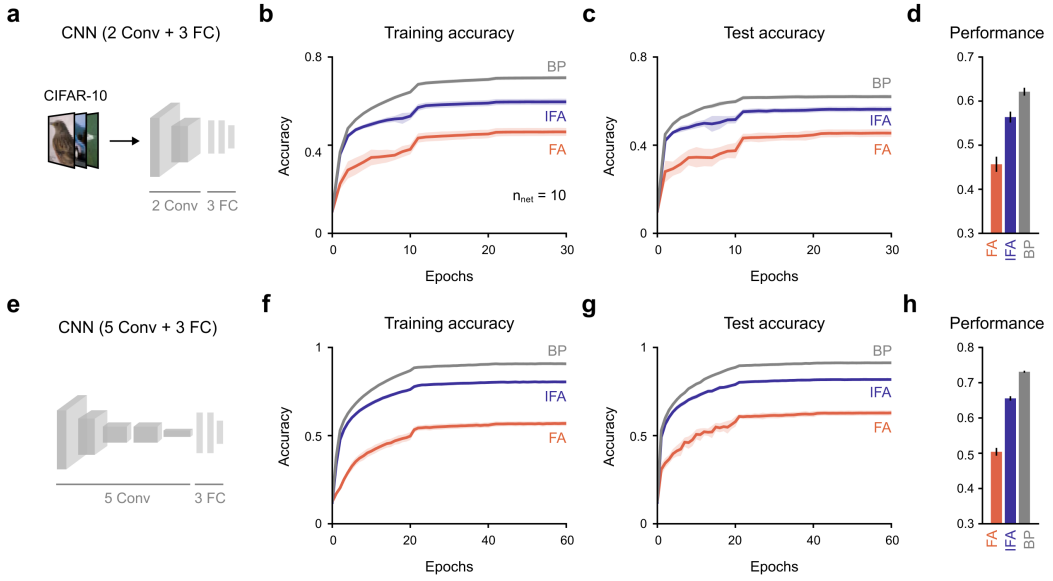


Figure S3: **Effect of initial weight alignment on convolutional neural networks. a,** A convolutional neural network variant of LeNet, consisting of two convolutional layers and three fully connected layers, is used. The model is trained on CIFAR-10. **b,** Training accuracy. **c,** Test accuracy. **d,** Final test accuracy. **e,** A convolutional neural network variant of AlexNet, consisting of five convolutional layers and three fully connected layers, is used. **f,** Training accuracy. **g,** Test accuracy. **h,** Final test accuracy. The shaded areas in the line plots and the error bars in the bar plots represent the standard deviation across ten independent trials.

of IFA were significantly higher than those of baseline FA, and the performance gap between IFA and BP was reduced (Figure S3b, c). The final test performance of IFA substantially outperformed that of baseline FA (Figure S3d; Table S1). We also evaluated an AlexNet-variant CNN [3] consisting of

five convolutional layers and three fully connected layers (Figure S3e), and observed a similar trend (Figure S3f–h; Table S1). For training, we used the Adam optimizer with learning rates of 0.001 and 0.0001, and applied weight decay with a decay coefficient of 0.0001 to prevent overfitting. These findings suggest that IFA is broadly applicable to neural architectures beyond multi-layer perceptrons and can scale effectively to modern deep learning models.

A.1.4 Benchmarking model performance on standard datasets

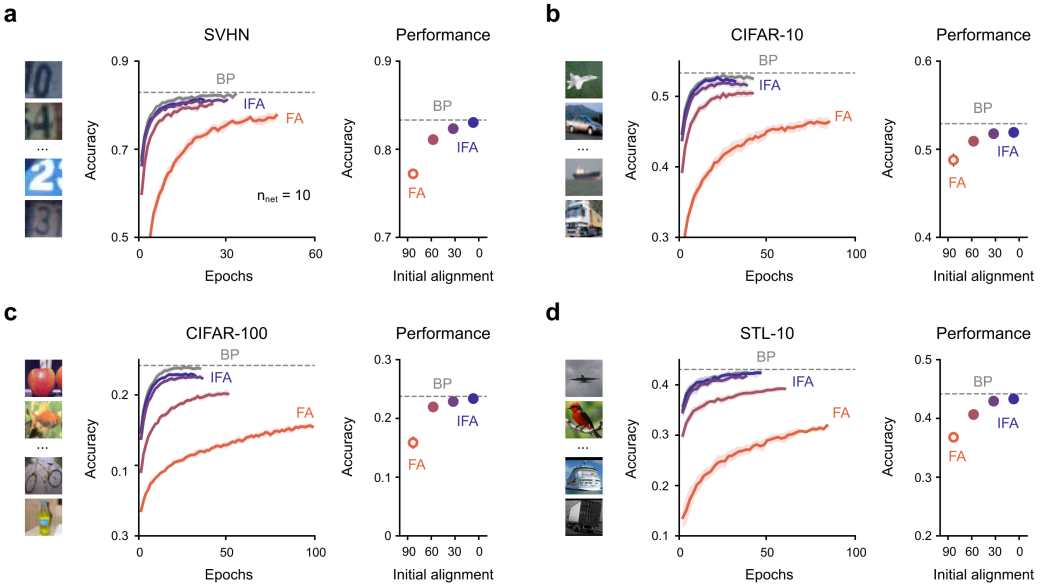


Figure S4: **Benchmarked network performance across various image datasets. a–d,** Learning curves showing test accuracy over the course of training (left) and the benchmarked final test accuracy (right). **a,** SVHN. **b,** CIFAR-10. **c,** CIFAR-100. **d,** STL-10.

Table S2: **Performance of the model on each dataset (SVHN, CIFAR-10, CIFAR-100, STL-10).** Each performance value (%) is reported as the mean ± standard deviation across ten independent trials.

Method	Initial angle	SVHN	CIFAR-10	CIFAR-100	STL-10
BP	—	82.90 ± 0.11	53.32 ± 0.27	24.18 ± 0.28	43.75 ± 0.22
FA	90°	78.78 ± 0.72	47.21 ± 0.35	16.79 ± 0.57	35.87 ± 1.01
IFA	60°	80.91 ± 0.16	51.09 ± 0.43	20.66 ± 0.36	41.95 ± 0.24
IFA	30°	81.76 ± 0.16	52.33 ± 0.30	22.93 ± 0.12	42.87 ± 0.30
IFA	0°	81.91 ± 0.20	53.04 ± 0.33	23.31 ± 0.20	43.37 ± 0.31

Our main results were obtained using the CIFAR-10 dataset. To demonstrate that IFA generalizes across various datasets, we benchmarked the performance of BP, FA, and IFA on SVHN [43] (Figure S4a), CIFAR-10 [44] (Figure S4b), CIFAR-100 [44] (Figure S4c), and STL-10 [45] (Figure S4d). Final accuracy was evaluated by extending training until test accuracy stopped improving for ten consecutive epochs (Table S2). Consistent with our main findings, the performance of baseline FA was significantly lower than that of BP. However, incorporating initial alignment substantially improved performance, in some cases reaching levels comparable to backpropagation. These results suggest that IFA is generalizable across different datasets and can effectively enhance learning performance.

A.1.5 Long-horizon training and persistence of the performance gap

To examine whether the performance advantage of IFA persists with longer training, we extended training of the CIFAR-10 multi-layer feedforward network to 150 epochs, five times the main training

horizon. The network contained two hidden layers with 512 units each, and we compared BP, FA, and IFA under the same training protocol. We evaluated test accuracy at epochs 30 and 150, the change in test accuracy between epochs 100 and 150, and the slope of training loss over the final 20 epochs (Table S3).

Table S3: **Long-horizon learning on CIFAR-10.** Values are reported as the mean ± standard deviation across 10 independent runs. Accuracy is expressed in %, and the change in accuracy is expressed in percentage points (pp). The training-loss slope is the slope of an ordinary least-squares fit over the final 20 epochs, in loss units per epoch.

Method	Test accuracy at epoch 30	Test accuracy at epoch 150	Accuracy change epochs 100–150 (pp)	Final-20-epoch training-loss slope
BP	53.12 ± 0.48	53.81 ± 0.41	-0.12 ± 0.69	$-1.59 \times 10^{-4} \pm 1.61 \times 10^{-3}$
FA	45.91 ± 1.12	47.54 ± 0.64	$+0.19 \pm 0.69$	$-9.61 \times 10^{-4} \pm 7.91 \times 10^{-4}$
IFA	52.34 ± 0.41	53.12 ± 0.53	$+0.73 \pm 1.96$	$-1.45 \times 10^{-3} \pm 2.70 \times 10^{-3}$

IFA maintained test accuracy close to BP throughout the extended training period. At epoch 150, IFA achieved $53.12 \pm 0.53\%$, compared with $53.81 \pm 0.41\%$ for BP and $47.54 \pm 0.64\%$ for FA. Thus, IFA remained 5.58 pp above FA and 0.69 pp below BP. The modest changes in late-stage accuracy and small mean training-loss slopes indicate that extending training did not eliminate the performance gap within the evaluated horizon. These results demonstrate persistent near-BP performance over 150 epochs, without establishing convergence to an identical asymptotic solution.

A.1.6 Training time to target accuracy

To compare learning speed, we measured the first epoch at which each method reached a target test accuracy using the same 150-epoch trajectories (Table S4). We first used a common target of 95% of the final accuracy of the corresponding BP run. We also measured the epochs required to reach 80%, 90%, and 99% of each method's own final accuracy, defined as its test accuracy at epoch 150. The first comparison evaluates progress toward a shared performance level, whereas the second characterizes the rate at which each method approaches its own final performance.

Table S4: **Epochs required to reach target test accuracy.** Values are reported as the mean ± standard deviation across 10 independent runs. The common target is 95% of the corresponding BP run's epoch-150 accuracy. Method-specific targets are fractions of each run's own epoch-150 accuracy. "Not reached" indicates that the common target was not reached within 150 epochs in any FA run.

	Common target	Method-specific target		
Method	95% of BP final	80% of final	90% of final	99% of final
BP	4.4 ± 0.5	1.00 ± 0.00	2.40 ± 0.52	11.30 ± 3.77
FA	Not reached	7.10 ± 1.20	16.10 ± 2.92	42.60 ± 9.97
IFA	5.4 ± 1.2	1.00 ± 0.00	2.30 ± 0.48	10.40 ± 4.60

IFA reached the common BP-based target at 5.4 ± 1.2 epochs, approximately one epoch after BP, whereas FA did not reach this target within 150 epochs. IFA also approached its own final accuracy at a rate similar to BP: reaching 99% of final accuracy required 10.40 ± 4.60 epochs for IFA and 11.30 ± 3.77 epochs for BP, compared with 42.60 ± 9.97 epochs for FA. These measurements describe first attainment of a target, rather than a time after which accuracy remains permanently above it, because test accuracy can fluctuate during training. Together, the results show that initial alignment improves learning speed relative to FA while preserving final accuracy close to BP.

A.1.7 Initial alignment across alternative learning rules

We next examined whether initial alignment can improve learning rules beyond fixed-feedback FA. We compared Weight Mirror (WM), predictive coding (PC) with fixed asymmetric feedback, and Difference Target Propagation (DTP), with and without initial alignment (IA). For each learning rule, we compared the two initialization conditions while keeping the remaining training settings matched within the pair. Unlike FA and the fixed-feedback PC condition, WM and DTP continue to update their feedback weights or inverse mappings during training; IA changes their initial condition rather than making these pathways fixed.

Table S5: **Effect of initial alignment across learning rules in a scratch-trained CIFAR-10 MLP.** The network contained two hidden layers of 512 units and was trained for 30 epochs. Values are reported as the mean ± standard deviation across 10 independent runs. Accuracy is expressed in %, and improvements are the mean ± standard deviation of within-pair accuracy differences in pp. FA with IA is IFA.

Learning rule	Baseline	With IA	Improvement (pp)
Backpropagation	53.12 ± 0.48	—	—
Feedback Alignment	45.91 ± 1.12	52.34 ± 0.41	$+6.44 \pm 0.84$
Weight Mirror	51.53 ± 0.65	52.71 ± 0.29	$+1.18 \pm 0.79$
Predictive coding	45.81 ± 1.11	50.05 ± 0.52	$+4.25 \pm 1.27$
Difference Target Propagation	50.79 ± 0.22	51.27 ± 0.16	$+0.49 \pm 0.14$

In the scratch-trained CIFAR-10 MLP, initial alignment improved the mean test accuracy of all four learning rules, although the magnitude varied by algorithm (Table S5). The largest gain was observed for FA, followed by PC, WM, and DTP. Thus, the benefit of initial alignment is not limited to standard FA, but depends on how each learning rule propagates errors and adapts its feedback pathway.

Table S6: **Effect of initial alignment during ImageNet-pretrained ResNet-18 fine-tuning on CIFAR-10.** Accuracy is expressed in %, and improvements are expressed in pp. Values are reported as the mean ± standard deviation. FA with IA is IFA; PC and DTP use ResNet-specific adaptations.

Learning rule	Baseline	With IA	Improvement (pp)
Backpropagation	96.97 ± 0.13	—	—
Feedback Alignment	57.07 ± 0.75	93.66 ± 0.07	$+36.59 \pm 0.82$
Weight Mirror	77.86 ± 0.31	93.02 ± 0.05	$+15.16 \pm 0.36$
Predictive coding	76.60 ± 0.00	94.06 ± 0.08	$+17.46 \pm 0.08$
Difference Target Propagation	89.03 ± 0.18	90.03 ± 0.23	$+1.00 \pm 0.05$

To test whether this benefit extends to a stronger vision backbone, we also evaluated fine-tuning of ImageNet-pretrained ResNet-18 on CIFAR-10 (Table S6). The pretrained forward weights were retained, and the corresponding feedback pathways were aligned only at initialization. BP achieved $96.97 \pm 0.13\%$ test accuracy. Initial alignment increased FA accuracy from $57.07 \pm 0.75\%$ to $93.66 \pm 0.07\%$, and also improved WM and PC by 15.16 and 17.46 pp, respectively. DTP showed a smaller improvement of 1.00 pp. These results support initial alignment as a useful initialization strategy across the evaluated learning rules, while the remaining gaps to BP show that its benefits depend on the algorithm and training setting.

A.2 Language-model fine-tuning and resource requirements

A.2.1 Perplexity comparison across downstream tasks

To complement the task accuracies reported in Table 1, we evaluated held-out perplexity after full fine-tuning of Qwen3-1.7B and Qwen3-8B (Tables S7 and S8). Perplexity was computed over supervised answer tokens, with prompt tokens excluded from the loss. Each model was fine-tuned for three epochs using AdamW with a learning rate of 2×10^{-5}, a maximum sequence length of 512, and a microbatch size of 1 with gradient accumulation over 16 microbatches. Since target formats differ across tasks, perplexity comparisons are made within each task. The evaluation code caps each run's perplexity at 10^{10} before aggregation; results for which all three runs reach this cap are reported as diverged.

Table S7: **Held-out perplexity after Qwen3-1.7B full fine-tuning.** Lower values indicate better prediction of the supervised answer tokens. Numerical values are reported as the mean $\pm$ standard deviation across three independent runs, with perplexity capped at 10^{10} per run before aggregation. Diverged indicates that all three runs reached this cap.

Task	BP	FA	IFA
StrategyQA	3.4485 ± 0.0064	$(1.015 \pm 1.706) \times 10^5$	3.4529 ± 0.0026
BoolQ	1.1977 ± 0.0015	Diverged	1.1962 ± 0.0019
ARC-Challenge	1.3724 ± 0.0055	$(4.012 \pm 6.569) \times 10^8$	1.3717 ± 0.0018
RTE	1.0763 ± 0.0011	$(3.339 \pm 5.769) \times 10^9$	1.0802 ± 0.0035
CommonsenseQA	1.4259 ± 0.0006	Diverged	1.4267 ± 0.0030

Table S8: **Held-out perplexity after Qwen3-8B full fine-tuning on ARC-Challenge and RTE.** Values are reported as the mean $\pm$ standard deviation across three independent runs. Perplexity is conditional on supervised answer tokens and is compared within each task.

Task	BP	FA	IFA
ARC-Challenge	1.1276 ± 0.0030	$(3.143 \pm 1.252) \times 10^4$	1.1290 ± 0.0013
RTE	1.0689 ± 0.0029	$(5.368 \pm 5.137) \times 10^8$	1.0684 ± 0.0019

Across the five Qwen3-1.7B tasks, IFA achieved perplexities close to BP, while FA produced much larger values and reached the evaluation cap on BoolQ and CommonsenseQA. The same pattern held for Qwen3-8B: IFA closely matched BP on ARC-Challenge and RTE, whereas FA showed substantially higher perplexity. These results complement the task-accuracy comparisons and indicate that initial alignment preserves effective answer-token prediction during full fine-tuning across the evaluated model sizes.

A.2.2 Fine-tuning with low-rank adaptation

We next examined whether initial alignment also supports parameter-efficient fine-tuning using low-rank adaptation (LoRA). We fine-tuned Qwen3-1.7B on ARC-Challenge and RTE with rank-8 adapters, a scaling parameter of 16, dropout of 0.05, and a learning rate of 2×10^{-4} for three epochs. The pretrained backbone, embeddings, and language-model head remained frozen. For FA and IFA, backward input-gradient paths in the frozen backbone, language-model head, and both low-rank factors used fixed feedback matrices. FA initialized these matrices as shuffled copies of the corresponding initial forward weights, whereas IFA retained their initial alignment. All methods started with the same pretrained output because the down factor was initialized to zero and the up factor was nonzero.

Table S9: **Task accuracy after Qwen3-1.7B LoRA fine-tuning.** BP, FA, and IFA values (%, higher is better) are reported as the mean $\pm$ standard deviation across three independent runs. Base denotes the pretrained baseline evaluated once before adapter training. LoRA uses rank 8, a scaling parameter of 16, and dropout of 0.05.

		Fine-tuned		
Task	Base	BP	FA	IFA
ARC-Challenge	71.76	79.58 ± 0.94	0.00 ± 0.00	79.69 ± 0.23
RTE	39.35	87.97 ± 2.12	0.00 ± 0.00	88.21 ± 0.21

Table S10: **Held-out perplexity after Qwen3-1.7B LoRA fine-tuning.** Perplexity is computed over supervised answer tokens. Numerical values are reported as the mean $\pm$ standard deviation across three independent runs, with perplexity capped at 10^{10} per run before aggregation. Diverged indicates that all three runs reached this cap.

Task	BP	FA	IFA
ARC-Challenge	1.3666 ± 0.0202	Diverged	1.3660 ± 0.0066
RTE	1.0745 ± 0.0048	Diverged	1.0739 ± 0.0018

IFA achieved task accuracies close to BP on both benchmarks (Table S9) and similarly low perplexities (Table S10). FA yielded zero task accuracy and perplexity at the evaluation cap in both cases. Thus, the benefit of initially aligned fixed feedback extends to the evaluated LoRA setting, where only a small set of adapter parameters is trained.

A.2.3 Feedback weight transport during fine-tuning

To quantify the cost of maintaining a feedback pathway, we examined Qwen3-8B full fine-tuning on ARC-Challenge under a hardware model with physically separate forward and feedback arrays. In this setting, BP requires the feedback array to track the current forward weights, whereas FA and IFA program a fixed feedback array once. We distinguished forward-to-feedback transport, which copies forward-weight information into the feedback pathway, from feedback-array writes, which program the array regardless of where its values originate (Table S11).

Table S11: **Feedback transport and array programming for Qwen3-8B full fine-tuning on ARC-Challenge.** The run comprises 210 optimizer steps. A pass copies or programs the complete feedback array. BP programs the initial transpose and refreshes it for each subsequent backward pass; FA initializes random feedback locally, and IFA copies the initial forward transpose once. Byte counts are calculated from feedback tensor sizes and step counts under the separate-array assumption. Transport reduction is relative to BP.

Method	Transport passes	Programming passes	Transported bytes	Feedback-array write bytes	Transport reduction (%)
BP	210	210	2.891 TiB	2.891 TiB	—
FA	0	1	0 B	14.097 GiB	100.000
IFA	1	1	14.097 GiB	14.097 GiB	99.524

IFA required one initial transport and programming pass, amounting to 14.097 GiB, compared with 2.891 TiB over the full BP run. This corresponds to a 99.524% reduction in forward-to-feedback transport. FA required the same one-time array programming volume as IFA but no forward-weight transport because its feedback values were generated locally. The initial alignment operation requires no additional training examples or alignment-optimization steps and has a cost linear in the number of feedback weights. These calculations characterize synchronization and programming traffic for separate pathways; they do not measure energy consumption or memory traffic on the GPUs used to evaluate learning performance.

A.2.4 GPU memory usage during fine-tuning

We also examined the measured peak GPU memory usage during the same Qwen3-8B fine-tuning task on an NVIDIA H200 (Table S12). In this dense GPU implementation, FA and IFA explicitly store fixed feedback matrices in addition to the forward weights. Their peak memory usage was therefore higher than that of BP.

Table S12: **Measured peak GPU memory usage during Qwen3-8B full fine-tuning on ARC-Challenge using an NVIDIA H200.** Values are mean peak high-bandwidth memory usage across three runs, obtained from Slurm GPU-memory accounting.

Method	Peak GPU memory (GiB)
BP	65.39
FA	79.49
IFA	79.50

IFA used approximately 14.1 GiB more peak GPU memory than BP, consistent with the additional fixed feedback array. Thus, eliminating repeated feedback synchronization does not reduce the memory capacity required by the present GPU implementation. The transport advantage in Table S11 concerns hardware with separate forward and feedback pathways, where repeated synchronization and array programming carry a distinct cost.

A.3 Stabilizing learning through initial weight alignment

A.3.1 Training networks across the parameter space

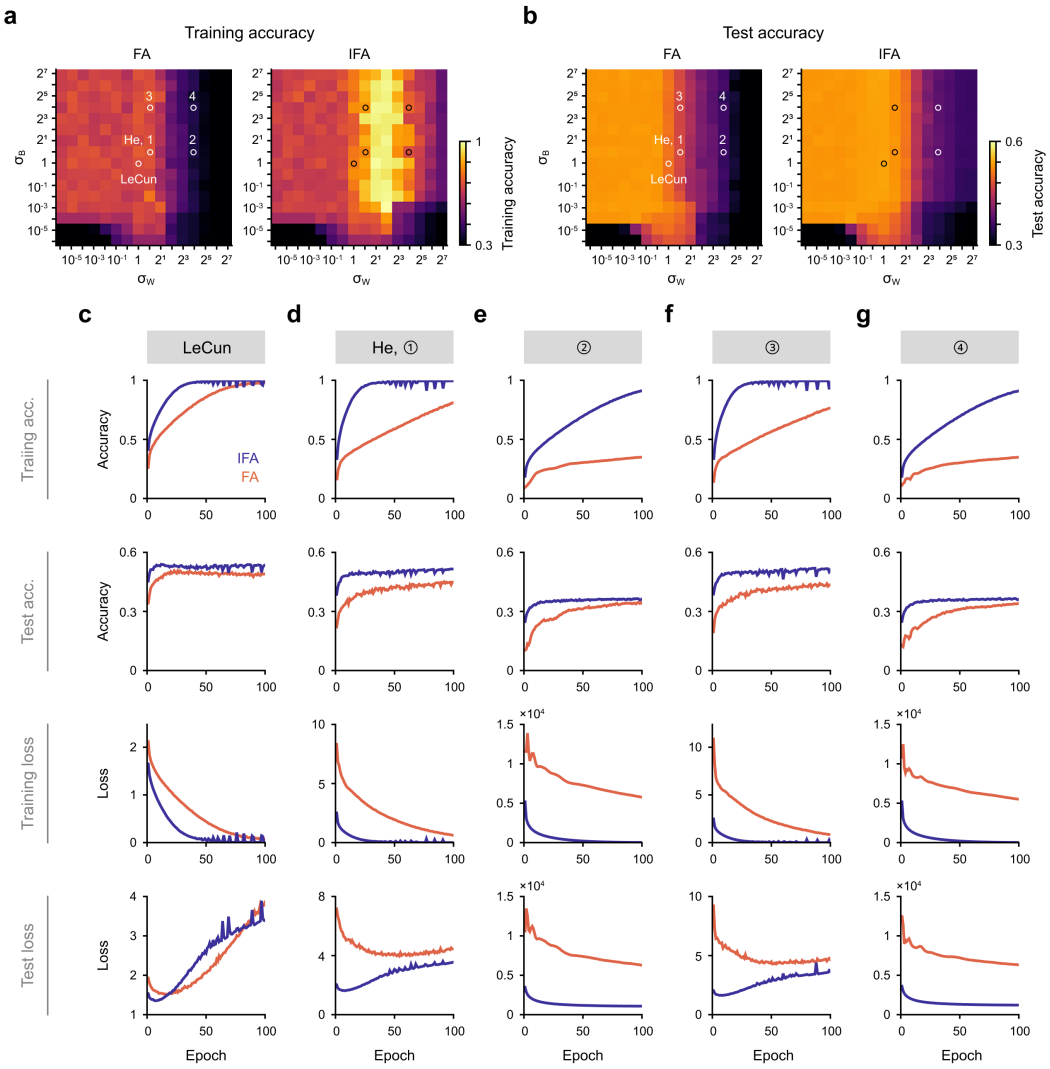


Figure S5: **Trainability and generalization across the parameter space. a,** Training accuracy across the parameter space, with color indicating the final training accuracy. **b,** Test accuracy across the parameter space, with color indicating the final test accuracy. The x- and y-axes represent the variances of the forward and backward weights, respectively. The point $(1, 1)$ corresponds to LeCun initialization in panel **c**, and $(\sqrt{2}, \sqrt{2})$ corresponds to He initialization in panel **d** (①). Additionally, we investigate several representative conditions to visualize learning curves: large forward weight variance at $(2^4, \sqrt{2})$ in panel **e** (②), large backward weight variance at $(\sqrt{2}, 2^4)$ in panel **f** (③), and large variance in both weights at $(2^4, 2^4)$ in panel **g** (④). For each variance condition, learning curves of training accuracy, test accuracy, training loss, and test loss are presented from top to bottom.

In the main results, we reported the trainability of networks using FA and IFA across a parameter space defined by a two-dimensional grid of varying forward and backward weight variances (Figure 4a; Figure S5a). We observed that baseline FA supports only a narrow range of variance settings for achieving high training accuracy, whereas IFA enables a broader trainable parameter space. To extend this analysis, we also evaluated generalization by measuring test accuracy across the same parameter space (Figure S5b). Consistent with the trainability results, we found that initial alignment

significantly expands the range of variance settings that yield high test accuracy. Furthermore, IFA consistently outperformed FA in nearly all regions of the parameter space.

To further characterize the learning behavior of FA and IFA, we selected five representative parameter settings and visualized the corresponding learning curves (Figure S5c–g). We first examined standard initialization methods: LeCun initialization (Figure S5c, $(1, 1)$) and He initialization (Figure S5d, $(\sqrt{2}, \sqrt{2})$, ①), confirming that IFA achieves faster convergence and higher performance in both training and test accuracy. We then extended the analysis to more extreme conditions, including large forward weight variance (Figure S5e, $(2^4, \sqrt{2})$, ②), large backward weight variance (Figure S5f, $(\sqrt{2}, 2^4)$, ③), and large variance in both forward and backward weights (Figure S5g, $(2^4, 2^4)$, ④). While FA exhibited extreme instability under these conditions, IFA demonstrated robust learning dynamics, with improved trainability and generalization.

A.3.2 Training networks with varying depth

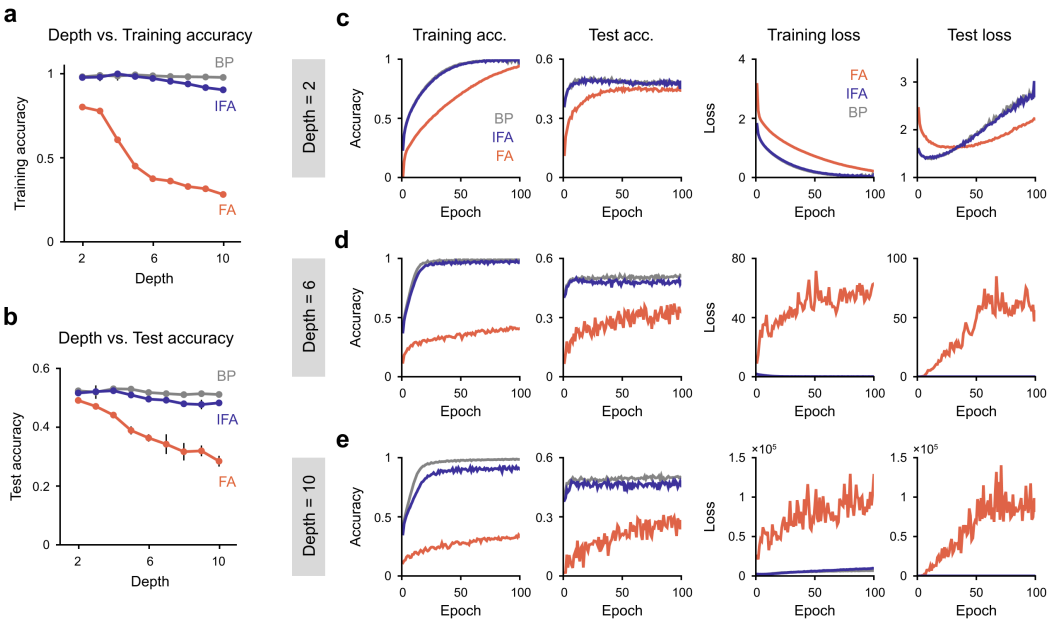


Figure S6: **Trainability and generalization across different network depths. a,** Training accuracy as a function of network depth, for feedforward networks with depths ranging from 2 to 10 layers. **b,** Test accuracy as a function of network depth, using the same range. **c–e,** Learning curves for representative depths: 2 in panel **c**, 6 in panel **d**, and 10 in panel **e**. For each depth, training accuracy, test accuracy, training loss, and test loss are shown from left to right.

We also extended our trainability analysis to networks with varying depths (Figure 4b; Figure S6a) and benchmarked test accuracy to assess generalization performance (Figure S6b). Consistent with the trainability results, IFA achieved significantly higher test accuracy than baseline FA and performed comparably to BP. Notably, FA exhibited a steep decline in test performance as network depth increased, whereas IFA maintained high test accuracy even in deeper architectures. To further investigate this trend, we selected three representative network depths and visualized the corresponding learning curves: depth 2 (Figure S6c), depth 6 (Figure S6d), and depth 10 (Figure S6e). As depth increased, baseline FA showed highly unstable learning dynamics — loss values diverged, and accuracy fluctuated significantly. In contrast, IFA demonstrated robust and stable learning, closely matching the behavior of standard BP. Interestingly, in deeper networks, FA's loss continued to increase even as accuracy slightly improved. This discrepancy between loss and accuracy suggests a breakdown in model calibration: the network's confidence estimates become misaligned with actual prediction performance.

To examine how depth affects the approximation to BP, we additionally trained networks with 2, 6, or 10 hidden layers, each containing 512 units, for 100 epochs. For each depth, we evaluated initial alignment angles of 0°, 30°, and 60° and measured the cosine similarity between IFA and exact BP weight gradients at the same forward parameters. We report the earliest, middle, and last hidden-layer gradients separately (Table S13). The output-layer gradient is excluded from these comparisons because, at the same forward parameters, its update has the same form under BP, FA, and IFA.

Table S13: **Accuracy and layerwise IFA–BP gradient alignment across network depths and initial alignment angles at epoch 100.** Values are reported as the mean $\pm$ standard deviation across five independent runs. Depth is the number of hidden layers. Gradient cosines are computed on fixed training probes using the same forward parameters for both learning rules. The middle hidden-layer index is $\lfloor (d-1)/2 \rfloor$ for zero-based indexing and depth d; it coincides with the earliest layer at depth 2.

Depth	Initial angle	Test accuracy (%)	Earliest hidden cosine	Middle hidden cosine	Last hidden cosine
2	0°	52.17 ± 2.41	0.878 ± 0.014	0.878 ± 0.014	0.965 ± 0.002
2	30°	52.01 ± 0.54	0.842 ± 0.007	0.842 ± 0.007	0.940 ± 0.004
2	60°	50.59 ± 0.89	0.776 ± 0.054	0.776 ± 0.054	0.837 ± 0.023
6	0°	49.57 ± 0.56	0.754 ± 0.013	0.872 ± 0.015	0.985 ± 0.002
6	30°	49.17 ± 0.65	0.671 ± 0.028	0.786 ± 0.022	0.875 ± 0.021
6	60°	46.31 ± 1.22	0.735 ± 0.023	0.619 ± 0.064	0.494 ± 0.062
10	0°	47.95 ± 0.94	0.679 ± 0.024	0.856 ± 0.020	0.986 ± 0.003
10	30°	48.26 ± 1.01	0.686 ± 0.036	0.751 ± 0.025	0.852 ± 0.014
10	60°	35.11 ± 2.23	0.286 ± 0.071	0.155 ± 0.073	0.327 ± 0.082

At exact initial alignment, the earliest hidden-layer cosine decreased from 0.878 at depth 2 to 0.679 at depth 10, while the last hidden-layer cosine remained high. At an initial angle of 60°, the depth-10 network showed a pronounced decrease in both accuracy and gradient alignment across all three reported layers. These results suggest that the approximation is especially sensitive to the length of the feedback path and the initial mismatch, although the dependence is not strictly monotonic for every layer and angle. The accumulation of layerwise mismatch is analyzed in Section A.6.

A.3.3 Training with limited data

Next, we investigated the effect of limited training data size (Figure 4c; Figure S7a) on generalization performance (Figure S7b). We benchmarked the test accuracy of networks trained on varying amounts of data. Notably, baseline FA consistently exhibited poor test performance compared to BP when trained on the same number of samples. In contrast, IFA demonstrated substantially improved generalization over FA, often achieving performance levels close to those of BP. The relationship between dataset size and test accuracy is frequently discussed in the context of neural scaling laws. Within this framework, initial alignment effectively shifts the scaling curve upward, enabling improved performance at every dataset size. To further examine this behavior, we selected three representative training data conditions and visualized the corresponding learning curves: 100 samples (Figure S7c), 1,000 samples (Figure S7d), and 10,000 samples (Figure S7e). Baseline FA notably struggled to learn under limited data, likely due to its reliance on data-driven alignment over time. In contrast, IFA produced stable and effective learning dynamics, even in data-scarce regimes.

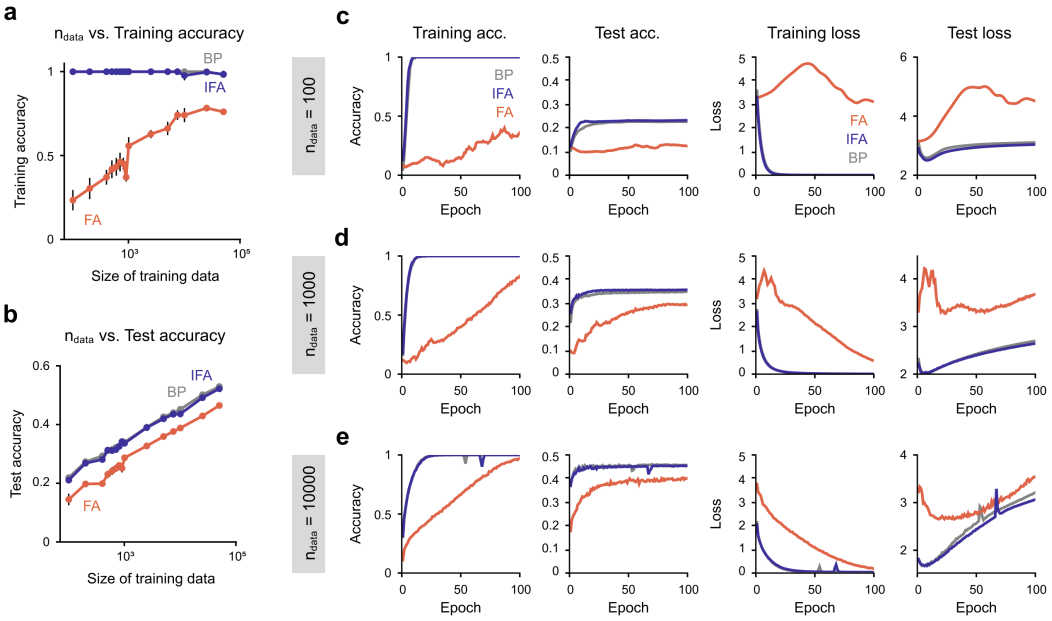


Figure S7: **Trainability and generalization under limited data. a,** Training accuracy as a function of training data size, for feedforward networks trained on datasets ranging from 100 to 50,000 samples. **b,** Test accuracy as a function of training data size, using the same range. **c–e,** Learning curves for representative data sizes: 100 samples in panel **c**, 1,000 samples in panel **d**, and 10,000 samples in panel **e**. For each data size, training accuracy, test accuracy, training loss, and test loss are shown from left to right.

A.3.4 Effects of network width

To examine whether increasing width improves learning with fixed feedback, we trained networks with six hidden layers and widths ranging from 32 to 2048 units for 100 epochs. We compared BP, FA, and IFA at initial alignment angles of 0° and 30° (Table S14). We quantified recovery of the BP–FA performance gap as $100 \times (\mathrm{Acc}_{\mathrm{IFA}} - \mathrm{Acc}_{\mathrm{FA}})/(\mathrm{Acc}_{\mathrm{BP}} - \mathrm{Acc}_{\mathrm{FA}})$ and measured gradient alignment and relative parameter drift for exactly aligned IFA (Table S15).

Table S14: **Test accuracy across network widths at depth 6 and epoch 100.** Values (%) are reported as the mean $\pm$ standard deviation across five independent runs. IFA was initialized at alignment angles of 0° or 30°.

Width	BP	FA	IFA (0°)	IFA (30°)
32	45.27 ± 0.25	26.32 ± 2.60	40.93 ± 0.25	41.89 ± 0.56
64	45.11 ± 0.40	29.88 ± 2.34	42.68 ± 0.44	43.12 ± 0.22
128	45.65 ± 0.40	34.55 ± 1.03	43.70 ± 0.46	44.06 ± 0.30
512	51.97 ± 0.33	36.65 ± 3.79	49.57 ± 0.56	49.17 ± 0.65
2048	55.20 ± 0.17	44.27 ± 2.46	49.25 ± 1.06	48.76 ± 1.45

Table S15: **BP–FA gap recovery, gradient alignment, and parameter drift across network widths for IFA initialized at 0°.** Values at depth 6 and epoch 100 are reported as the mean $\pm$ standard deviation across five independent runs. Gap recovery is computed within each run before averaging. Gradient cosine is computed from concatenated hidden-weight gradients, excluding the output layer and biases, at the current forward parameters. Relative parameter drift is normalized by the initial forward-parameter norm.

Width	Gap recovery (%)	IFA–BP gradient cosine	Relative parameter drift
32	76.82 ± 2.97	0.906 ± 0.010	0.467 ± 0.004
64	83.86 ± 5.38	0.920 ± 0.008	0.461 ± 0.002
128	82.42 ± 3.02	0.872 ± 0.004	0.486 ± 0.001
512	83.81 ± 6.00	0.887 ± 0.012	0.506 ± 0.001
2048	44.68 ± 7.94	0.812 ± 0.021	0.806 ± 0.001

IFA recovered approximately 77–84% of the BP–FA gap at widths 32–512, but recovery decreased to $44.68 \pm 7.94\%$ at width 2048. At this largest width, relative parameter drift increased to 0.806 ± 0.001 and the IFA–BP gradient cosine decreased to 0.812 ± 0.021. Thus, wider networks did not monotonically improve the approximation to BP. The joint changes in performance, drift, and gradient alignment indicate that width affects the learning dynamics as well as the capacity of the network.

A.3.5 Parameter drift and representation changes during learning

We next examined whether the effectiveness of IFA requires the forward parameters to remain close to their initialization. Using the 150-epoch CIFAR-10 MLP experiment, we measured relative parameter drift, $\|\theta_t - \theta_0\|_2 / \|\theta_0\|_2$, where θ_t contains the forward parameters at epoch t. We also measured linear centered kernel alignment (CKA) between the initial and current hidden representations on fixed training probes and averaged it across hidden layers. CKA equals 1 for maximally similar representations under this measure. Table S16 summarizes the final values across methods, and Table S17 shows the evolution of these quantities under IFA.

Table S16: **Parameter drift and representation similarity at epoch 150.** Values are reported as the mean $\pm$ standard deviation across 10 independent runs. Relative forward-parameter drift and mean hidden-representation CKA use the initial state of the corresponding run as the reference. Feedback weights are excluded from parameter drift.

Method	Relative parameter drift	Hidden-representation CKA
BP	0.746 ± 0.007	0.741 ± 0.020
FA	1.304 ± 0.025	0.911 ± 0.012
IFA	0.762 ± 0.006	0.811 ± 0.018

Table S17: **Evolution of parameter drift and representation similarity under IFA initialized at $0°$.** Values are reported as the mean $\pm$ standard deviation across 10 independent runs. Both measures use the initial state as the reference. Hidden-representation CKA is evaluated on the same fixed training probes throughout learning, and feedback weights are excluded from parameter drift.

Epoch	Relative parameter drift	Hidden-representation CKA
0	0.000 ± 0.000	1.000 ± 0.000
1	0.061 ± 0.000	0.932 ± 0.009
10	0.216 ± 0.000	0.850 ± 0.018
30	0.414 ± 0.000	0.829 ± 0.019
100	0.679 ± 0.004	0.816 ± 0.024
150	0.762 ± 0.006	0.811 ± 0.018

IFA remained close to BP in test accuracy despite a relative parameter drift of 0.762 ± 0.006 at epoch 150. Its hidden-representation CKA decreased from 1 at initialization to 0.811 ± 0.018, indicating that the representations also changed during learning. FA exhibited larger parameter drift but higher representation similarity, showing that these two measurements capture different aspects of learning. In this scratch-trained MLP, distance from initialization alone therefore does not explain the performance of IFA.

A.3.6 Alignment with the current backpropagation gradient

To determine whether IFA continues to provide a useful learning signal as the forward parameters change, we compared its update direction with the exact BP gradient evaluated at the current forward parameters. On fixed training probes, we computed $C_{\mathrm{init}}(t) = \cos(g_{\mathrm{BP}}(\theta_0), g_{\mathrm{BP}}(\theta_t))$, which measures rotation of the BP gradient from its initial direction, and $C_{\mathrm{rule}}(t) = \cos(g_{\mathrm{rule}}(\theta_t), g_{\mathrm{BP}}(\theta_t))$, which measures alignment of the learning rule with the current exact gradient. Each gradient vector concatenates the hidden-weight gradients, excluding the output layer and biases. We compared the final cosines across methods (Table S18) and their evolution under IFA (Table S19).

Table S18: **Alignment with the current BP gradient at epoch 150.** Values are reported as the mean $\pm$ standard deviation across 10 independent runs on fixed training probes. The exact BP gradient is recomputed at each model's own forward parameters. C_{init} compares the initial and current BP gradients, and C_{rule} compares the learning-rule and current BP gradients. The output layer and biases are excluded from all reported cosines.

Method	C_{init}	C_{rule}
BP	0.002 ± 0.059	1.000 ± 0.000
FA	-0.005 ± 0.058	0.540 ± 0.035
IFA	0.003 ± 0.045	0.914 ± 0.005

Table S19: **Evolution of gradient alignment under IFA initialized at 0°.** Values are reported as the mean $\pm$ standard deviation across 10 independent runs on fixed training probes. C_{init} compares the initial and current BP gradients, and C_{rule} compares the IFA and current BP gradients. The exact BP gradient is recomputed at each current forward state; the output layer and biases are excluded.

Epoch	C_{init}	C_{rule}
0	1.000 ± 0.000	1.000 ± 0.000
1	0.083 ± 0.058	0.990 ± 0.001
10	0.052 ± 0.054	0.948 ± 0.005
30	0.056 ± 0.077	0.927 ± 0.005
100	0.011 ± 0.057	0.917 ± 0.006
150	0.003 ± 0.045	0.914 ± 0.005

By epoch 150, the current BP gradient along the IFA trajectory was nearly orthogonal to its initial direction, with $C_{\mathrm{init}} = 0.003 \pm 0.045$. In contrast, the IFA update remained strongly aligned with the current BP gradient, with $C_{\mathrm{rule}} = 0.914 \pm 0.005$. This alignment remained above 0.9 at every reported time point (Table S19), even as parameter drift increased and hidden representations changed (Table S17). Together, these observations show that useful alignment with the current gradient can persist after substantial departure from the initial parameters and gradient direction.

A.4 Initial alignment enhances robust generalization

A.4.1 Layer-wise dynamics of weight alignment

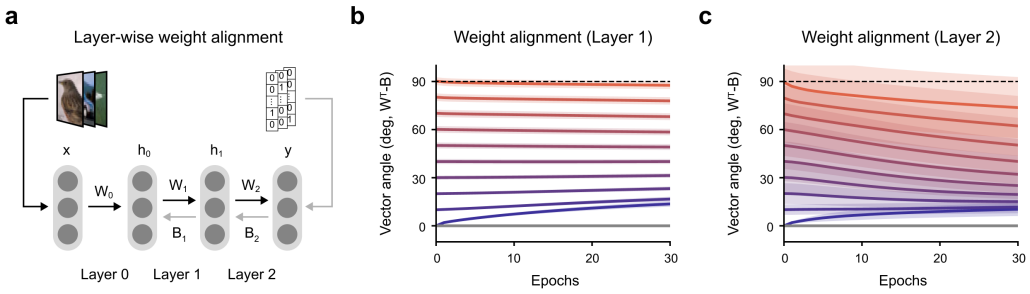


Figure S8: **Effect of initial weight alignment on alignment dynamics during learning. a,** Schematic of a three-layer feedforward neural network. Forward and backward weights, excluding those in the first layer (Layer 0), are initialized with a soft alignment, ranging from $0°$ to $90°$. An initial angle of $90°$ corresponds to the baseline feedback alignment. **b–c,** Alignment angles between forward and backward weights in each layer. **b,** Weight alignment in Layer 1. Shaded areas indicate the standard deviation of alignment angles across 512 neurons. **c,** Weight alignment in Layer 2, the final layer of the network. Shaded areas indicate the standard deviation of alignment angles across 512 neurons.

In the main results (Figure 5b), we visualized the alignment dynamics in the penultimate layer under various initial alignment conditions. We further extended this analysis to all layers (Figure S8a). With the exception of the first layer, we examined the angle between forward and backward weight vectors in both the second layer (Figure S8b) and the final layer (Figure S8c). Consistent with our main findings, the learning dynamics were highly dependent on the initial alignment. When the initial alignment was strong, the network gradually decreased the alignment during training. In contrast, when the initial alignment was weak, the network progressively increased the alignment through learning.

A.4.2 Training details for different learning models

Next, we examined how initial alignment influences neural network learning. To isolate its effect, we visualized learning curves across a range of initial alignment angles while keeping all other experimental conditions constant. Specifically, we plotted training accuracy (Figure S9a), training loss (Figure S9b), test accuracy (Figure S9c), and test loss (Figure S9d) over the course of training. Our results show that although baseline FA performs poorly compared to BP, increasing the initial alignment markedly improves both learning speed and performance. As the alignment increases, the learning curves progressively resemble those of standard BP.

We additionally quantified how the initial alignment angle affects the approximation to BP in a network with six hidden layers of 512 units trained for 100 epochs (Table S20). Gradient cosines were measured using concatenated hidden-weight gradients at the same forward parameters, excluding the output layer and biases. For the Gaussian initialization used here, the normalized expected feedback mismatch is $2(1 - \cos\theta_{\text{init}})$, as derived in Section A.6.

Exact initial alignment produced the same initial gradient as BP. As the initial angle increased, the expected feedback mismatch increased and the initial gradient cosine decreased. This ordering persisted after training: the final gradient cosine decreased from 0.887 at $0°$ to 0.102 at $90°$, accompanied by a decrease in test accuracy from 49.57% to 36.65%. Thus, initial alignment controls the quality of the initial learning signal and is associated with improved gradient alignment after training. The initial mismatch alone does not determine final performance, which also depends on the subsequent evolution of the network.

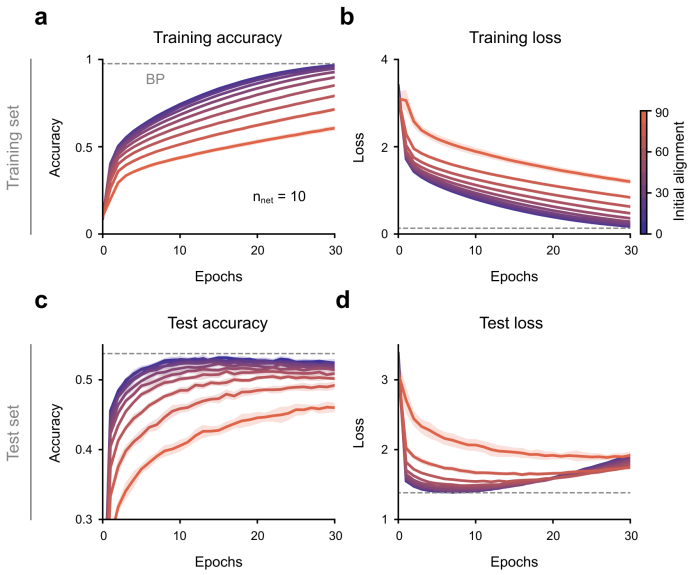


Figure S9: **Learning curves across various initial alignment angles. a–d,** Learning curves over the course of training. **a,** Training accuracy. **b,** Training loss. **c,** Test accuracy. **d,** Test loss. The dotted gray horizontal line indicates the final performance of BP. Each solid line represents the learning curve of networks with a different initial alignment angle. Shaded areas indicate the standard deviation across ten independent trials.

Table S20: **Initial alignment angle, feedback mismatch, and learning performance at depth 6.** Empirical values are reported as the mean $\pm$ standard deviation across five independent runs. Expected mismatch is the ratio $\mathbb{E}\|\mathbf{W}_l(0)^\top - \mathbf{B}_l\|_F^2 / \mathbb{E}\|\mathbf{B}_l\|_F^2$ under the Gaussian initialization. Initial and final gradient cosines compare the learning-rule and exact BP hidden-weight gradients at epochs 0 and 100, respectively. Test accuracy is measured at epoch 100.

Initial angle	Expected mismatch	Initial IFA–BP gradient cosine	Final IFA–BP gradient cosine	Test accuracy (%)
$0°$	0.000	1.000 ± 0.000	0.887 ± 0.012	49.57 ± 0.56
$30°$	0.268	0.708 ± 0.030	0.817 ± 0.017	49.17 ± 0.65
$60°$	1.000	0.253 ± 0.028	0.451 ± 0.052	46.31 ± 1.22
$90°$ (FA)	2.000	-0.004 ± 0.025	0.102 ± 0.055	36.65 ± 3.79

A.4.3 Extended spectral analysis

To further investigate how initial weight alignment influences the convergence behavior of neural networks, we conducted a spectral analysis of the loss landscape (Figure S10). For each trained model, we visualized the local geometry of the loss surface by perturbing the parameters within a two-dimensional subspace spanned by the top two eigenvectors of the Hessian matrix. Specifically, we plotted the loss as a function of perturbations along these principal curvature directions, denoted by α and β, to assess the flatness and sharpness of the local minima. In addition, we examined the Hessian spectrum by plotting the eigenvalue density distribution. We first observed that BP exhibits a relatively flat loss landscape and a narrow eigenvalue spectrum (Figure S10a), in contrast to baseline FA, which shows a sharper landscape and a broader spectrum (Figure S10b). However, as the initial alignment improved (i.e., as the alignment angle decreased), the loss landscape became progressively flatter, and the Hessian spectra became increasingly concentrated within a narrower range (Figures S10c–k). These findings suggest that stronger initial alignment leads to smoother loss surfaces and more favorable curvature properties, ultimately enhancing robustness and generalization.

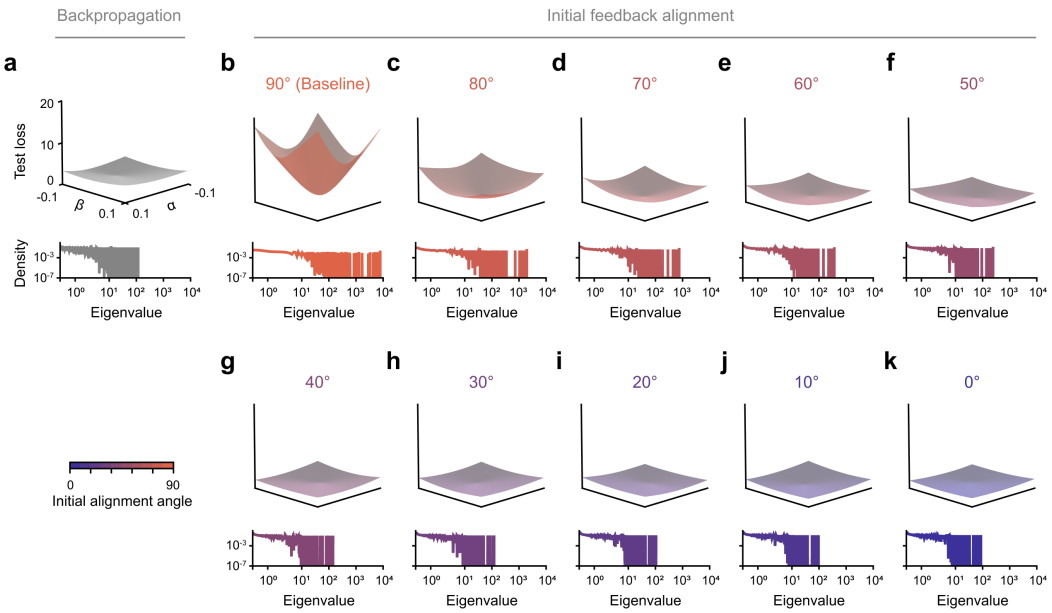


Figure S10: **Spectral analysis of convergence behavior across initial alignment angles. a–k,** Loss landscapes visualized by perturbing the trained model parameters within a two-dimensional subspace defined by the top two eigenvectors of the Hessian (top). The axes, denoted by α and β, correspond to scaling factors for perturbations along the first and second eigenvector directions, respectively. Below each landscape, the spectral density of the Hessian eigenvalues is shown (bottom). **a,** BP. **b–k,** IFA with varying initial alignment angles ranging from $0°$ to $90°$. An initial angle of $90°$ in panel **b** corresponds to the baseline feedback alignment.

A.4.4 Robustness under input corruption

To further assess the impact of initial weight alignment on model robustness, we systematically evaluated performance under varying levels of input corruption (Figure S11). Following a standard image corruption benchmark [37], we applied fifteen corruption types—categorized into noise, blur, weather, and digital—to the test set, each at five severity levels ranging from level 1 (least severe) to level 5 (most severe) (Figure S11a). While results for severity level 1 were presented in the main text (Figure 7b,c), here we report extended results for severity levels 2 through 5 (Figures S11b–e). We compared the corruption robustness of networks initialized with different alignment conditions by measuring their classification accuracy on both clean and corrupted inputs. Consistent with our main results, FA showed substantially lower robustness compared to standard BP. However, increasing the degree of initial alignment significantly improved robustness, bringing performance closer to that of BP. Although all models exhibited performance degradation as corruption severity increased, the decline was significantly less pronounced in networks with stronger initial alignment. These results highlight the robustness benefits conferred by initial alignment—networks with better-aligned initial weights demonstrated greater resilience to input corruptions, suggesting that alignment at initialization helps stabilize learning and supports more reliable performance under distributional shifts. This observation aligns with our earlier spectral analysis of the loss landscape, which showed that stronger initial alignment leads to flatter optima.

A.5 The role of weight misalignment in promoting adversarial robustness

A.5.1 Adversarial robustness to Basic Iterative Method (BIM)

In the main results, we demonstrated that the misalignment introduced by IFA enhances robustness against adversarial attacks, particularly using the Fast Gradient Sign Method (FGSM) (Figure 8). To further support this finding and extend the robustness of IFA to other gradient-based attacks, we also evaluated performance under the Basic Iterative Method (BIM) [46]. BIM is an extension of FGSM that applies small, iterative perturbations in the direction of the gradient of the loss with respect to the

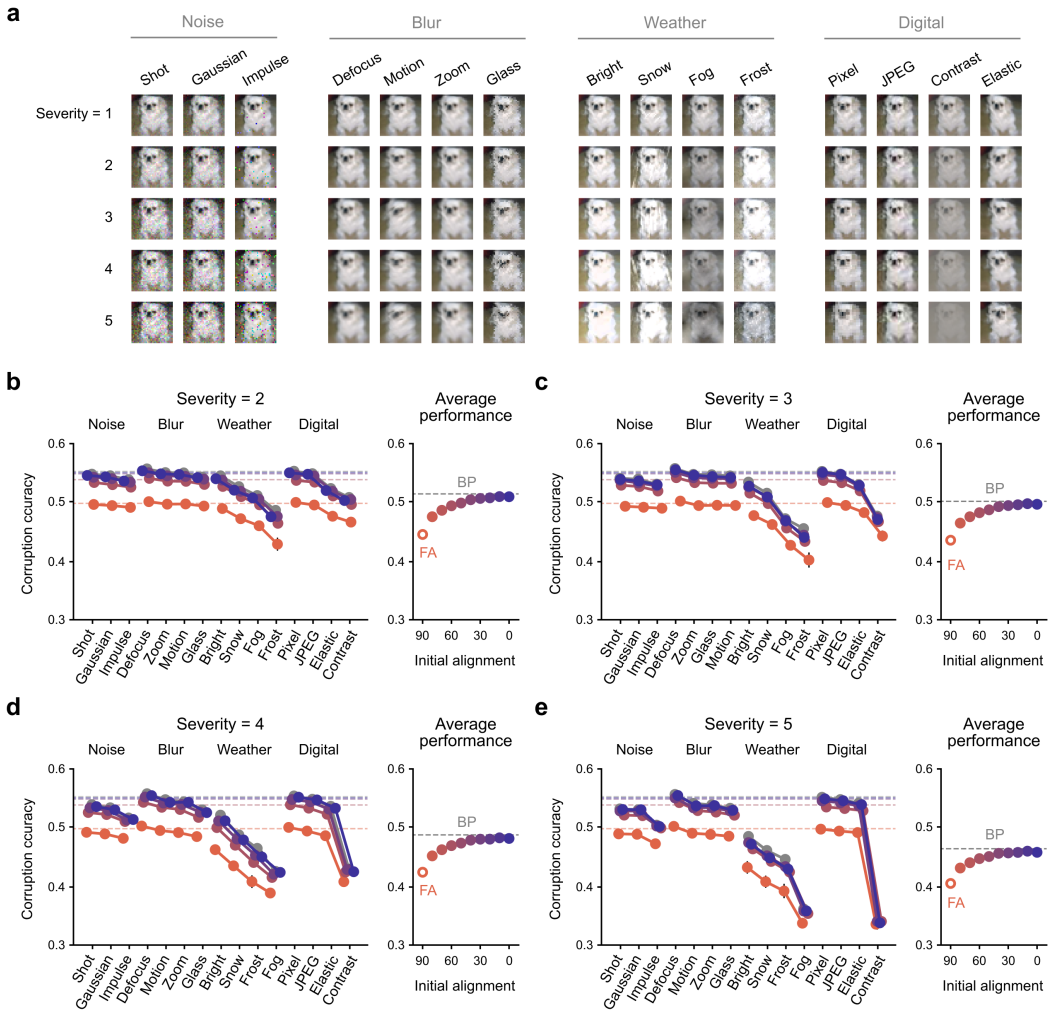


Figure S11: **Effect of initial weight alignment on robustness to input corruption of varying severity. a,** Fifteen image corruption types, grouped into four categories—noise, blur, weather, and digital—were applied to test samples to assess network robustness. Each corruption type was tested across five severity levels, ranging from 1 (least severe) to 5 (most severe). **b–e,** Classification accuracy on corrupted inputs (solid lines with markers) compared to accuracy on clean, uncorrupted images (dotted lines) (left). The relationship between initial alignment angle and robustness, measured as the average accuracy across all corruption types, is shown on the right. **b,** Severity 2 (Severity 1 is shown in the main results). **c,** Severity 3. **d,** Severity 4. **e,** Severity 5.

input. At each step, the perturbation is constrained within a defined epsilon-ball around the original input to preserve perceptual similarity. We applied BIM to networks trained with various initial alignment angles and measured classification accuracy under increasing adversarial perturbation (Figure S12a). Consistent with our FGSM results, model accuracy under BP rapidly declined as perturbation strength increased, whereas models trained with IFA maintained significantly higher accuracy even under stronger attacks (Figure S12b).

A.5.2 Adversarial robustness to Projected Gradient Descent (PGD)

We also evaluated robustness against the Projected Gradient Descent (PGD) attack [47], widely considered one of the most effective first-order adversarial attacks. PGD is an iterative extension of the Basic Iterative Method (BIM) that incorporates a projection step to ensure that adversarial examples remain within an ϵ-bounded ℓ_∞-ball around the original input. At each iteration, the input

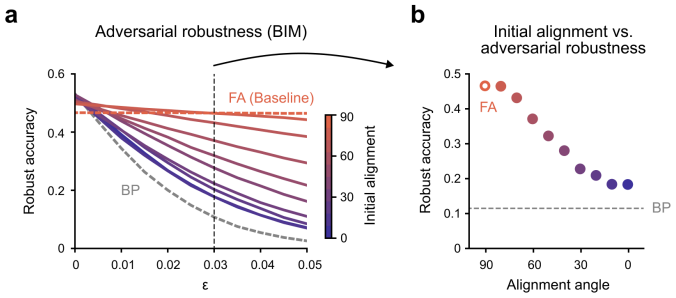


Figure S12: **Effect of weight misalignment on adversarial robustness to the Basic Iterative Method (BIM). a,** Classification accuracy as a function of increasing adversarial perturbation strength under attack using the BIM. **b,** Relationship between the initial alignment angle and adversarial robustness, measured as classification accuracy under attack with perturbation magnitude $\epsilon = 0.03$.

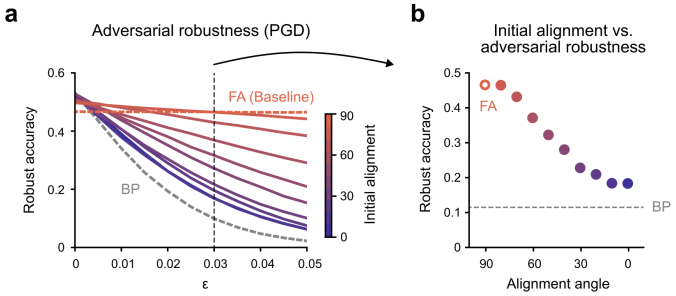


Figure S13: **Effect of weight misalignment on adversarial robustness to the Projected Gradient Descent (PGD). a,** Classification accuracy as a function of increasing adversarial perturbation strength under attack using the PGD. **b,** Relationship between the initial alignment angle and adversarial robustness, measured as classification accuracy under attack with perturbation magnitude $\epsilon = 0.03$.

is updated by ascending the gradient of the loss with respect to the input, followed by a projection back into the allowed perturbation region if the update exceeds the specified bound. We applied PGD to networks with varying initial alignment angles and measured their accuracy under attack (Figure S13a). Consistent with the results from FGSM and BIM, IFA-trained models showed significantly greater resilience to PGD compared to those trained with standard BP (Figure S13b). These findings reinforce the role of initial alignment in enhancing both generalization and robustness to strong adversarial attacks. In particular, they suggest that greater initial misalignment further improves adversarial robustness, highlighting a key functional advantage of IFA over standard BP.

A.6 Theoretical analysis of initial feedback alignment

A.6.1 Initial alignment and gradient approximation

To provide intuition for the effect of initial alignment, we compare the error signals produced by BP and IFA at the same forward parameters and activations. Let $\mathbf{W}_l \in \mathbb{R}^{n_{l+1} \times n_l}$ map the activity of layer l to the preactivation of layer $l+1$, and let $\mathbf{B}_l \in \mathbb{R}^{n_l \times n_{l+1}}$ denote the fixed feedback matrix. For a single input, define $\mathbf{D}_l = \mathrm{diag}(\phi'(\mathbf{o}_l))$ at hidden layer l. Both learning rules share the output error $\boldsymbol{\delta}_L$, while the hidden-layer errors satisfy

$$\boldsymbol{\delta}_l^{\mathrm{BP}} = \mathbf{D}_l \mathbf{W}_l^\top \boldsymbol{\delta}_{l+1}^{\mathrm{BP}}, \qquad \boldsymbol{\delta}_l^{\mathrm{IFA}} = \mathbf{D}_l \mathbf{B}_l \boldsymbol{\delta}_{l+1}^{\mathrm{IFA}}. \tag{5}$$

The IFA error is a surrogate learning signal. Its discrepancy from BP, $\mathbf{e}_l = \boldsymbol{\delta}_l^{\mathrm{IFA}} - \boldsymbol{\delta}_l^{\mathrm{BP}}$, obeys the exact recursion

$$\mathbf{e}_l = \mathbf{D}_l \left[\mathbf{B}_l \mathbf{e}_{l+1} + (\mathbf{B}_l - \mathbf{W}_l^\top) \boldsymbol{\delta}_{l+1}^{\mathrm{BP}} \right], \qquad \mathbf{e}_L = \mathbf{0}. \tag{6}$$

This follows by subtracting the two recursions in Equation 5 and adding and subtracting $\mathbf{D}_l \mathbf{B}_l \boldsymbol{\delta}_{l+1}^{\mathrm{BP}}$. Each layer therefore propagates the downstream discrepancy and adds a term determined by its own feedback mismatch. At exact initial alignment, $\mathbf{B}_l = \mathbf{W}_l(0)^\top$ for every layer, so all error discrepancies vanish before the first parameter update. As the forward weights change, the mismatch terms can become nonzero even though the feedback matrices remain fixed.

The role of the initial angle can be made explicit for the scratch-training initialization. Write $\alpha = \theta_{\mathrm{init}}$ and $\mathbf{B}_l = \mathbf{W}_l(0)^\top \cos\alpha + \mathbf{R}_l \sin\alpha$, where $\mathbf{W}_l(0)^\top$ and $\mathbf{R}_l$ are independent, zero-mean matrices with equal second moments. This gives $\mathbb{E}\|\mathbf{B}_l\|_F^2 = \mathbb{E}\|\mathbf{W}_l(0)\|_F^2$, and hence

$$\begin{aligned} \mathbb{E}\|\mathbf{W}_l(0)^\top - \mathbf{B}_l\|_F^2 &= \left[(1-\cos\alpha)^2 + \sin^2\alpha\right] \mathbb{E}\|\mathbf{W}_l(0)\|_F^2, \\ \frac{\mathbb{E}\|\mathbf{W}_l(0)^\top - \mathbf{B}_l\|_F^2}{\mathbb{E}\|\mathbf{B}_l\|_F^2} &= 2(1-\cos\alpha) = 4\sin^2(\alpha/2). \end{aligned} \tag{7}$$

Independence and zero means remove the cross term in the squared Frobenius norm. The expected mismatch increases monotonically for $0 \le \alpha \le \pi/2$ and is approximately α^2 near zero when α is expressed in radians. This predicts the ordering of initial feedback mismatch, consistent with the initial gradient cosines in Table S20. It does not by itself predict the final accuracy or robustness of a trained network.

Positive alignment with BP also has a local optimization interpretation. Let $g_{\mathrm{BP}} = \nabla\mathcal{L}(\theta)$ and let u_{IFA} denote the full IFA parameter-update direction before any adaptive optimizer transformation. On a region where the loss has a β-Lipschitz gradient,

$$\mathcal{L}(\theta - \eta u_{\mathrm{IFA}}) \le \mathcal{L}(\theta) - \eta \langle g_{\mathrm{BP}}, u_{\mathrm{IFA}} \rangle + \frac{\beta \eta^2}{2} \|u_{\mathrm{IFA}}\|_2^2. \tag{8}$$

Thus, if $\langle g_{\mathrm{BP}}, u_{\mathrm{IFA}} \rangle > 0$, the right-hand side is strictly below $\mathcal{L}(\theta)$ for $0 < \eta < 2\langle g_{\mathrm{BP}}, u_{\mathrm{IFA}} \rangle / (\beta \|u_{\mathrm{IFA}}\|_2^2)$, provided the step stays in that smooth region. This condition explains why a BP-aligned surrogate can provide a useful descent direction. It is a local statement about the untransformed update and does not establish convergence of the adaptive optimizers used in the experiments.

For a single sample, the weight-gradient matrix at layer l has the form $\boldsymbol{\delta}_{l+1} \mathbf{h}_l^\top$. At the same forward parameters, both rules share $\mathbf{h}_l$, so the cosine between their nonzero weight-gradient matrices equals the cosine between their error signals at layer $l+1$. For a batch, per-sample outer products are summed and this equality need not hold. The output-layer update uses the shared output error directly and has the same functional form under BP, FA, and IFA; feedback weights affect how that error is propagated to earlier layers.

A.6.2 Depth-dependent accumulation of feedback mismatch

We next make the dependence on feedback-path length explicit. For hidden layer l, define $\mathbf{A}_k = \mathbf{D}_k \mathbf{W}_k^\top$ and $\widetilde{\mathbf{A}}_k = \mathbf{D}_k \mathbf{B}_k$, and let

$$\mathbf{P}_l^{\mathrm{BP}} = \mathbf{A}_l \mathbf{A}_{l+1} \cdots \mathbf{A}_{L-1}, \qquad \mathbf{P}_l^{\mathrm{IFA}} = \widetilde{\mathbf{A}}_l \widetilde{\mathbf{A}}_{l+1} \cdots \widetilde{\mathbf{A}}_{L-1}. \tag{9}$$

These operators map the shared output error to the error signal at layer l. Their difference admits the telescoping decomposition

$$\mathbf{P}_l^{\mathrm{IFA}} - \mathbf{P}_l^{\mathrm{BP}} = \sum_{j=l}^{L-1} \left(\prod_{k=l}^{j-1} \widetilde{\mathbf{A}}_k \right) \left(\widetilde{\mathbf{A}}_j - \mathbf{A}_j \right) \left(\prod_{k=j+1}^{L-1} \mathbf{A}_k \right), \tag{10}$$

where products are ordered by increasing index from left to right and empty products are identity maps. The identity follows by replacing one factor at a time, beginning at the leftmost factor, and summing the resulting changes.

Let $\|\cdot\|$ denote the spectral operator norm. If $\|\mathbf{A}_k\|, \|\widetilde{\mathbf{A}}_k\| \le \rho$ and $\|\widetilde{\mathbf{A}}_k - \mathbf{A}_k\| \le \epsilon$ along a path of length $m = L - l$, submultiplicativity and the triangle inequality give

$$\|\mathbf{P}_l^{\mathrm{IFA}} - \mathbf{P}_l^{\mathrm{BP}}\| \le m \rho^{m-1} \epsilon. \tag{11}$$

To relate this operator difference to the actual error-signal alignment, we additionally require that BP does not strongly attenuate the particular output error under consideration. Specifically, for nonzero $\boldsymbol{\delta}_L$, assume $\|\mathbf{P}_l^{\mathrm{BP}} \boldsymbol{\delta}_L\|_2 \ge \underline{\rho}^m \|\boldsymbol{\delta}_L\|_2$ for some $0 < \underline{\rho} \le \rho$. This condition is imposed on the propagated error direction; a lower bound on the operator norm alone would not ensure it. Writing $\kappa = \rho / \underline{\rho} \ge 1$ yields

$$\frac{\|\mathbf{e}_l\|_2}{\|\boldsymbol{\delta}_l^{\mathrm{BP}}\|_2} \le \frac{m \rho^{m-1} \epsilon \|\boldsymbol{\delta}_L\|_2}{\|\mathbf{P}_l^{\mathrm{BP}} \boldsymbol{\delta}_L\|_2} \le m \kappa^{m-1} \frac{\epsilon}{\underline{\rho}} \equiv \eta_l. \tag{12}$$

The relative bound is invariant to a common rescaling of the feedback operators and increases with path length when $\kappa \ge 1$. If $\eta_l < 1$, it also implies

$$\cos\left(\boldsymbol{\delta}_l^{\mathrm{IFA}}, \boldsymbol{\delta}_l^{\mathrm{BP}} \right) \ge \sqrt{1 - \eta_l^2}. \tag{13}$$

To see this, set $r = \|\boldsymbol{\delta}_l^{\mathrm{IFA}}\|_2 / \|\boldsymbol{\delta}_l^{\mathrm{BP}}\|_2 > 0$ and denote the cosine by c. The relative-error bound gives $r^2 + 1 - 2rc \le \eta_l^2$, so $c \ge (r^2 + 1 - \eta_l^2)/(2r) \ge \sqrt{1 - \eta_l^2}$.

This analysis shows how each additional feedback operator can contribute mismatch, with its effect modulated by the scale and conditioning of the remaining path. It provides an explanation for the reduced early-layer alignment at larger depths and the pronounced degradation at weak initial alignment (Table S13). The bounds are informative when mismatch is small and the BP error path satisfies the stated nondegeneracy condition. They are operator-level and single-sample statements, rather than a convergence theorem or a quantitative prediction of final test accuracy.

B Methods

B.1 Initial alignment between forward and backward weights

In baseline feedback alignment (FA), the forward weights $\mathbf{W}_l$ and backward weights $\mathbf{B}_l$ are initialized independently. Initial feedback alignment (IFA) instead introduces a soft alignment at initialization. For networks trained from scratch, we first sampled $\mathbf{W}_l(0)$ and an independent noise matrix $\mathbf{R}_l$ from zero-mean Gaussian distributions with variance a^2/fan_{in}, and initialized

$$\mathbf{B}_l = \mathbf{W}_l(0)^\top \cos(\theta_{\text{init}}) + \mathbf{R}_l \sin(\theta_{\text{init}}).$$

Here, $\mathbf{R}_l$ has the same shape as $\mathbf{W}_l(0)^\top$, fan_{in} denotes the number of input units, and $a = \sqrt{2}$ follows He initialization for ReLU networks. Exact alignment is obtained at $\theta_{\text{init}} = 0^\circ$, whereas 90° gives independent forward and backward weights, corresponding to FA.

For pretrained models, we retained the pretrained forward weights and initialized $\mathbf{B}_l = (\mathbf{W}_l^{\text{pre}})^\top$. In both settings, feedback weights remained fixed while forward parameters were updated. The output-layer weight update uses the loss derivative directly; feedback from that layer propagates errors to preceding layers.

B.2 Multilayer perceptron training

We employed a multi-layer feedforward neural network f_θ for the controlled image-classification experiments. The network received a flattened image vector and processed it through fully connected hidden layers with ReLU activations, followed by a linear classifier. Unless otherwise specified, we used two hidden layers with 512 units each. Forward weights were initialized using the Gaussian scheme described in Section B.1, and biases were initialized to zero.

The primary dataset was CIFAR-10, which contains 50,000 training and 10,000 test images. To assess generalizability, we also evaluated SVHN, CIFAR-100, and STL-10. Images were normalized using training-set channel statistics. The CIFAR-10 MLP experiments used unaugmented images, flattened to 3,072 input features.

We compared BP, FA, and IFA using cross-entropy loss and Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a fixed learning rate of 10^{-4}, and a batch size of 128. Training lasted 30 epochs in the main experiments. We extended training to 100 epochs for trainability and the additional depth, width, and initial-angle analyses, and to 150 epochs for long-horizon learning. Unless otherwise indicated, IFA used exact initial alignment. Other training conditions were matched across methods. Results are reported as the mean and standard deviation across independent runs, with run counts given in the corresponding captions.

B.3 Vision model training and fine-tuning

To assess whether initial alignment generalizes across neural network architectures, we trained convolutional variants of LeNet and AlexNet on CIFAR-10 (Figure S3). The LeNet variant contained two convolutional layers followed by three fully connected layers, whereas the AlexNet variant contained five convolutional layers followed by three fully connected layers. We compared BP, FA, and IFA while keeping the forward architecture, data, and training conditions matched across methods. These networks were trained from scratch using cross-entropy loss and the feedback initialization described in Section B.1.

We also evaluated fine-tuning of an ImageNet-pretrained ResNet-18 on CIFAR-10 (Table S6). We retained the pretrained backbone with its standard 7×7 convolutional stem and replaced the final classifier with a randomly initialized 10-class linear layer. Training images were randomly cropped at 32×32 with padding of 4 pixels and randomly flipped horizontally, then resized to 224×224 and normalized using ImageNet channel statistics. Test images were resized and normalized without augmentation. For BP, FA, and IFA, the full network was fine-tuned using cross-entropy loss and stochastic gradient descent with a batch size of 128, an initial learning rate of 0.01, momentum of 0.9, and weight decay of 5×10^{-4}, with cosine learning-rate decay over 40 epochs. IFA initialized each feedback tensor as a copy of the corresponding forward tensor. FA instead shuffled its output channels to remove the initial correspondence while preserving the tensor's values and norm. These feedback tensors remained fixed throughout fine-tuning.

B.4 Language-model fine-tuning experiments

B.4.1 Models and datasets

We evaluated pretrained Qwen3-1.7B on GSM8K, StrategyQA, BoolQ, ARC-Challenge, RTE, and CommonsenseQA, and Qwen3-8B on ARC-Challenge and RTE. All methods used the same pretrained forward weights and tokenizer for a given model.

ARC-Challenge used its official training, validation, and test splits. For the other tasks, the first 90% of the training split was used for training and the remaining 10% for validation. Final evaluation used the labeled test split for GSM8K and StrategyQA, and the official validation split for BoolQ, RTE, and CommonsenseQA, whose test labels are not public. StrategyQA used the partition distributed as `ChilleD/StrategyQA`. GSM8K and StrategyQA targets contained step-by-step reasoning followed by a final numerical or Yes/No answer. GSM8K reasoning was taken from the annotated solution, whereas StrategyQA reasoning was constructed from the available supporting facts or decomposition. BoolQ, ARC-Challenge, RTE, and CommonsenseQA used the corresponding Yes/No, answer-choice, or entailment labels. The question and task instructions formed the prompt. Training loss was computed over target tokens, including the end-of-sequence token when present, with prompt tokens masked. Examples were truncated to the maximum sequence length; short-label tasks retained the target while truncating long prompts.

B.4.2 Full fine-tuning

We fine-tuned the pretrained models using BP, FA, and IFA. For FA and IFA, input-gradient calculations in transformer attention projections, feedforward projections, and the language-model head used separate fixed feedback tensors; other operations retained their usual derivatives. IFA initialized feedback from the pretrained forward weights, whereas FA used output-channel-shuffled copies. Both retained the pretrained forward state, and feedback tensors were excluded from optimization. Training used AdamW with a learning rate of 2×10^{-5}, weight decay of 0.01, cosine learning-rate decay, and a 3% linear warmup. The microbatch size was 1, with gradient accumulation over 16 microbatches. The task comparisons in Table 1 used three epochs and a maximum sequence length of 512; GSM8K used four epochs and a maximum supervised sequence length of 1,024. Projection weights and their feedback tensors were stored in BF16. Qwen3-8B used activation checkpointing.

B.4.3 Low-rank adaptation

We fine-tuned Qwen3-1.7B on ARC-Challenge and RTE using rank-8 LoRA adapters in the transformer projection layers, with a scaling parameter of 16 and dropout of 0.05. The pretrained backbone, embeddings, and language-model head remained frozen. Only adapter factors were trained, using AdamW with a learning rate of 2×10^{-4} for three epochs; other training settings followed full fine-tuning. For each projection, the adapter contributed a rank-8 weight update scaled by $16/8$. The down factor was initialized to zero and the up factor was nonzero, preserving the initial pretrained output while permitting the down factor to learn from the first step. FA and IFA used fixed feedback for the input-gradient paths of the frozen backbone, language-model head, and both adapter factors. Feedback was initialized as shuffled or exact copies of the corresponding forward tensors, respectively. Adapter factors and their feedback tensors were stored in FP32.

B.4.4 Evaluation metrics

We measured accuracy using greedy generation and compared the extracted answer with the reference. GSM8K and StrategyQA allowed up to 512 and 256 generated tokens, respectively; short-label tasks allowed 8. We extracted the final numerical answer for GSM8K, the Yes/No answer for StrategyQA, and the first standalone valid task label for the remaining tasks. Explicit final-answer expressions took priority for the reasoning tasks, with fallback extraction when these were absent. Missing or incorrect answers were counted as errors, and accuracy was the percentage of evaluated examples with a correct extracted answer.

We also computed held-out perplexity under teacher forcing over the supervised target tokens:

$$\mathrm{PPL} = \exp\left[-\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(y_i \mid x, y_{<i})\right],$$

where x denotes the prompt, y_i a target token, and N the total number of non-masked target tokens. Loss was summed over target tokens before normalization, excluding prompt and padding tokens. Thus, perplexity included reasoning tokens for GSM8K and StrategyQA and label tokens for the short-label tasks. Since target formats differ, comparisons were made only within each task. Perplexity was capped at 10^{10} per run before aggregation; *diverged* denotes all three runs reaching this cap. Full fine-tuning task comparisons and LoRA results are reported as the mean $\pm$ standard deviation across three independent runs. Pretrained task accuracy was evaluated once.

B.4.5 Feedback transport and GPU memory measurement

For Qwen3-8B fine-tuning on ARC-Challenge, we calculated forward-to-feedback transport and feedback-array writes assuming physically separate forward and feedback arrays. Forward-to-feedback transport counted bytes copied from the forward weights into the feedback pathway; feedback-array writes counted all bytes programmed into that array. Byte counts were obtained from tensor sizes and storage precision. For T optimizer steps, BP programmed the initial transpose and refreshed it before each subsequent backward pass, giving T programming passes. IFA transported and programmed the initial feedback array once. FA required one programming pass and no forward-weight transport, assuming local random-feedback generation. These calculations describe separate-pathway transport and programming, rather than measured GPU traffic or energy consumption.

Peak GPU memory was measured on an NVIDIA H200 during the same task and averaged across three independent runs. Model, batch, and training settings were matched across methods. FA and IFA stored an additional fixed feedback array, so GPU memory capacity and feedback transport were evaluated as distinct resource requirements.

B.5 Trainability and convergence analysis

To assess trainability, we trained MLPs for 100 epochs and recorded the maximum training accuracy while varying initialization scale, network depth, or training-set size. For the initialization-scale analysis, forward and feedback weights were sampled from zero-mean Gaussian distributions with variances a^2/fan_{in} and b^2/fan_{in}, respectively. We varied the two scales independently to identify the trainable region. Depth analyses kept hidden-layer width fixed at 512, whereas limited-data analyses varied the number of training examples while keeping the architecture and optimizer fixed.

The additional depth–angle analysis used 2, 6, or 10 hidden layers and initial angles of 0°, 30°, 60°, or 90° (Table S13). The width analysis used six hidden layers with widths ranging from 32 to 2,048 (Table S14). In these analyses, depth denotes the number of hidden layers, excluding the output layer, and test accuracy was evaluated at epoch 100. BP–FA gap recovery was calculated within each matched run as $(A_{\text{IFA}} - A_{\text{FA}})/(A_{\text{BP}} - A_{\text{FA}})$ using IFA initialized at 0°, where A denotes final test accuracy.

For long-horizon learning, we compared test accuracy at epochs 30 and 150, its change over epochs 100–150, and the training-loss slope obtained by a linear fit over the final 20 epochs. Learning speed was measured by the first epoch reaching 95% of the corresponding BP run's epoch-150 accuracy. We also measured the first epoch reaching 80%, 90%, or 99% of each method's own epoch-150 accuracy. Thresholds were calculated separately for each run, and unreached thresholds were reported as not reached.

B.6 Measurement of the alignment between forward and backward weights

To quantify the correspondence between forward and feedback pathways, we measured the angle between corresponding weight vectors. For neuron i at layer l, the vectors were the i-th rows of $\mathbf{W}_l^\top$ and $\mathbf{B}_l$. Their angular similarity was computed using cosine similarity [26]:

$$\theta_i = \cos^{-1}\left(\frac{\mathbf{W}_l^{\top(i)} \cdot \mathbf{B}_l^{(i)}}{\|\mathbf{W}_l^{\top(i)}\| \, \|\mathbf{B}_l^{(i)}\|}\right).$$

We reported the average angle across neurons as a summary measure. Smaller angles indicate stronger alignment, with 0° corresponding to identical weight directions. This measure describes weight correspondence and is distinct from the gradient-alignment measures below.

B.7 Parameter drift and representation similarity

To quantify changes in the forward parameters during learning, we measured relative parameter drift from initialization:

$$D(t) = \frac{\|\theta_t - \theta_0\|_2}{\|\theta_0\|_2}.$$

Here, θ_t concatenates all forward weights and biases, including the output layer. Fixed feedback tensors and optimizer state were excluded. The denominator was the norm of the initial parameter vector, so the measure describes the overall change relative to initialization.

We assessed representation changes using centered linear kernel alignment (CKA) between initial and current hidden activations. On a fixed subset of 1,024 CIFAR-10 training images, we recorded the post-ReLU activation matrix at each hidden layer. After centering each feature across images, we computed

$$\mathrm{CKA}(\mathbf{X}_0, \mathbf{X}_t) = \frac{\|\mathbf{X}_0^\top \mathbf{X}_t\|_F^2}{\|\mathbf{X}_0^\top \mathbf{X}_0\|_F \, \|\mathbf{X}_t^\top \mathbf{X}_t\|_F}.$$

Rows of $\mathbf{X}_0$ and $\mathbf{X}_t$ correspond to the same images at initialization and the current epoch. We reported the mean CKA across hidden layers. Thus, parameter drift measured changes in forward parameters, whereas CKA measured changes in the representations produced by those parameters.

B.8 Gradient alignment analysis

To assess the learning signal provided by fixed feedback, we compared the learning-rule gradient with the exact BP gradient at the same forward parameters. Both gradients were computed from the mean cross-entropy loss on the same fixed subset of 1,024 CIFAR-10 training images. Loss gradients were accumulated over all probe images before computing cosine similarity. Learning-rule gradients were measured before the Adam transformation, so the comparison concerned the gradient approximation rather than the final optimizer step.

We concatenated hidden-weight gradients into a single vector, excluding biases and the output-layer weights. Let $g_{\mathrm{BP}}(\theta_t)$ denote the exact BP gradient at the current forward state and $g_{\mathrm{rule}}(\theta_t)$ the corresponding BP, FA, or IFA gradient. We computed

$$C_{\mathrm{init}}(t) = \cos\left(g_{\mathrm{BP}}(\theta_0), g_{\mathrm{BP}}(\theta_t)\right), \qquad C_{\mathrm{rule}}(t) = \cos\left(g_{\mathrm{rule}}(\theta_t), g_{\mathrm{BP}}(\theta_t)\right).$$

The first quantity measures rotation of the exact BP gradient from its initial direction; the second measures how closely the learning rule follows the current BP gradient. The exact BP gradient was recomputed at each model's own forward parameters, rather than taken from a separately trained BP trajectory.

For the depth–angle analysis, we additionally compared gradients separately in the first, middle, and last hidden layers. Initial and final cosines in the angle analysis were evaluated at epochs 0 and 100, respectively.

B.9 Visualization of trajectory in loss landscape

To visualize the optimization trajectory within the loss landscape [30], we recorded model parameters during training and flattened them into one-dimensional vectors. We applied Principal Component Analysis (PCA) to the parameter differences $\theta_t - \theta_{\mathrm{final}}$. The two leading components defined a two-dimensional subspace within the parameter space. We evaluated training loss on a grid of parameter perturbations within this subspace and projected the training trajectory onto the same plane. This provided a common coordinate system for displaying the loss surface and the path followed during optimization.

B.10 Spectral analysis on Hessian of loss

To characterize loss-landscape geometry, we analyzed the Hessian $H_\theta = \nabla_\theta^2 \mathcal{L}$ without explicitly forming the full matrix [31]. For a fixed probe vector v, Hessian–vector products were computed by differentiating the gradient inner product,

$$H_\theta v = \nabla_\theta \left(g_\theta^\top v\right), \qquad g_\theta = \nabla_\theta \mathcal{L}.$$

We estimated the largest eigenvalue by power iteration [48]. The trace was estimated using the identity $\mathrm{Tr}(H_\theta) = \mathbb{E}[v^\top H_\theta v]$, averaging over independent random vectors $v \sim \mathcal{N}(0, I)$ [49]. We also estimated the spectral density using stochastic Lanczos quadrature [50, 51]. These measurements characterized curvature along training trajectories and around the final solutions.

B.11 Perturbed loss landscape analysis

To evaluate the stability of the final solution, we computed the Hessian eigenvectors corresponding to its two largest eigenvalues. These vectors, v_1 and v_2, defined a two-dimensional perturbation plane:

$$\theta_{\mathrm{perturbed}} = \theta_{\mathrm{final}} + \alpha v_1 + \beta v_2.$$

We evaluated test loss across a grid of α and β, with $(0, 0)$ corresponding to the unperturbed model. The resulting surface described sensitivity to perturbations along the dominant curvature directions. Flatter surfaces indicated smaller loss changes around the final parameters.

B.12 Corruption robustness

To assess robustness to input degradation, we evaluated models trained on clean CIFAR-10 using CIFAR-10-C [37]. This dataset contains 15 corruption types, including noise, blur, weather, and digital distortions, each at five severity levels. Robustness was measured by mean classification accuracy across corruption types at a given severity. The main analysis used severity level 1 to represent mild corruption; results for all five severity levels are provided in the Supplementary Results. Higher mean corruption accuracy indicates better retention of performance under these distributional shifts.

B.13 Adversarial robustness

To assess the robustness of the neural network against adversarial attacks, we employed the Fast Gradient Sign Method (FGSM) [38]. FGSM generates adversarial examples by adding perturbations in the direction that maximally increases the loss. Given an input x, label y, and loss function $\mathcal{L}(x, y)$, the adversarial example is constructed as:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y)),$$

where ϵ controls the magnitude of the perturbation. We evaluated adversarial robustness by varying ϵ from 0 to 0.05 and measuring the corresponding test accuracy on the perturbed inputs. A steep decline in accuracy with increasing ϵ indicates vulnerability to adversarial perturbations, whereas more stable accuracy suggests stronger robustness.

C Data availability

The vision datasets used in this study are available at the following links.

- CIFAR-10 and CIFAR-100 [44]: `https://www.cs.toronto.edu/~kriz/cifar.html`
- SVHN [43]: `http://ufldl.stanford.edu/housenumbers/`
- STL-10 [45]: `https://cs.stanford.edu/~acoates/stl10/`
- CIFAR-10-C [37]: `https://zenodo.org/records/2535967`

The language-model datasets used in this study are available at the following links.

- GSM8K: `https://huggingface.co/datasets/openai/gsm8k`
- StrategyQA: `https://huggingface.co/datasets/ChilleD/StrategyQA`
- BoolQ: `https://huggingface.co/datasets/google/boolq`
- ARC-Challenge: `https://huggingface.co/datasets/allenai/ai2_arc`
- RTE: `https://huggingface.co/datasets/nyu-mll/glue`
- CommonsenseQA: `https://huggingface.co/datasets/tau/commonsense_qa`

The pretrained model checkpoints used in this study are available at the following links.

- ResNet-18: `https://docs.pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html`
- Qwen3-1.7B: `https://huggingface.co/Qwen/Qwen3-1.7B`
- Qwen3-8B: `https://huggingface.co/Qwen/Qwen3-8B`

D Computing resources

All experiments were implemented in PyTorch, using GPU parallelization for efficient computation. An NVIDIA GeForce RTX 4090 GPU was used for the main experiments, and NVIDIA H100 and H200 GPUs were used for the LLM experiments.

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 and introduction clearly state the paper's main claims, including its contributions, assumptions, and limitations.

   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: We include a separate "Limitations" section in the paper, where we discuss the scope of our claims and assumptions.

   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: [Yes].

Justification: We state the assumptions and provides derivations for each theoretical result.

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: We describe detailed methods and instructions for reproducing the experimental results in the corresponding sections of the Appendix.

   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: We provide open access to the data and code via a GitHub repository.

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: We provide experimental details in the Appendix.

   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: We report error bars for all results shown in the figures, as well as detailed numerical statistical values.

   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: We provide information on the compute resources used for the experiments in the Appendix.

   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 reviewed the NeurIPS Code of Ethics, and our research conforms to it.

   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: We discuss both the potential positive and negative societal impacts of the work.

    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: [N/A].

    Justification: Since our paper is basic academic research, it does not pose such risks.

    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: We cite the original papers that provide the publicly available assets and describe how to access them in the Appendix.

    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: [Yes].

    Justification: We provide detailed documentation, such as README files, alongside the 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: Our paper does not involve crowdsourcing or research with human subjects.

    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: Our paper does not involve crowdsourcing or research with human subjects.

    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: The core method development in this research does not involve LLMs as important, original, or non-standard components.

    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.