Overcoming Rank Collapse in Feedback Alignment

Gauthier Boeshertz
Department of Bioengineering
Imperial College London

Razvan Pascanu*
Mila

Claudia Clopath*
Department of Bioengineering
Imperial College London

Abstract

Backpropagation (BP) is widely viewed as biologically implausible, in part because it requires feedback weights to be the transpose of forward weights for error propagation. Interestingly, when training a network with fixed random feedback weights to circumvent this issue, learning aligns the forward weights with the feedback weights, leading the backpropagated error signal to become an approximation of the standard gradient used by BP. This process, called feedback alignment (FA), occurs in MLPs and very shallow CNNs but does not scale well to deeper architectures. In this work, we first investigated differences between BP and different FA methods, trained on CIFAR10, specifically focusing on the effective rank of the error signal. We found that FA methods produce pseudo-gradient signals that have a considerably lower rank and hence are constrained to a lower-dimensional subspace compared to BP, limiting exploration of the parameter space. Motivated by this observation, we evaluated two mechanisms for increasing the effective dimensionality of FA: Muon, an optimiser that orthogonalises weight updates, and activity normalisation, which promotes activation orthogonality. Across larger architectures and benchmarks, we find that these methods consistently improve over FA baselines, for example, on ImageNet with AlexNet, adding Batch Normalisation (BN) increases FA accuracy from 0.1% to 11.3%, using Muon on top of BN increases the accuracy to 20.2%. Our results identify low-dimensional gradient dynamics as a key limitation of FA, and suggest that inducing higher-dimensional update geometry is a promising route toward scaling alternatives to BP.

1 Introduction

Most neural networks are trained using error backpropagation (BP) [Rumelhart et al., 1986]. This method performs credit assignment by computing the gradient of a loss function for every parameter inside the network. A biological view of this process says that the neurons project an error signal backwards, using synapses that share the same weights as the synapses projecting upwards. Sharing the weight between the forward and backward connections has been one of the main issues with regard to the biological plausibility of BP [Crick, 1989, Lillicrap et al., 2016, 2020]. However, [Lillicrap et al., 2016] has shown that it is possible to train networks with random feedback projections and achieve results similar to BP on simple tasks. They observed that throughout training, the forward weights would align to the random feedback, allowing a meaningful error to be backpropagated, as quantified by the alignment of the gradients with the random feedback and gradients obtained with weight sharing. While the error signal propagated in feedback alignment (FA) is not a gradient of the loss, we will refer to it as a gradient to be consistent with prior works.

This observation created a vast literature on making more biological training methods using random feedback projections. Direct feedback alignment (DFA) [Nøkland, 2016] showed that it is possible to directly feed the error signal from the output to any hidden layer using feedback projections of

*Co-senior Authors

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

the appropriate size. Building on this idea, [Frenkel et al., 2021] showed that even the error can be replaced by the target itself.

However all of these methods fail to scale to more difficult tasks [Liao et al., 2016, Xiao et al., 2018, Moskovitz et al., 2018, Launay et al., 2019, Bartunov et al., 2018]. For example, [Launay et al., 2019] reported that convolutional neural networks (CNNs) trained with DFA do not perform better than chance on CIFAR100. In general, FA methods have been found to perform much worse than BP even in shallow CNNs. The reason behind this difference in performance was attributed to the lack of flexibility of convolutional layers due to weight sharing [Refinetti et al., 2021, Launay et al., 2019]. To mitigate this lack of flexibility, several methods use the signs of the forward weights to set the signs of the feedback [Liao et al., 2016, Xiao et al., 2018, Moskovitz et al., 2018], and achieved performance closer to that of BP. Instead of imposing a shared structure, some methods start with random feedback and train the feedback along with the forward weights [Akrout et al., 2019, Kunin et al., 2020, Hanut and Kadmon, 2025], and reach performance even closer to BP.

In this work, we focus on three different formulations of FA. The original FA [Lillicrap et al., 2016], DFA [Nøkland, 2016], and LDFA [Hanut and Kadmon, 2025], which trains low rank feedback weights. Our objective is not to completely close the gap between BP and FA. Rather, it is to understand FA in deep networks and explain what stops alignment from happening. To pursue this objective, we investigated the differences in the geometry of the gradients in networks trained using BP and FA. We find that in deep FA networks, when the weights start aligning, there is a marked decrease in the effective rank [Roy and Vetterli, 2007] of gradients. This means that the gradients will have a few directions that dominate the others, which may not allow the alignment process to continue, since the parameter updates have to explore much smaller subspaces. We therefore try two different methods aimed at increasing the dimensionality of the gradients. When writing dimensionality, we mean the effective dimensionality or rank.

First, we used a recently proposed optimiser, Muon [Bernstein and Newhouse, 2024, Bernstein, 2025, Jordan et al., 2024]. Muon differs from standard SGD with momentum by adding an orthogonalisation step to the momentum update. This means setting every singular value to 1. The second way is to normalise the activity of the layers using Batch Normalisation (BN). BN was shown to orthogonalise activity in finite width random networks [Daneshmand et al., 2020, 2021]. Liao et al. [2016] had already showed that BN had a positive effect on the performance of FA models, but had not explained why. We hypothesise that this is due to the decorrelation of the activations that occurs when using BN. Offergeld and Ahmad [2026] showed that a method aimed at decorrelating activity improved learning in BP and FA networks. Overall, we analyse multiple feedback methods and investigate the effects of the different interventions by studying how low-dimensional gradient dynamics can limit alignment in deep networks. Through our attempt of understanding FA, we also provide new perspectives on the effects of widely used optimisation techniques that differ from those usually obtained in BP.

We tested these methods on the CIFAR10 [Krizhevsky, 2009] dataset and found that they improved the performance of FA models considerably. Using them, the effective dimensionality of the gradients did not exhibit the marked decrease observed before, which supports the claim that FA requires gradients to have large rank for alignment to happen. We then measured the impact of the two methods using ResNet-18 [He et al., 2016] and AlexNet [Krizhevsky et al., 2012] on STL-10 [Coates et al., 2011], CIFAR-100 [Krizhevsky, 2009], and ImageNet [Russakovsky et al., 2015]. Both interventions improved FA and DFA, with their combination giving the highest mean accuracy in every reported benchmark.

2 Optimisation with Feedback Alignment

2.1 Backpropagation and Feedback Alignment

BP [Rumelhart et al., 1986] computes gradients by traversing the computation graph from the output layer back to the input, propagating error signals layer by layer through the transpose of the forward weights. For a standard MLP network with L layers:

$$a_\ell = W_\ell h_{\ell-1} + b_\ell, \quad h_\ell = \phi(a_\ell) \tag{1}$$

where a_ℓ is the pre-activation vector, h_ℓ is the post-activation vector, W_ℓ is the weight matrix, b_ℓ is the bias vector, and $\phi(\cdot)$ is the activation function, for layer $\ell = 1, \ldots, L$. Using the loss computed at

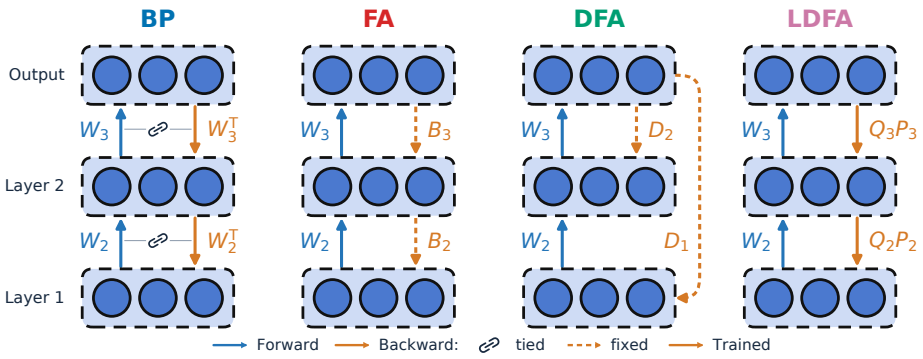


Figure 1: **Feedback pathways for the four learning rules.** BP uses transposed forward weights; FA uses fixed random connections between adjacent layers; DFA uses fixed projections of the output error directly to hidden layers; and LDFA learns the factors Q_ℓ, P_ℓ of each layerwise feedback map. Blue arrows denote forward connections and orange arrows denote feedback.

the output of the network, BP computes the error for each layer recursively:

$$\delta_L = \frac{\partial E}{\partial a_L}, \quad \delta_\ell^{BP} = \left(W_{\ell+1}^\top \delta_{\ell+1}^{BP}\right) \odot \phi'(a_\ell) \quad \text{for } \ell = L-1, \ldots, 1 \tag{2}$$

where $\odot$ denotes the Hadamard product, ϕ' is the derivative of the activation function. The use of the transposed weight in Eq. 2 is the core of the Weight Transport problem. Training a network with FA replaces the transpose with a weight B_ℓ^{FA}, usually sampled from the same distribution as the forward weights, and keeps it fixed during training. DFA projects the output error directly to the hidden layers using feedback weights B_ℓ^{DFA}:

$$\delta_\ell^{FA} = \left(B_{\ell+1}^{FA} \delta_{\ell+1}^{FA}\right) \odot \phi'(a_\ell), \quad \delta_\ell^{DFA} = \left(B_\ell^{DFA} \delta_L\right) \odot \phi'(a_\ell) \quad \text{for } \ell = L-1, \ldots, 1 \tag{3}$$

To investigate the impact of low-dimensional errors, we also considered low-dimensional feedback alignment (LDFA) [Hanut and Kadmon, 2025], which uses learned, factorised feedback maps $B_\ell^{LDFA} = Q_\ell P_\ell$. Writing n_ℓ for the width of layer ℓ, the factors have dimensions $P_\ell \in \mathbb{R}^{r_\ell \times n_\ell}$ and $Q_\ell \in \mathbb{R}^{n_{\ell-1} \times r_\ell}$, so B_ℓ^{LDFA} has rank at most r_ℓ. The projection P_ℓ compresses the incoming error to r_ℓ components, and Q_ℓ maps these components back to the preceding layer:

$$\delta_\ell^{LDFA} = \left(Q_{\ell+1} P_{\ell+1} \delta_{\ell+1}^{LDFA}\right) \odot \phi'(a_\ell) \quad \text{for } \ell = L-1, \ldots, 1. \tag{4}$$

All four methods use the same output error δ_L. The errors are multiplied by the activity of the layer the weights project from to make the gradient, or pseudo-gradient for the FA methods:

$$\frac{\partial E}{\partial W_\ell} = \delta_\ell^{(BP,FA,DFA,LDFA)} (h_{\ell-1})^\top, \quad \frac{\partial E}{\partial b_\ell} = \delta_\ell^{(BP,FA,DFA,LDFA)} \tag{5}$$

where we set h_0 to be the input to the network.

Although we wrote the equations for an MLP, the same ideas apply to CNNs. In convolutional layers, backpropagation still propagates errors backward and forms gradients from error signals and layer activations, but the linear maps are convolutional operators with shared weights across spatial locations rather than dense matrices. Accordingly, the backward pass uses the adjoint convolution operator, and FA replaces this operator with a fixed random feedback map of compatible shape. For LDFA, a learned 1×1 projection compresses the error channels before applying the adjoint convolution associated with the learned spatial kernel Q_ℓ.

Additional implementation details for all four methods are provided in Appendix A.

2.2 Optimisation

Once gradients have been computed, their use is determined by the optimiser, which is separate from the choice of learning rule used to obtain them. FA can therefore use any optimiser that works for

BP. We will especially be interested in a recently proposed optimiser, called Muon [Bernstein and Newhouse, 2024, Bernstein, 2025, Jordan et al., 2024]. This optimiser first received attention because it leads to faster convergence under mild additional computational costs [Jordan et al., 2024], and recently because it leads to lower loss for the same compute budget in large language models [Liu et al., 2025]. It works similarly to SGD with momentum, where β is the momentum parameter, but adds an orthogonalisation step:

$$M_t = \beta M_{t-1} + (1-\beta)\frac{\partial E}{\partial W^\ell}, \quad W^\ell_{t+1} = W^\ell_t - \eta \ \mathrm{Orth}(M^\ell_t) \tag{6}$$

The orthogonalisation function Orth computes the closest orthogonal matrix to its input, which can be written in terms of the SVD of the input, $X = U\Sigma V^\top$, then the orthogonalised matrix is given by $\mathrm{Orth}(X) = UV^\top$. This operation is usually done with the Newton-Schulz approximation [Jordan et al., 2024], which is quicker than computing the singular vectors. An intuitive explanation of the orthogonalisation is given in [Jordan et al., 2024], which we rely on for our results. It states that updates in standard optimisers have low rank, and orthogonalising the update will make the lesser explored directions more relevant. For vector parameters which cannot have their updates orthogonalised and need to use AdamW [Loshchilov and Hutter, 2017]. We use a version of Muon that scales the updates to have a similar RMS norm to AdamW [Liu et al., 2025], which allows the reuse of the learning rate across parameters trained with AdamW and Muon.

3 Pathologies of FA

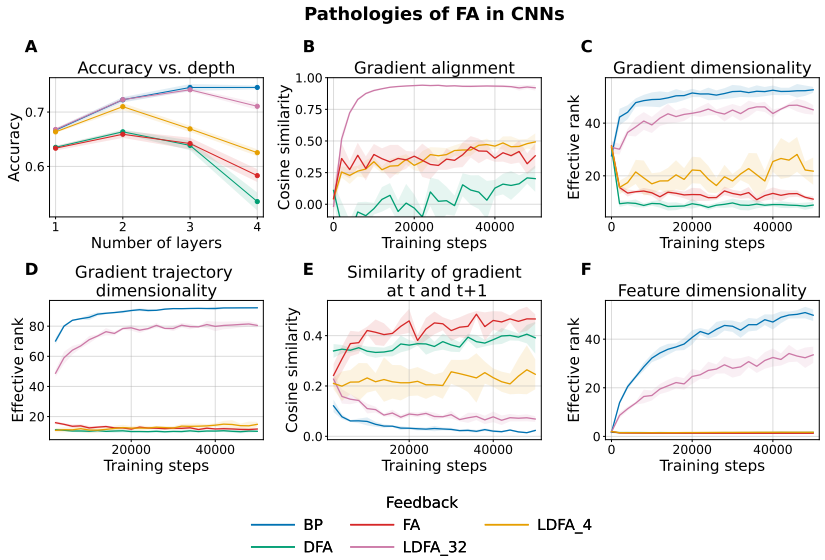


Figure 2: **Deep FA networks suffer from low dimensional gradients. A** Increasing the number of layers results in lower performance in FA networks. **B-F** Metrics for the last layer of the 4 convolutional layer model. **B** Gradient alignment measured by the cosine similarity of the gradient obtained using the feedback weight and the transpose of the forward weights for the feedback projections. **C** Effective rank of the gradient in the last convolutional layer. The FA networks have much lower dimensionality. **D** Effective rank of the Gram matrix of the gradients. Again, FA networks have much lower dimensionality. **E** Cosine similarity of the gradients at step t and t+1. FA networks have similar gradients, which explains the low-dimensional gradient trajectory. **F** The feature dimensionality is much lower than for the BP models.

We first explored the effects of FA in deep networks to understand why it performs substantially worse than BP in deep CNNs. We trained CNNs with depth varying from 1 to 4 before the classifier. Each convolutional layer had 64 channels, a 3x3 kernel, and was followed by 2x2 max-pooling; on the CIFAR10 [Krizhevsky, 2009] dataset. Using more channels leads to a decrease of the gap between FA and BP [Boopathy and Fiete, 2022]; however, in sufficiently deep networks, this gap remains large, leading us to focus on the settings in which FA fails. We use 64 as it strikes a balance between

too narrow and too wide networks, and because it usually is the number of channels in the first layer of deep CNNs, for example, Resnets [He et al., 2016] and VGGs Simonyan and Zisserman [2014]. We used ReLU activations, as it is the activation used in these deep networks. For the optimiser, we used AdamW [Loshchilov and Hutter, 2017], but note that SGD with momentum gives similar results. The weights of the convolutional and linear layers were initialised using the Kaiming uniform initialisation as in [Frenkel et al., 2021]. We also found that it was the one that worked best for FA networks, which heavily depended on having the right initialisation. We first measured the effect of depth on accuracy, using identical architectures for every feedback method at each depth, allowing a matched comparison as convolutional blocks are added. We then turned to diagnostics metrics to investigate the source of the gap (Figure 2).

Consistent with earlier studies [Cheon et al., 2024, Launay et al., 2019], we found that the accuracy of the FA models decreases with depth (Figure 2 A). To understand this failure of FA, we turned our attention to networks with 4 convolutional layers. The gradient alignment, measured by the cosine similarity of the flattened pseudo gradients with the true gradients, as computed by BP, increases at first, then stops (Figure 2 B). Since the gradient alignment is small, the FA gradient will not follow the true gradient of the loss, which explains why the accuracy changes slowly. To explain why FA gradient alignment does not increase much after an initial bump, we analysed the geometry of the updates of the different FA models.

Guided by the intuition that FA makes optimisation happen in lower-dimensional subspaces, we measured three metrics that may explain the failure of FA and DFA to align to BP. They are all based on the effective rank of different matrices. The effective rank [Roy and Vetterli, 2007] is a continuous approximation to the rank of the matrix. It is computed by taking the exponentiated entropy of the normalised singular values of the matrix: $\text{erank}(M) = \exp(-\sum_{i=1}^{r} p_i \log p_i)$, with $p_i = \frac{\sigma_i}{\sum_{j=1}^{r} \sigma_j}$ where σ_i are the singular values. We measured this metric on three different matrices:

- On the gradients of the convolutional layers, to measure the dimensionality of the error propagated by different methods (Figure 2 C). Convolutional layers weights have gradients with shape $C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w$, we retain output channels as rows and flatten input channels and kernel coordinates into columns, obtaining $D_t \in \mathbb{R}^{C_{\text{out}} \times (C_{\text{in}} k_h k_w)}$, as done in the Muon optimizer [Jordan et al., 2024]. We report $r_{\text{grad}}(t) = \text{erank}(D_t)$.
- On the trajectory of the gradients (Figure 2 D). We flattened the gradients from the 100 optimisation steps before computing the metric, and computed the effective rank of the Gram matrix of the stacked gradients. Stacking $T = 100$ such vectors as rows gives $H \in \mathbb{R}^{T \times P}$, where P is the number of weight parameters in that layer. We normalise each row of H to unit ℓ_2 norm, $\tilde{H}_t = H_t / \|H_t\|_2$, so that the metric captures the directions of the gradients and not their magnitudes. The Gram matrix $\tilde{H}\tilde{H}^\top$ is then the matrix of cosine similarities between gradients at different steps. We report $r_{\text{traj}} = \text{erank}(\tilde{H}\tilde{H}^\top)$, which is bounded above by $\min(T, P)$.
- On the features of the last convolutional layer (Figure 2 F). We computed the Gram matrix using the activations that feed into the classifier, and then computed the effective rank of this matrix, as done in [Huh et al., 2021].

We found that the dimensionality of the gradients of FA networks is always lower than that of similar BP networks (Figure 2 C). Moreover, the trajectory of the gradients is also low-dimensional (Figure 2 D). This indicates that the gradients are more aligned with one another. In other words, the gradients span a lower-dimensional subspace, so the parameter updates are confined to fewer directions. The optimisation explores only a limited region of parameter space, which may be insufficient to produce meaningful alignment. In agreement with our hypothesis, the dimensionality of the trajectory of FA networks decreases early, and stays much lower than that of the BP networks (Figure 2 D). Indeed, while the BP trajectory dimensionality almost reaches its maximum rank, 100, the FA one is stuck under 20, and ends at 12. To better explain this, we looked at the entries next to the diagonal of the trajectory Gram matrix, which are the cosine similarity of gradients adjacent in the trajectory. We computed the mean of theses values across the whole matrix and found high cosine similarity for the FA network where it is almost zero for the BP networks (Figure 2 E). The trajectory of the gradients will be higher for BP as it uses updated weights to propagate the error, which may increase the dimensionality when keeping other parts of the gradient equal. FA by construction can not do

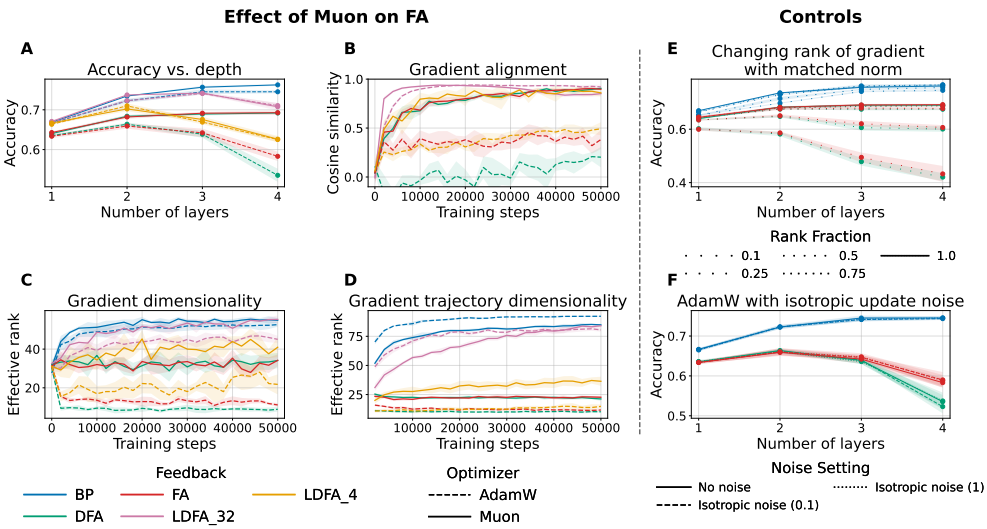


Figure 3: **Orthogonalised updates improve FA learning, and restricting their rank reduces the benefit. A** Test accuracy versus convolutional depth for AdamW (dashed) and Muon (solid). Muon substantially reduces the loss of accuracy with depth for FA and DFA. **B-D** Training dynamics in the last convolutional layer of the four-layer networks.(**B**) Muon increases the gradient alignment. (**C**) The effective rank of the gradient matrix. (**D**) The effective rank of the Gram matrix of the gradient trajectory. The effect on these metrics varies across feedback rules. **E,F**, controls of possible Muon confounding factors. **E** Norm-matched Muon retains the leading 10%, 25%, 50%, 75%, or 100% of singular directions and equalises their singular values, with the same target update RMS before learning-rate scaling. Retaining more directions improves FA and DFA accuracy, especially at greater depth. **F** AdamW with no added noise or isotropic Gaussian noise with norm equal to 0.1 or 1 times the base update norm. Adding noise does not reproduce Muon's accuracy gains in FA and DFA.

this, however, it is not the only reason for the lower trajectory dimensionality, as we show in the next section.

One of the effects of low-dimensional gradients could be that the representations will be low-dimensional too. As expected, the features of the FA networks are much lower-dimensional than those of BP networks, which may be due to the small subspace spanned by the gradients. While low-dimensional features have been found to be beneficial for generalisation [Valle-Perez et al., 2018, Huh et al., 2021, Andriushchenko et al., 2023], we argue here that the small feature dimensionality may result from limited exploration of the parameter space, which limits the alignment of the weights, and therefore the ability of the network to learn useful task related features.

Together, these results suggest that the poor performance of FA in deeper CNNs is due to imperfect alignment with BP, which may be limited because of the geometry of the updates. In particular, FA appears to restrict learning to a low-dimensional subspace, producing highly correlated gradients and low-dimensional features that may be insufficient for discriminating between classes.

4 Higher-dimensional updates enable better FA

The previous section suggests a direct intervention: if the failure mode of FA is caused by updates that are too low-dimensional, then methods that increase the dimensionality of the updates or of the intermediate representations should improve FA training. Therefore, we tried different ways of increasing the dimensionality of the updates, using the same setup as in the previous sections.

Using orthogonalised updates First, we used the Muon optimiser [Jordan et al., 2024]. The low effective rank of the FA gradient suggests that a few singular directions dominate the learning signal, while other directions contribute little to the update. Muon flattens the singular-value spectrum of the momentum update, giving greater relative weight to these weaker directions. Since the feedback rule

determines how gradients are computed and the optimiser determines how they are applied, we can simply swap AdamW with Muon for each feedback rule.

Compared with AdamW, Muon substantially reduces the degradation in accuracy with depth for both FA and DFA (Figure 3 A). In the four-convolutional-layer FA network, test accuracy reaches 69.8% with Muon, compared with 58.3% with AdamW. The benefits are much smaller for BP and LDFA. BP is already well optimised with AdamW, and in LDFA, Muon has less directions to increase dimensionality in due to the nature of the feedback rule. For the four-layer networks, we measured gradient alignment with BP (Figure 3 B), gradient effective rank (Figure 3 C), and gradient-trajectory effective rank (Figure 3 D). For FA and DFA, Muon improves gradient alignment and increases both gradient and trajectory effective rank relative to AdamW.

To test the contribution of update rank while controlling its scale, we used a norm-matched variant of Muon (Figure 3 E). We replace the orthogonalisation step with a truncated SVD, retaining only the leading singular directions and normalising the resulting matrix to a fixed target RMS: $0.2\sqrt{\frac{mn}{r}}U_{[:,1:r]}V_{[:,1:r]}^{\top}$. We swept over a range of fraction of the total number of singular values to retain. Retaining only a small fraction of directions strongly reduces FA and DFA accuracy, particularly in deeper networks, whereas BP is less affected. Increasing the retained fraction recovers much of the benefit of full-rank Muon. Thus, the benefit of retaining more directions cannot be attributed simply to a larger norm of the update.

A complementary control tests whether adding arbitrary directions is sufficient. We added isotropic Gaussian noise to the parameter updates, with noise norm equal to 0.1 or 1 times the base update norm, which artificially increases the effective rank of the update. Adding noise does not reproduce Muon's accuracy gains in FA or DFA (Figure 3 F). Together with the norm-matched rank control, it supports the interpretation that distributing updates across directions present in the learning signal is useful, while arbitrary high-dimensional motion alone is insufficient.

Given that an optimiser that increases the dimensionality of the update improves the performance of FA models. We investigated how an optimiser that does the opposite affects FA. To do so, we replaced the orthogonalisation function in Muon with the rank r approximation of the momentum matrix using the SVD of the momentum. We used the rank 2 and rank 5 approximation (Figure 4) and found that while BP models are not so affected, the effect on FA models is catastrophic.

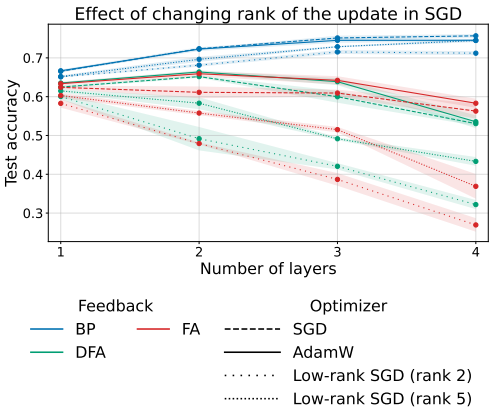


Figure 4: **Using the top-K directions as the opposite of Muon.** Using low-rank SGD results in much worse performance with FA, whereas the effect is not so large with BP.

Taken together, these results support the role of update geometry in improving random-feedback learning. Muon allows weaker singular directions to contribute to learning, and the rank control shows that removing these directions can reintroduce the degradation with depth even when the update norm is held fixed.

Normalising the activities A second way to increase the dimensionality of the updates is to act directly on the activities of the network. As the gradient is the outer product of the error signal and the previous activation, increasing the dimensionality of the activation may increase the dimensionality of the gradients. BN [Ioffe and Szegedy, 2015], has been shown to promote more orthogonal hidden representations in deep networks [Daneshmand et al., 2020, 2021], suggesting that it may prevent the collapse of the representations onto low-dimensional subspaces. Since our previous results indicate that FA fails partly because its gradients and features become too low-dimensional, we added BN layers after each convolutional layer, using the same experimental setup as above.

The results were very similar to those obtained with Muon (Figure 5). BN substantially reduced the degradation in accuracy with depth (Figure 5 A), and increased gradient alignment (Figure 5 B). Consistent with our hypothesis, it also increased the dimensionality of the gradients (Figure 5 C), the

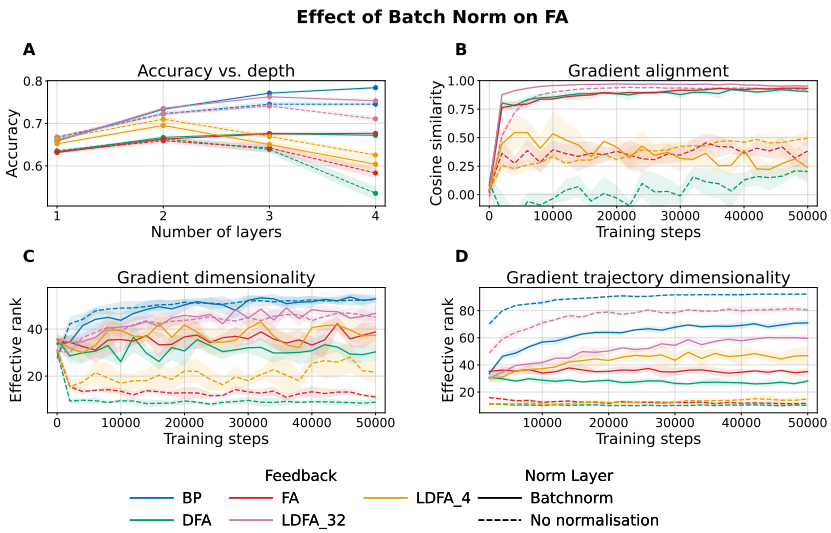


Figure 5: **Optimising FA networks with BN results in better FA. A** Effect of adding batch normalisation layers on the accuracy of FA networks. The accuracy of models trained without BN decreases, while the accuracy of models trained with BN does not. **B-D** Metrics for the last layer of the 4 convolutional layer model. **B** Models with BN layers have higher and smoother gradient alignment. **C** The gradients of models with BN have higher dimensionality. **D** The trajectory of the gradients is also higher for models with BN.

gradient trajectory (Figure 5 D). Thus, improving the geometry of the representations can mitigate the main pathologies of FA, further supporting the view that the poor performance of FA in deep CNNs is due to low-dimensional updates not allowing the alignment process to continue.

Together, these results may explain why BN and Muon produce similar improvements in FA: both help maintain a richer set of directions through which alignment and learning can continue. Conversely, forcing updates to be low-rank severely harms FA, confirming that higher-dimensional optimisation is crucial for robust FA training.

5 Scaling to deeper models

Given that Muon and BN improved the trainability of FA networks in our previous setting, we next asked whether these interventions also scaled to harder datasets and more realistic architectures. We trained AlexNet [Krizhevsky et al., 2012] and ResNet-18 [He et al., 2016] on STL-10 [Coates et al., 2011], CIFAR-100 [Krizhevsky, 2009], and ImageNet [Russakovsky et al., 2015]. For each dataset and architecture, we compared BP, FA, and DFA under four settings: AdamW [Loshchilov and Hutter, 2017], BN, Muon, and Muon combined with BN (Table 1).

Consistent with the results from the previous section, baseline FA performed poorly in deeper models. The degradation was especially severe for ResNet-18, where baseline FA reached only 10.2% on STL-10 and 1.1% on CIFAR-100. Both BN and Muon substantially improved FA performance in all reported dataset–architecture combinations. On STL-10, AlexNet trained with baseline FA reached 49.0%, whereas adding BN increased performance to 69.7%, Muon to 70.8%, and Muon+BN to 76.4%. For ResNet-18, Muon+BN improved performance from 10.2% to 63.0%.

The same trend held on CIFAR-100. Baseline FA reached 13.2% with AlexNet and 1.1% with ResNet-18. Adding either BN or Muon produced gains, and their combination gave the best performance, reaching 49.0% and 46.8%. DFA also benefited from both interventions: Muon+BN increased STL-10 accuracy from 47.3% to 74.3% for AlexNet and from 11.0% to 70.9% for ResNet-18. On CIFAR-100, the corresponding improvements were from 21.4% to 50.3% and from 1.2% to 41.3%. The relative benefit of each intervention varied. For ResNet-18 trained with DFA on CIFAR-100, BN alone reached 36.9%, compared with 10.3% for Muon alone.

Table 1: **Scaling FA and DFA to deeper architectures and harder datasets.** Top-1 accuracy (%, mean ± standard deviation across seeds) of AlexNet and ResNet-18 on STL-10, CIFAR-100, and ImageNet. For each feedback method, we compare AdamW [Loshchilov and Hutter, 2017], BN, Muon, and Muon+BN. Bold indicates the best mean accuracy within each feedback method for each dataset and architecture. NA indicates that the loss diverged for all seeds during the first epoch.

	Dataset	STL-10		CIFAR-100		ImageNet	
	Network	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
Feedback	Setting						
BP	Baseline	71.1 ± 0.7	72.2 ± 1.5	58.6 ± 0.3	63.1 ± 1.1	53.0 ± 0.1	60.3 ± 0.2
	BN	80.8 ± 0.3	78.2 ± 0.6	68.3 ± 0.3	73.7 ± 0.4	53.1 ± 0.2	67.1 ± 0.1
	Muon	85.2 ± 0.2	81.6 ± 0.4	66.0 ± 0.5	69.9 ± 0.4	53.2 ± 0.1	65.9 ± 0.2
	Muon + BN	**86.1 ± 0.3**	**84.1 ± 0.1**	**69.3 ± 0.3**	**75.3 ± 0.3**	**56.6 ± 0.1**	**68.2 ± 0.2**
FA	Baseline	49.0 ± 1.9	10.2 ± 0.4	13.2 ± 1.2	1.1 ± 0.1	0.1 ± 0.0	NA
	BN	69.7 ± 0.9	54.2 ± 0.8	45.0 ± 0.3	34.5 ± 0.8	11.3 ± 0.1	19.0 ± 0.1
	Muon	70.8 ± 0.5	59.8 ± 1.1	45.9 ± 0.5	34.7 ± 0.7	11.7 ± 0.1	6.6 ± 0.2
	Muon + BN	**76.4 ± 0.6**	**63.0 ± 0.5**	**49.0 ± 0.5**	**46.8 ± 0.4**	**20.2 ± 0.2**	**27.4 ± 0.2**
DFA	Baseline	47.3 ± 1.1	11.0 ± 0.8	21.4 ± 1.5	1.2 ± 0.2	0.2 ± 0.0	NA
	BN	69.1 ± 0.9	67.3 ± 0.3	47.7 ± 0.4	36.9 ± 0.3	16.1 ± 0.1	14.9 ± 0.2
	Muon	66.6 ± 1.2	60.0 ± 0.8	46.4 ± 0.3	10.3 ± 0.2	18.9 ± 0.1	2.1 ± 0.2
	Muon + BN	**74.3 ± 0.4**	**70.9 ± 0.6**	**50.3 ± 0.6**	**41.3 ± 0.5**	**26.2 ± 0.1**	**17.6 ± 0.2**

On ImageNet, baseline AlexNet performance was only 0.1% with FA and 0.2% with DFA. With FA, BN increased accuracy to 11.3%, Muon to 11.7%, and Muon+BN to 20.2%. With DFA, the corresponding accuracies were 16.1%, 18.9%, and 26.2%. Thus, the combined intervention enabled both methods to learn on ImageNet, although a substantial gap remained to AlexNet trained with BP, which reached 56.6% with Muon+BN. For the ResNet-18 architecture, baseline FA and DFA diverged on ImageNet. Muon alone did not perform as well on ImageNet, reaching 6.6%. Still, Muon combined with BN performed best, reaching 27.4% with FA and 17.6% with DFA, respectively.

Muon and BN were complementary for both FA and DFA: their combination gave the highest mean accuracy for each benchmark combination. Muon directly modifies the geometry of the parameter update by flattening the spectrum of the momentum, whereas BN acts on the forward representations and changes the geometry of the gradients. The accuracy results are consistent with the benefits of preserving richer learning directions observed in the previous section.

6 Discussion

While feedback alignment was surprisingly shown to work in shallow MLP [Lillicrap et al., 2016], the fact that it did not scale well to deeper architectures raised doubts on whether the brain could be functioning on this principle. Indeed, convolutional networks are not biologically plausible due to neurons sharing exact weights, even though they are inspired by the visual system [Bartunov et al., 2018, Pogodin et al., 2021], this weight sharing is what makes optimisation easier with backpropagation. Therefore, it would be desirable that an approximation to BP also performs well with such architectures. In this work, we presented ways of making FA work in deeper CNNs. Using a different optimiser or by normalising the activities, we found that FA did not have such a steep decrease in accuracy when adding layers as it does when using standard methods. While we did not investigate why alignment happens, as previous work already investigated this question [Lillicrap et al., 2016, Refinetti et al., 2021], we focused on why it seemingly stops after a period of early alignment, and explained how to allow alignment to continue after this period.

Biological plausibility directly motivates the study of FA, but the interventions used here should not be interpreted as biologically plausible learning algorithms in their current form. In particular, Muon requires orthogonalising a matrix-valued momentum update, which depends on coordinated information about the spectrum of an entire weight-update matrix rather than on purely synapse-local quantities. Although local plasticity rules such as Oja-like mechanisms can extract principal directions [Oja, 1982, Sanger, 1989], our implementation of Muon should be viewed as a probe of an alternative rule for updates, not as a direct model of synaptic plasticity.

The other mechanism we investigated was normalising the activities, in our case by using BN [Ioffe and Szegedy, 2015]. Beyond its original motivation as a way of reducing internal covariate shift, BN can be understood as helping deep networks preserve high-dimensional representations across layers. In unnormalised deep networks, the activity matrix over a batch can progressively collapse to a low-rank representation [Huh et al., 2021, Daneshmand et al., 2020]. By contrast, BN counteracts this rank collapse and helps maintain multiple active directions of variation in the hidden activities [Daneshmand et al., 2020, 2021]. In addition, BN often makes training less sensitive to initialisation, permits the use of larger learning rates, and improves gradient propagation [Ioffe and Szegedy, 2015, Santurkar et al., 2018]. While we did not explicitly control for these factors, both Muon and BN improved FA and DFA in all reported settings in Table 1, and their combination gave the highest mean accuracy. Their individual benefits varied across settings, so these results are consistent with complementary effects on learning rather than establishing a single shared mechanism. Although standard BN is not strictly biologically plausible because it relies on batch-level statistics and global normalisation across examples, normalisation mechanisms are widespread in biological neural systems [Carandini and Heeger, 2012]. In particular, divisive normalisation and homeostatic regulation [Wen and Turrigiano, 2024, Shen et al., 2021] could provide more local approximations that similarly stabilise activity scales and prevent representational collapse. Transformations of neural activity at the population level have been studied in the visual cortex [Failor et al., 2025], and are thought to be behind its high dimensionality [Rigotti et al., 2013, Fusi et al., 2016, Stringer et al., 2019, Failor et al., 2025].

This perspective is also consistent with a growing body of work studying the dimensionality of neural population activity in the visual cortex. Rather than being confined to a small number of easily interpretable axes, cortical activity often occupies high-dimensional, structured subspaces, which can support flexible downstream readout and improve the separability of behaviourally relevant variables [Rigotti et al., 2013, Fusi et al., 2016, Failor et al., 2025]. This has been studied particularly clearly in the visual cortex, where large-scale recordings in awake mice showed that responses to natural images span many dimensions and exhibit a scale-free eigenspectrum, suggesting that visual representations preserve a rich set of stimulus-related directions rather than collapsing onto a few dominant modes [Stringer et al., 2019]. This high dimensionality is what would allow FA alignment to work well.

Beyond biological plausibility, feedback alignment may be useful as a distinct optimisation system from standard backpropagation. Because FA uses fixed random feedback, it follows different parameter-space trajectories and can expose how optimisation techniques behave when the learning signal is only approximate. This makes FA a useful testbed for studying whether methods such as normalisation or update orthogonalisation improve learning by better conditioning BP gradients specifically, or by more generally shaping update geometry. Moreover, alternative credit-assignment rules can affect loss-landscape exploration [Pascanu et al., 2025], for example, FA could use structured feedback that guides learning to different solutions than what BP would reach. For instance, FA has been shown to allow better continual learning [Folchini et al., 2025].

Acknowledgements

This work was supported by Wellcome Trust 200790/Z/16/Z, Simons Foundation 564408 and EPSRC EP/R035806/1.

References

Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.

Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. Sharpness-aware minimization leads to low-rank features. *Advances in Neural Information Processing Systems*, 36: 47032–47051, 2023.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International conference on machine learning*, pages 244–253. PMLR, 2018.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in neural information processing systems*, 32, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.

Jeremy Bernstein. Deriving muon, 2025. URL `https://jeremybernste.in/writing/deriving-muon`.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Akhilan Boopathy and Ila Fiete. How to train your wide neural network without backprop: An input-weight alignment perspective. In *International Conference on Machine Learning*, pages 2178–2205. PMLR, 2022.

Matteo Carandini and David J Heeger. Normalization as a canonical neural computation. *Nature reviews neuroscience*, 13(1):51–62, 2012.

Jeonghwan Cheon, Sang Wan Lee, and Se-Bum Paik. Pretraining with random noise for fast and robust learning without weight transport. *Advances in Neural Information Processing Systems*, 37: 13748–13768, 2024.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

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

Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33:18387–18398, 2020.

Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. *Advances in Neural Information Processing Systems*, 34: 4896–4906, 2021.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in neural information processing systems*, 31, 2018.

Samuel W Failor, Matteo Carandini, and Kenneth D Harris. Visual experience orthogonalizes visual cortical stimulus responses via population code transformation. *Cell Reports*, 44(2), 2025.

Sara Folchini, Viplove Arora, and Sebastian Goldt. Exploring the potential of direct feedback alignment for continual learning. *Transactions on Machine Learning Research*, 2025.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in neuroscience*, 15:629892, 2021.

Kenji Fukumizu. Effect of batch learning in multilayer neural networks. *Gen*, 1(04):1E–03, 1998.

Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current opinion in neurobiology*, 37:66–74, 2016.

Manuela Girotti, Ioannis Mitliagkas, and Gauthier Gidel. Convergence analysis and implicit regularization of feedback alignment for deep linear networks. *arXiv preprint arXiv:2110.10815*, 2021.

Maher Hanut and Jonathan Kadmon. Training large neural networks with low-dimensional error feedback. *arXiv preprint arXiv:2502.20580*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Amir Joudaki, Hadi Daneshmand, and Francis Bach. On the impact of activation and normalization in obtaining isometric embeddings at initialization. *Advances in Neural Information Processing Systems*, 36:39855–39875, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Daniel Kunin, Aran Nayebi, Javier Sagastuy-Brena, Surya Ganguli, Jonathan Bloom, and Daniel Yamins. Two routes to scalable credit assignment without weight symmetry. In *International Conference on Machine Learning*, pages 5511–5521. PMLR, 2020.

Julien Launay, Iacopo Poli, and Florent Krzakala. Principled training of neural networks with direct feedback alignment. *arXiv preprint arXiv:1906.04554*, 2019.

Qianli Liao, Joel Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1): 13276, 2016.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Theodore H Moskovitz, Ashok Litwin-Kumar, and LF Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Joshua Offergeld and Nasir Ahmad. Correlations are ruining your gradient descent, 2026. URL `https://arxiv.org/abs/2407.10780`.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Razvan Pascanu, Clare Lyle, Ionut-Vlad Modoranu, Naima Elosegui Borras, Dan Alistarh, Petar Velickovic, Sarath Chandar, Soham De, and James Martens. Optimizers qualitatively alter solutions and we should leverage this. *arXiv preprint arXiv:2507.12224*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Roman Pogodin, Yash Mehta, Timothy Lillicrap, and Peter E Latham. Towards biologically plausible convolutional networks. *Advances in neural information processing systems*, 34:13924–13936, 2021.

Maria Refinetti, Stéphane d'Ascoli, Ruben Ohana, and Sebastian Goldt. Align, then memorise: the dynamics of learning with feedback alignment. In *International Conference on Machine Learning*, pages 8925–8935. PMLR, 2021.

Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497 (7451):585–590, 2013.

Zachary Robertson and Oluwasanmi Koyejo. Implicit regularization in feedback alignment learning mechanisms for neural networks. *arXiv preprint arXiv:2306.01870*, 2023.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Yang Shen, Julia Wang, and Saket Navlakha. A correspondence between normalization strategies in artificial and biological neural networks. *Neural computation*, 33(12):3179–3203, 2021.

Zakhar Shumaylov, Nathaël Da Costa, Peter Zaika, Bálint Mucsányi, Alex Massucco, Yoav Gelberg, Carola-Bibiane Schönlieb, Yarin Gal, and Philipp Hennig. Muon is not that special: Random or inverted spectra work just as well. *arXiv preprint arXiv:2605.11181*, 2026.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

Wei Wen and Gina G Turrigiano. Keeping your brain in balance: homeostatic regulation of network function. *Annual review of neuroscience*, 47, 2024.

Will Xiao, Honglin Chen, Qianli Liao, and Tomaso Poggio. Biologically-plausible learning algorithms can scale to large datasets. *arXiv preprint arXiv:1811.03567*, 2018.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

A Feedback methods and implementation

In this section, we describe the implementation details we omitted in the main text of the four feedback methods: BP, FA, DFA, and LDFA, presented in the main text and illustrated in Figure 1. All four methods use the same forward operations and compute the task loss at the output. They differ in how the output error reaches the hidden layers and whether the feedback weights are tied to the forward weights, fixed, or learned.

Backpropagation. BP [Rumelhart et al., 1986] uses the transposed forward weights to propagate the output error back to the hidden layers. This is the standard method for training neural networks and usually serves as an upper bound for comparison with feedback methods.

Feedback alignment. FA [Lillicrap et al., 2016] replaces the input derivative of each linear or convolutional operation by a projection through a fixed random map. Feedback tensors are initialised with Kaiming-uniform sampling, same as the forward weights.

Direct feedback alignment. DFA [Nøkland, 2016] bypasses the intervening feedback path. In our implementation, the feedback is sent after the block's normalisation, activation, and pooling. The projection is reshaped to the block output, and differentiation through those local operations then produces the convolutional parameter updates.

Following [Launay et al., 2019], the direct maps share one fixed random matrix. If the output has C classes and the largest hidden state has $m_{\max}$ elements, we sample $R \in \mathbb{R}^{C \times m_{\max}}$ with independent entries uniform on $[-m_{\max}^{-1/2}, m_{\max}^{-1/2}]$. For a hidden state with m_ℓ elements, the effective map is

$$D_\ell = \sqrt{\frac{m_{\max}}{m_\ell}} \left(R_{:,1:m_\ell}\right)^\top. \tag{7}$$

Thus, the maps have the appropriate hidden-size scaling but share entries across layers, rather than being independently sampled. For a convolutional block, m_ℓ includes both channels and spatial positions after pooling. It is necessary to have such a shared structure as using distinct feedback maps uses too much memory. Moreover, as explained in [Refinetti et al., 2021], it is fine to have aligned projections, as such alignment naturally occurs in linear BP networks [Ji and Telgarsky, 2018]. We also experimented with independent projections for each layer on CIFAR10, but found that this did not improve performance.

Low-dimensional feedback alignment. We followed the local formulation of LDFA presented in Hanut and Kadmon [2025]. LDFA propagates errors between adjacent layers using a factorised map $B_\ell = Q_\ell P_\ell$. For $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, we use $Q_\ell \in \mathbb{R}^{n_{\ell-1} \times r_\ell}$ and $P_\ell \in \mathbb{R}^{r_\ell \times n_\ell}$, so the dense feedback map has rank at most r_ℓ. Both factors are learned alongside the forward weights using local signals. An Oja-type rule adapts P_ℓ to the leading directions of the incoming error's second moment, while a Hebbian rule updates Q_ℓ using the correlation between $h_{\ell-1}$ and the compressed error $P_\ell \delta_\ell^{LDFA}$. This allows the feedback subspace to adapt during training while retaining the dimensionality constraint. Q_ℓ is initialised using Kaiming-uniform and P_ℓ starts with orthonormal rows. We trained them using Adamw and included the learning rate in the hyperparameter search. For convolutional layers, P_ℓ is a shared 1×1 projection from C_{out} error channels to r_ℓ channels, followed by the adjoint operation associated with a learned spatial kernel Q_ℓ.

B Biologically plausible methods

As mentioned in the discussion, the interventions used in the main text are not biologically plausible in their current implementation. We also tried two biologically plausible methods with low computational cost to increase the dimensionality of the updates: adding a local loss at each layer and using Layer Normalisation (LN) [Ba et al., 2016]. The results appear similar to those obtained with Muon and BN.

B.1 Local loss

We added a local loss at each layer, aimed at making the activity dimensionality higher. Specifically, we used a loss introduced in [Cogswell et al., 2015] that minimises the covariance of the channels

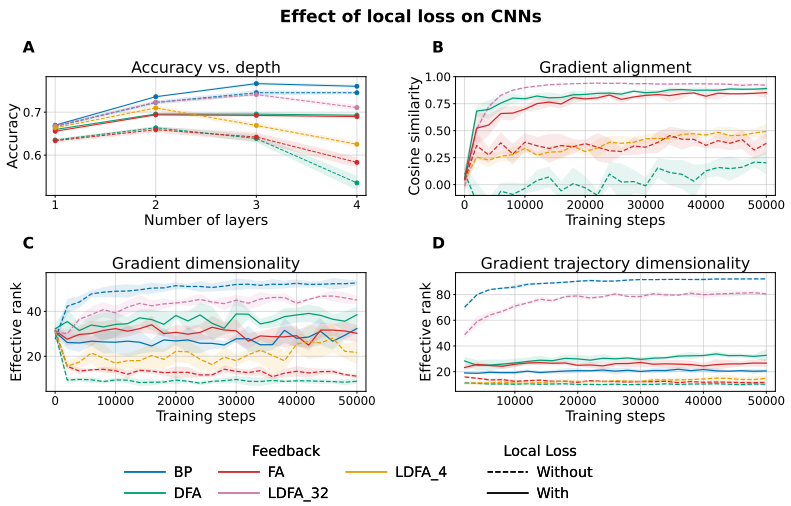


Figure 6: **Adding a local loss to the FA gradients. A** Adding layers on the accuracy of FA networks. The accuracy of models trained with standard optimisers decreases, while those trained with the local loss do not. **B-D** Metrics for the 4 convolutional layer model. **B** Models trained with local loss have smoother and higher gradient alignment. **C** The gradients of models trained with the local loss have higher effective rank. **D** The trajectory of the gradients is also higher for models trained with the local loss.

inside a layer; we did not backpropagate the loss of one layer to the previous layers. We followed the formulation used in [Bardes et al., 2021]:

$$\mathcal{L}_{\text{decor}}(X) = \frac{1}{D}\sum_{\substack{i,j=1 \\ i\neq j}}^{D}\left[\frac{1}{B-1}\sum_{b=1}^{B}\left(x_{bi}-\bar{x}_i\right)\left(x_{bj}-\bar{x}_j\right)\right]^2, \bar{x}_i = \frac{1}{B}\sum_{b=1}^{B}x_{bi} \tag{8}$$

It minimises the off-diagonal components of the covariance of the activity in a given layer for one batch. The goal of this loss is to make the dimensionality higher by reducing the redundancies of the representations inside a layer. It is related to a lower bound approximation to the rank of a matrix introduced in [Daneshmand et al., 2020] $r(X) = \frac{\text{Tr}(M(X))^2}{\|M(X)\|_F^2}, M(X) = \frac{X^\top X}{N}$, as minimising the off diagonal components of the covariance will have a similar objective to maximising the ratio of the squared trace to the squared frobenius norm of the covariance. We add the gradient of this loss weighted by λ to the gradient of the cross-entropy loss used for classification. We optimised the network using AdamW [Loshchilov and Hutter, 2017].

We ran the same analysis as the one we did on Muon, and the results appear similar (Figure 6). The accuracy does not decrease much when adding layers (Figure 6 A). When looking at the 4 convolutional layers network, we also see better behavior (Figure 6 B-D). The gradients are more aligned, and the alignment is more stable (Figure 6 B). The geometry of the updates is also better behaved. The gradients are higher-dimensional (Figure 6 C), and their trajectory is also higher-dimensional (Figure 6 D). The local loss did not scale well when using the deeper networks on the datasets we tested.

B.2 Layer normalisation

We also normalised the activities within each layer using LN [Ba et al., 2016]. For each example, LN computes the mean and variance across the units in the layer, without using statistics from other examples in the mini-batch, which makes it more biologically plausible. Unlike the local loss, LN does not explicitly minimise correlations between units, however, similarly to BN, LN was found to promote isometry in the activities [Joudaki et al., 2023].

We repeated the analysis used for the local loss and found similar effects for FA and DFA (Figure 7). Without normalisation, their accuracy decreases as the number of convolutional layers increases

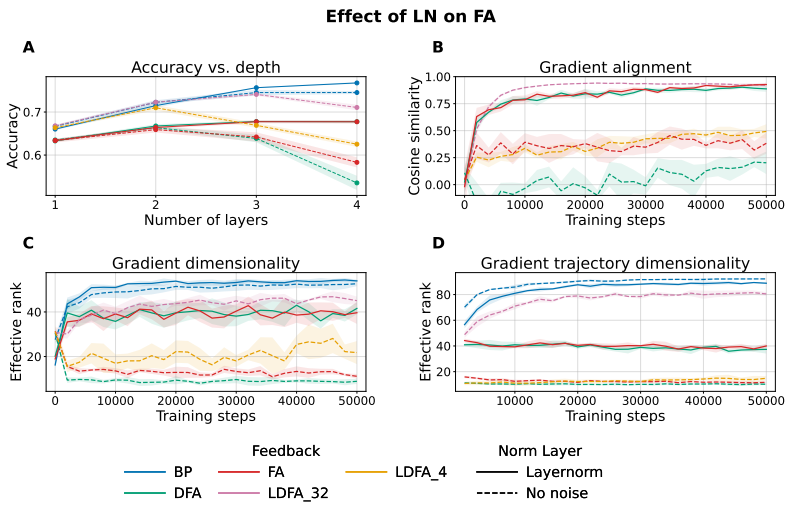


Figure 7: **Layer normalisation improves the trainability of FA networks. A** Accuracy as a function of network depth. Adding LN reduces the loss of accuracy with increasing depth for FA and DFA. **B-D** Metrics for the last convolutional layer of the four convolutional-layer network. **B** With LN, FA and DFA have higher and more stable gradient alignment. **C** Their gradients maintain a higher effective rank. **D** Their gradient trajectories also have higher effective rank.

beyond two, whereas with LN it remains approximately stable (Figure 7 A). In the four convolutional-layer network, both methods achieve higher and more stable gradient alignment with LN (Figure 7 B). Their gradients also maintain a higher effective rank throughout training (Figure 7 C), and their gradient trajectories span a higher-dimensional subspace (Figure 7 D). These results are consistent with the view that normalising the activities helps sustain alignment by preventing the learning signal from becoming confined to a small number of directions. Table 2 extends the comparison of LN and Muon to AlexNet and ResNet-18 on STL-10, CIFAR-100, and Tiny ImageNet, a subset of ImageNet [Russakovsky et al., 2015]. LN performs slightly worse than BN in general. The combination of Muon and LN still achieves the highest reported mean accuracy for each benchmark pair.

Table 2: **Layer normalisation and Muon in AlexNet and ResNet-18.** Top-1 accuracy (%, mean $\pm$ standard deviation across seeds) on STL-10, CIFAR-100, and Tiny ImageNet for BP, FA and DFA. Baseline uses AdamW [Loshchilov and Hutter, 2017] without normalisation; LN adds layer normalisation, Muon changes the optimiser, and Muon + LN combines both interventions. Bold marks the highest mean accuracy within each feedback method for each dataset and architecture.

	Dataset	STL-10		CIFAR-100		Tiny ImageNet	
	Network	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
Feedback	Setting						
BP	Baseline	71.1 $\pm$ 0.7	72.2 $\pm$ 1.5	58.6 $\pm$ 0.3	63.1 $\pm$ 1.1	38.2 $\pm$ 0.4	31.0 $\pm$ 0.4
	LN	74.6 $\pm$ 0.3	70.7 $\pm$ 0.4	65.6 $\pm$ 0.2	68.6 $\pm$ 0.4	47.8 $\pm$ 0.1	39.1 $\pm$ 0.3
	Muon	85.2 $\pm$ 0.2	81.6 $\pm$ 0.4	66.0 $\pm$ 0.5	69.9 $\pm$ 0.4	46.0 $\pm$ 0.2	37.4 $\pm$ 0.1
	Muon + LN	**85.5 $\pm$ 0.3**	**83.0 $\pm$ 0.5**	**69.7 $\pm$ 0.2**	**75.4 $\pm$ 0.2**	**49.7 $\pm$ 0.2**	**46.1 $\pm$ 0.3**
FA	Baseline	49.0 $\pm$ 1.9	10.2 $\pm$ 0.4	13.2 $\pm$ 1.2	1.1 $\pm$ 0.1	9.2 $\pm$ 1.5	0.5 $\pm$ 0.1
	LN	58.1 $\pm$ 1.7	38.7 $\pm$ 1.3	36.8 $\pm$ 1.2	23.7 $\pm$ 0.9	29.4 $\pm$ 0.4	16.7 $\pm$ 0.2
	Muon	70.8 $\pm$ 0.5	59.8 $\pm$ 1.1	**45.9 $\pm$ 0.5**	34.7 $\pm$ 0.7	31.7 $\pm$ 0.4	14.7 $\pm$ 0.4
	Muon + LN	**74.3 $\pm$ 0.7**	**63.8 $\pm$ 0.7**	45.9 $\pm$ 0.5	**43.8 $\pm$ 0.8**	**33.4 $\pm$ 0.6**	**21.9 $\pm$ 0.6**
DFA	Baseline	47.3 $\pm$ 1.1	11.0 $\pm$ 0.8	21.4 $\pm$ 1.5	1.2 $\pm$ 0.2	12.2 $\pm$ 0.7	0.6 $\pm$ 0.1
	LN	65.5 $\pm$ 0.3	64.5 $\pm$ 0.2	44.6 $\pm$ 0.5	34.2 $\pm$ 0.3	32.1 $\pm$ 0.4	24.4 $\pm$ 0.2
	Muon	66.6 $\pm$ 1.2	60.0 $\pm$ 0.8	46.4 $\pm$ 0.3	10.3 $\pm$ 0.2	27.6 $\pm$ 0.1	8.3 $\pm$ 0.6
	Muon + LN	**73.3 $\pm$ 0.8**	**69.7 $\pm$ 0.3**	**49.1 $\pm$ 0.5**	**42.3 $\pm$ 0.3**	**32.5 $\pm$ 0.2**	**25.1 $\pm$ 0.1**

C Interpolating between SGD and Muon

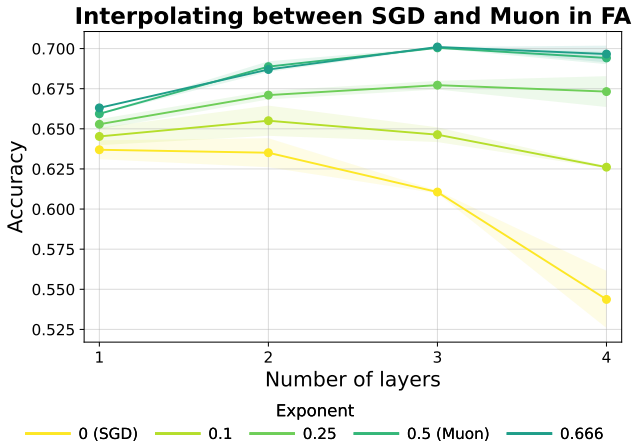


Figure 8: **A more balanced singular value spectrum improves performance.** Increasing the exponent in Freon optimiser [Shumaylov et al., 2026], which, until 0.5, flattens the spectrum of the updates, then makes it skew the other way, increases the performance of the FA networks.

To better understand how the distribution of the singular values of the updates impacts the performance of the network, we used the Freon optimiser [Shumaylov et al., 2026], which transforms the momentum matrix M to: $U\ \mathrm{diag}\left(\left(\frac{\sigma}{\|M\|_q}\right)^{q-1}\right) V^\top$, where $U\ \mathrm{diag}(\sigma)\, V^\top$ is the SVD of M. The exponent $c = 1 - q/2$ controls the transformation of the singular values. With $c = 0$, the update is equivalent to SGD with momentum, and directions with large singular values dominate the update. As c increases, the spectrum of the update becomes progressively flatter, giving more weight to directions that would otherwise contribute very little. Using $c = 0.5$ corresponds to Muon, where all non-zero singular directions are weighted equally. Using $c > 0.5$ scales up the smaller singular values and scales down the bigger ones. This interpolation is relevant for our work since c controls the dimensionality of the updates, complementing the norm-matched rank control in Figure 3 E. When applied to FA models, performance improves as the Freon exponent is increased from the SGD regime toward the Muon regime (Figure 8), supporting the claim that making the update higher-dimensional improves FA.

D Theoretical analysis of Feedback Alignment

In this section, we first explain when imperfect alignment may slow down optimisation, then we provide intuition for why the singular values can become more concentrated across layers when training with FA. By doing so, we bring new perspectives to past theoretical analysis of FA [Lillicrap et al., 2016, Girotti et al., 2021, Robertson and Koyejo, 2023]. Much of the theory on neural network learning dynamics is done in the analytically tractable setting of linear networks [Fukumizu, 1998, Saxe et al., 2013, Arora et al., 2018, 2019]. We follow this setup, and consider a linear network without biases, with weights $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and end-to-end weights $W_e = W_L \cdots W_1$. Throughout this section, we use gradient flow with a learning rate $\eta > 0$, and no momentum or weight decay. We write $G = \nabla_{W_e} E(W_e)$ for the gradient of the loss with respect to the end-to-end weights.

First, for BP, Arora et al. [2018] showed that depth changes the optimisation dynamics of linear networks by preconditioning the end-to-end gradient:

$$\dot{W}_e^{BP} = -\eta \sum_{j=1}^{L} \left[W_e^{BP}(W_e^{BP})^\top\right]^{\frac{j-1}{L}} G \left[(W_e^{BP})^\top W_e^{BP}\right]^{\frac{L-j}{L}}. \tag{9}$$

This results from the layers of the network being balanced, which means that if the weights are initialised such that $W_j W_j^\top = W_{j+1}^\top W_{j+1}$ then it holds throughout training. The preconditioner depends on the singular values and singular vectors of the end-to-end weights. This relation between the

geometry of the weights and the optimisation dynamics is central to studies of implicit regularisation in deep linear networks [Arora et al., 2019, Ji and Telgarsky, 2018].

To obtain the FA end-to-end dynamics, we define the forward prefix $A_\ell = W_{\ell-1} \cdots W_1$, the forward suffix $R_\ell = W_L \cdots W_{\ell+1}$, and the cumulative feedback map $F_\ell = B_{\ell+1} \cdots B_L$, with $A_1 = I_{n_0}$ and $R_L = F_L = I_{n_L}$. For linear activations, Eq. 3 gives $\dot{W}_\ell = -\eta F_\ell G A_\ell^\top$. Applying the product rule, $\dot{W}_e^{FA} = \sum_{\ell=1}^{L} R_\ell \dot{W}_\ell A_\ell$, therefore gives:

$$\dot{W}_e^{FA} = -\eta \sum_{\ell=1}^{L} (R_\ell F_\ell) G (A_\ell^\top A_\ell). \tag{10}$$

Writing Eq. 10 in vectorised form, with $g = \mathrm{vec}(G)$ and columns stacked in vec, gives:

$$\mathrm{vec}(\dot{W}_e^{FA}) = -\eta \mathcal{P} g, \qquad \mathcal{P} = \sum_{\ell=1}^{L} (A_\ell^\top A_\ell) \otimes (R_\ell F_\ell). \tag{11}$$

Unlike the preconditioning induced by BP, $\mathcal{P}$ is not necessarily positive semidefinite, therefore the preconditioning may not keep the loss decreasing. To see that, we can compute the change in loss along this flow, by applying the chain rule to $E(W_e(t))$:

$$\dot{E} = \mathrm{tr}(G^\top \dot{W}_e^{FA}) = g^\top \mathrm{vec}(\dot{W}_e^{FA}) = -\eta g^\top \mathcal{P} g,$$

where the last equality uses Eq. 11. Since $g^\top \mathcal{P} g$ is a scalar, it equals its transpose $g^\top \mathcal{P}^\top g$. Averaging the two expressions gives

$$g^\top \mathcal{P} g = \frac{1}{2} \left(g^\top \mathcal{P} g + g^\top \mathcal{P}^\top g \right) = g^\top \mathrm{sym}(\mathcal{P}) g,$$

with $\mathrm{sym}(\mathcal{P}) = (\mathcal{P} + \mathcal{P}^\top)/2$. Thus, only the symmetric part of $\mathcal{P}$ contributes to the instantaneous change in loss. The loss therefore decreases whenever $g^\top \mathrm{sym}(\mathcal{P}) g > 0$. Positive semidefiniteness of $\mathrm{sym}(\mathcal{P})$ guarantees that the loss does not increase for any gradient, but is not necessary for a decrease along the current gradient. Still, poor alignment can make the quadratic form small or negative, resulting in slow progress or an increase in loss.

To understand how the weight spectra can change across layers in FA, we first explain that singular values are shared across layers in BP. We then derive an identity related to the balanced-layer invariant in BP [Arora et al., 2018, Du et al., 2018]. This identity was used by Lillicrap et al. [2016] for two-layer linear networks. A related neuronwise conservation law for nonlinear networks was studied by Robertson and Koyejo [2023].

For balanced linear BP, the relation $W_\ell W_\ell^\top = W_{\ell+1}^\top W_{\ell+1}$ is preserved by gradient flow when it holds at initialisation. The nonzero eigenvalues of these Gram matrices are the squared nonzero singular values of the corresponding weights, so adjacent layers have identical nonzero singular values: $\sigma_i(W_\ell) = \sigma_i(W_{\ell+1})$ for $i = 1, \ldots, r$, where r is their common rank. Thus, whenever $r > 0$, the ratios $\sigma_i(W_\ell)/\sigma_1(W_\ell)$ are the same across layers, although the shared spectrum can evolve during training.

For a linear network trained by FA gradient flow with fixed feedback weights, the quantity

$$C_\ell = W_\ell W_\ell^\top - B_{\ell+1} W_{\ell+1} - W_{\ell+1}^\top B_{\ell+1}^\top \tag{12}$$

is conserved for every adjacent pair of layers, $\ell = 1, \ldots, L-1$. In particular, $C_\ell(t) = C_\ell(0)$ throughout training.

Proof. The layer dynamics are $\dot{W}_\ell = -\eta F_\ell G A_\ell^\top$. Using $F_\ell = B_{\ell+1} F_{\ell+1}$ and $A_{\ell+1} = W_\ell A_\ell$, we obtain:

$$\begin{aligned} \dot{W}_\ell W_\ell^\top &= -\eta B_{\ell+1} F_{\ell+1} G A_\ell^\top W_\ell^\top \\ &= -\eta B_{\ell+1} F_{\ell+1} G A_{\ell+1}^\top = B_{\ell+1} \dot{W}_{\ell+1}. \end{aligned} \tag{13}$$

Since $B_{\ell+1}$ is fixed, differentiating Eq. 12 and applying this relation and its transpose gives:

$$\dot{C}_\ell = \dot{W}_\ell W_\ell^\top + W_\ell \dot{W}_\ell^\top - B_{\ell+1} \dot{W}_{\ell+1} - \dot{W}_{\ell+1}^\top B_{\ell+1}^\top = 0.$$

Hence C_ℓ remains equal to its initial value. □

General relation between layer spectra. Let $W_{\ell+1}v_i = \tau_i u_i$ be a singular pair of the upper layer, with $\tau_i > 0$ and unit vectors u_i, v_i. Write $s_j = \sigma_j(W_\ell)$ and let p_j be an orthonormal eigenbasis of $W_\ell W_\ell^\top$. Define the directional feedback gain $\alpha_i = v_i^\top B_{\ell+1} u_i$ and the initialisation contribution $c_i = v_i^\top C_\ell v_i$. The invariant gives the exact relation:

$$2\alpha_i\tau_i = \|W_\ell^\top v_i\|_2^2 - c_i = \sum_{j=1}^{n_\ell} s_j^2 |\langle p_j, v_i\rangle|^2 - c_i. \tag{14}$$

Thus, the upper-layer singular values depend on the squared lower-layer spectrum, the overlap between neighbouring singular directions, and the feedback gain along each direction.

Proof. Multiplying Eq. 12 on the left by $v_i^\top$ and on the right by v_i gives:

$$\begin{aligned}\|W_\ell^\top v_i\|_2^2 - c_i &= v_i^\top B_{\ell+1} W_{\ell+1} v_i + v_i^\top W_{\ell+1}^\top B_{\ell+1}^\top v_i \\ &= 2\tau_i v_i^\top B_{\ell+1} u_i = 2\alpha_i\tau_i.\end{aligned}$$

Expanding v_i in the basis (p_j) gives the sum in Eq. 14. □

An identity on relative singular values follows by controlling the directional quantities above. Order the singular values of both layers from largest to smallest, assume $s_1 > 0$, and define

$$e_i = \|W_\ell^\top v_i\|_2^2 - s_i^2 - c_i. \tag{15}$$

This residual includes both the mixing of lower-layer modes and the contribution of the conserved initialisation term. Eq. 14 gives:

$$\tau_i = \frac{s_i^2 + e_i}{2\alpha_i} \tag{16}$$

If the residuals vanish and the gains are equal, the ratios are squared exactly, as illustrated by the following special case.

Spectral squaring across layers. As an example of Eq. 14, consider square layers of equal width and scaled-orthogonal feedback, $B_{\ell+1} = b_{\ell+1}Q_{\ell+1}$, where $b_{\ell+1} > 0$ and $Q_{\ell+1}^\top Q_{\ell+1} = I$. Keeping the notation $s_i = \sigma_i(W_\ell)$ and $\tau_i = \sigma_i(W_{\ell+1})$, ordered from largest to smallest. Assuming that $C_\ell = 0$ and $K_\ell = \mathrm{skew}(Q_{\ell+1}W_{\ell+1}) = 0$, where $\mathrm{skew}(X) = (X - X^\top)/2$. Then:

$$\tau_i = \frac{s_i^2}{2b_{\ell+1}}, \qquad \frac{\tau_i}{\tau_1} = \left(\frac{s_i}{s_1}\right)^2. \tag{17}$$

where the ratio assumes $s_1 > 0$. Thus, a mode with $0 < s_i/s_1 < 1$ becomes relatively smaller in the next layer. Using recursive application across layers, write $s_i^{(k)} = \sigma_i(W_k)$, so that $s_i = s_i^{(\ell)}$ and $\tau_i = s_i^{(\ell+1)}$. If these conditions hold at every layer, then

$$\frac{s_i^{(k)}}{s_1^{(k)}} = \left(\frac{s_i^{(1)}}{s_1^{(1)}}\right)^{2^{k-1}}. \tag{18}$$

Proof. Eq. 12 can be rearranged as:

$$Q_{\ell+1}W_{\ell+1} = \frac{W_\ell W_\ell^\top - C_\ell}{2b_{\ell+1}} + K_\ell, \qquad K_\ell = \mathrm{skew}(Q_{\ell+1}W_{\ell+1}). \tag{19}$$

The invariant therefore determines the symmetric part of $Q_{\ell+1}W_{\ell+1}$; its skew-symmetric part K_ℓ can still change during training. When $C_\ell = 0$ and $K_\ell = 0$, this reduces to $Q_{\ell+1}W_{\ell+1} = W_\ell W_\ell^\top/(2b_{\ell+1})$. Using the eigenbasis (p_i) from the general relation, we can therefore choose $v_i = p_i$ and $u_i = Q_{\ell+1}^\top p_i$, which gives

$$W_{\ell+1}v_i = \frac{s_i^2}{2b_{\ell+1}}u_i, \qquad \alpha_i = v_i^\top B_{\ell+1}u_i = b_{\ell+1}, \qquad c_i = e_i = 0.$$

Thus, for each positive mode, Eq. 14 reduces to $2b_{\ell+1}\tau_i = s_i^2$. Dividing by the leading singular value gives the squared-ratio relation in Eq. 17. Application across layers gives Eq. 18. □

A similar relation was found in [Girotti et al., 2021] for diagonal linear FA dynamics, where the feedback is fixed and the weights are initialised to be diagonal. In that setting, the singular values of each layer are simply the absolute values of the diagonal entries, and the squared-ratio relation holds exactly. The more general relation in Eq. 14 shows that this behaviour can also emerge in non-diagonal networks, provided that the directional feedback gains are similar across modes and that the residuals are small.

These results identify a mechanism by which FA can make optimisation harder than BP. In balanced linear BP, adjacent layers have identical nonzero singular values, preserving their relative spectra across depth. In FA, the invariant instead couples upper-layer singular values to mixtures of squared lower-layer singular values, modulated by directional feedback gains and the conserved initialisation term (Eq. 14). Under the spectral-squaring conditions, successive layers increasingly separate weak modes from dominant ones (Eq. 18), consistent with related power relations in diagonal linear FA dynamics [Girotti et al., 2021]. Task-relevant errors in these directions can consequently receive much smaller corrective updates than errors in dominant directions, even when the overall update decreases the loss. BP can also exhibit unequal learning rates across directions, but its end-to-end preconditioner is positive semidefinite. FA lacks this guarantee, allowing poor alignment to further reduce progress or even increase the loss. Spectral concentration can therefore compound imperfect feedback alignment, slowing the optimisation of task-relevant errors outside the dominant learning directions.

E Additional metrics

For each panel, every seed is summarised by the mean of the metric over the last 20% of training. Each group is then compared with its reference at the same depth using a two-sided Welch's t-test (5 seeds per group). The references are BP for the feedback methods, AdamW for Muon, and no normalisation for BN. Muon and BN are compared with their reference separately for each feedback method. Every group uses the hyperparameter configuration with the highest mean test accuracy for that depth. p-values are Holm-Bonferroni corrected across all comparisons in a figure. Symbols to the right of each panel give the corrected significance (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, n.s. not significant), and arrows show whether the group lies above (↑) or below (↓) its reference.

E.1 Comparing feedback methods without Muon or Batch Norm

In deep networks, FA and DFA produce gradients with significantly lower effective rank than BP in almost every layer (Figure 9). The gap widens with layer index: in four layer networks, the reduction grows from 12% in the first layer to 76% in the last layer for FA, and from 25% to 83% for DFA. LDFA with rank 4 shows a weaker but still significant reduction. LDFA with rank 32 stays close to BP, differing significantly only in the last layer. The deficit appears with depth: in one- and two-layer networks, FA and DFA actually have a significantly higher first-layer gradient rank than BP. Gradient trajectories show the same pattern (Figure 10). In four layer networks, FA and DFA trajectories are 64-89% lower-dimensional than those of BP beyond the first layer. LDFA with rank 4 is significantly lower at every depth.

E.2 Comparing Muon with AdamW

Muon significantly increases the gradient effective rank of FA and DFA networks in every layer of four layer networks (Figure 11), by 28-154% for FA and 50-277% for DFA. The effect is largest in the deepest layers, where AdamW-trained networks are most collapsed. In contrast, Muon barely changes the gradient rank of BP (5-16%, mostly not significant). For LDFA, it only raises the rank significantly in the last layer. The trajectory dimensionality of FA and DFA also increases significantly (Figure 12; 29-94% and 63-124%). For BP, Muon slightly but significantly lowers trajectory dimensionality in the deeper layers. For LDFA with rank 32, it lowers it in most layers. The benefit of Muon is therefore specific to methods whose updates are low-dimensional under AdamW.

E.3 Comparing Batch Norm with no normalisation

BN has an effect similar to Muon. It significantly increases the gradient effective rank of FA and DFA networks at depths three and four (Figure 13). In four layer networks, the increase grows from

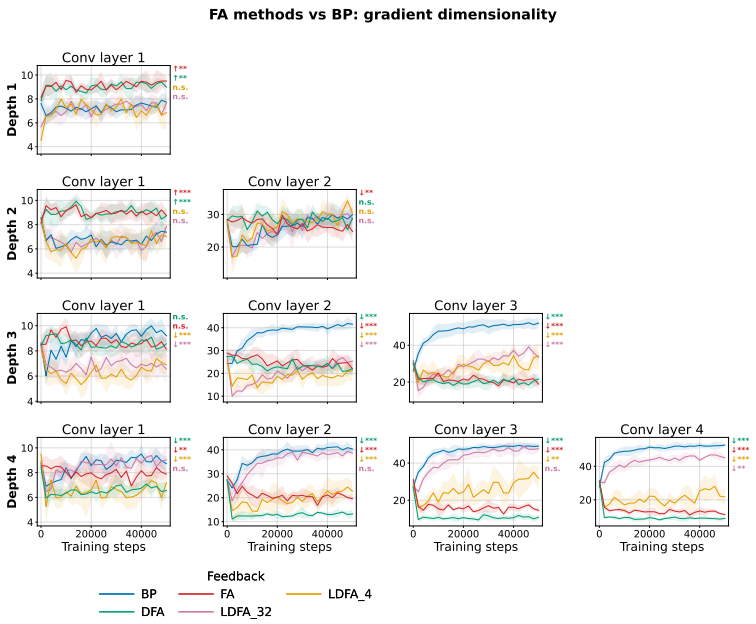


Figure 9: **FA and DFA have lower gradient dimensionality than BP in deep networks.** Effective rank of the gradients over training for each layer (columns) and network depth (rows), all trained with AdamW and without normalisation. Lines show the mean over 5 seeds and shading ± 1 standard deviation.

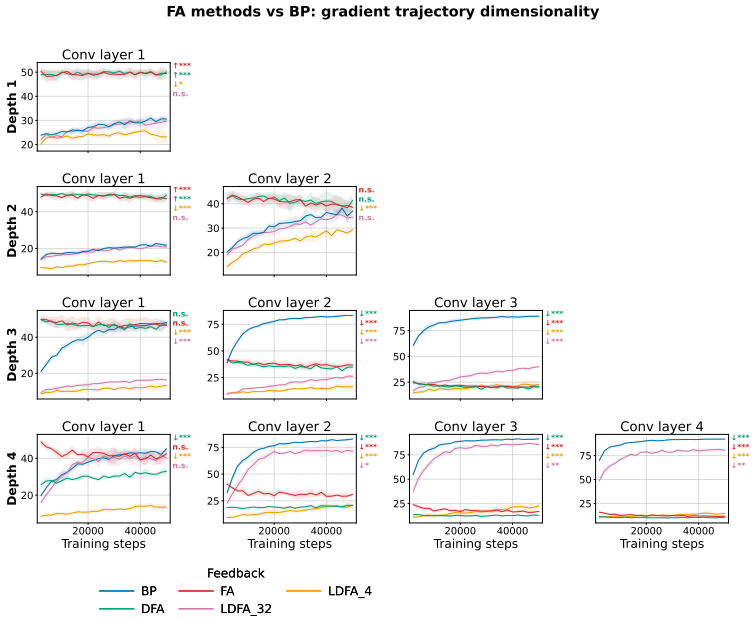


Figure 10: **FA and DFA have lower-dimensional gradient trajectories than BP in deep networks.** Effective rank of the Gram matrix of the gradient trajectory for each layer (columns) and network depth (rows), all trained with AdamW and without normalisation. Lines show the mean over 5 seeds and shading ± 1 standard deviation.

18% to 196% across layers for FA and from 39% to 249% for DFA. BP is essentially unaffected. LDFA with rank 4 shows a smaller significant increase (41-59%), and rank 32 is mostly unchanged. BN also significantly increases the trajectory dimensionality of FA and DFA (Figure 14), by up to

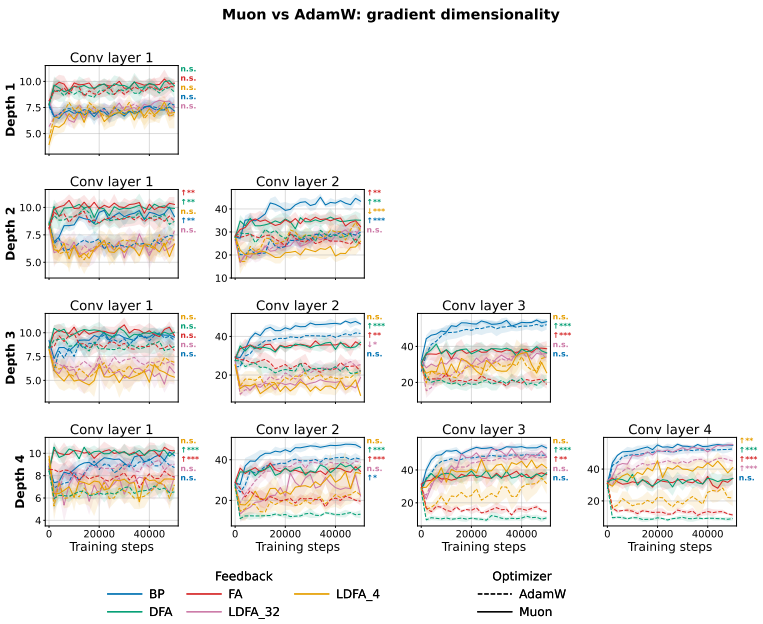


Figure 11: **Muon increases the gradient dimensionality of FA and DFA across layers.** Effective rank of the gradients for each layer (columns) and network depth (rows), for every feedback method trained with Muon (solid) or AdamW (dashed). Lines show the mean over 5 seeds and shading ± 1 standard deviation.

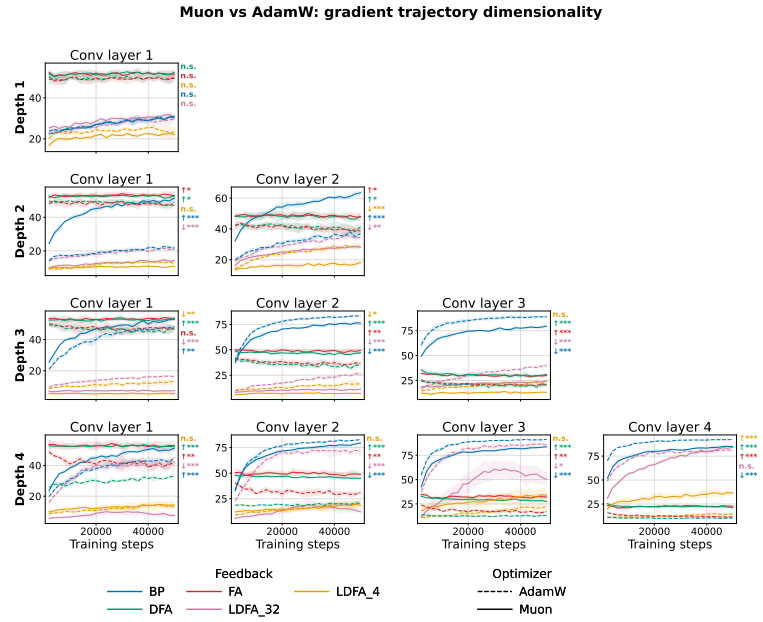


Figure 12: **Muon increases the gradient trajectory dimensionality of FA and DFA** Effective rank of the Gram matrix of the gradient trajectory for each layer (columns) and network depth (rows), for every feedback method trained with Muon (solid) or AdamW (dashed). Lines show the mean over 5 seeds and shading ± 1 standard deviation.

201% and 160%. For BP, it reduces trajectory dimensionality by 18-31%. Like Muon, BN therefore mainly helps the methods whose gradients collapse without it.

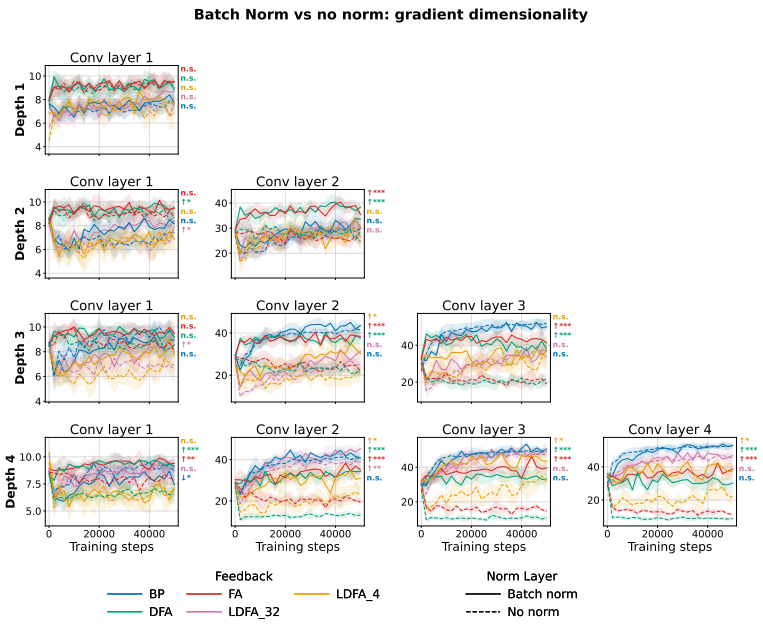


Figure 13: **BN increases the gradient dimensionality of FA and DFA across layers.** Effective rank of the gradients for each layer (columns) and network depth (rows), for every feedback method trained with AdamW, with BN (solid) or without normalisation (dashed). Lines show the mean over 5 seeds and shading ± 1 standard deviation.

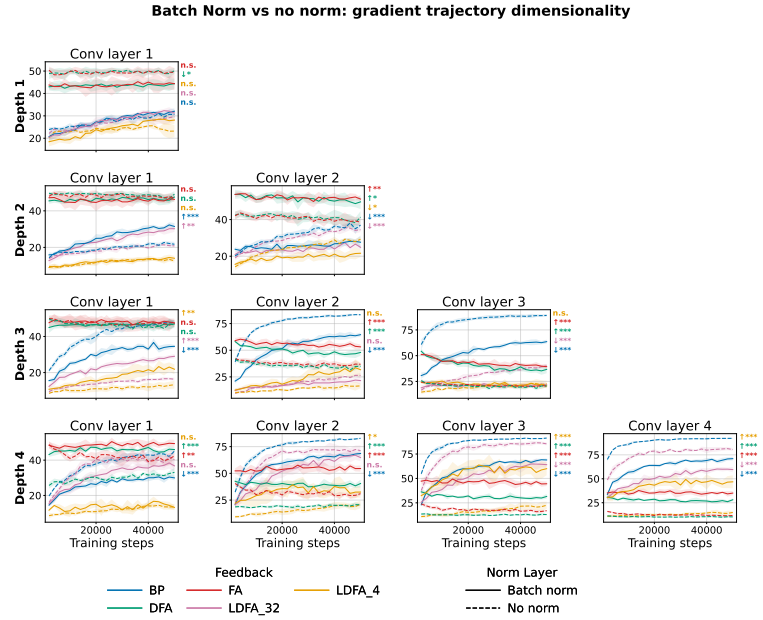


Figure 14: **BN increases the gradient trajectory dimensionality of FA and DFA.** Effective rank of the Gram matrix of the gradient trajectory for each layer (columns) and network depth (rows), for every feedback method trained with AdamW, with BN (solid) or without normalisation (dashed). Lines show the mean over 5 seeds and shading ± 1 standard deviation.

F Reproducibility

For sections 3 and 4, we performed hyperparameter searches with 2 seeds, then selected the best validation hyperparameters and ran the simulations for 5 seeds. We used the cross-entropy loss,

mini-batch size 64 for 50,000 steps, the configured optimiser, while periodically evaluating train/test loss, and gradient-trajectory metrics at fixed intervals with early stopping for convergence or failed runs.

For unbiased results, for every feedback method, we optimised each models' hyperparameter separately using a grid search over network depths (1,2,3,4), normalisation in (None, Batch Normalisation), Muon with learning rates $(10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$ and momentum $(0, 0.9, 0.95)$, and weight decay $(0, 10^{-3})$, and AdamW with learning rates $(10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$ and weight decay $(0, 10^{-3})$.

For section 5, we used the epoch-based training pipeline for vision models on STL-10, CIFAR-100, and ImageNet. STL-10 and CIFAR-100 models were trained for 90 epochs with a batch size of 128; hyperparameters were searched with one seed and the selected configurations were evaluated with five seeds. ImageNet models were trained for 50 epochs with a batch size of 256 and 3 seeds. We used automatic mixed precision with float16 and a cosine-annealing learning-rate schedule.

We selected hyperparameters separately for each dataset, architecture, feedback method (BP, FA, or DFA), and normalisation setting. ImageNet used configuration-specific learning-rate searches, rather than the grid used for the smaller-scale experiments. ImageNet training used random resized crops to 224×224, random horizontal flips, and RandAugment; evaluation used resizing to 256 followed by a 224×224 centre crop. We used the standard training and test splits for STL-10 and CIFAR-100, and the training and validation splits for ImageNet. Table 1 reports the mean and standard deviation across available seeds of each run's best top-1 evaluation accuracy.

Experiments were run in PyTorch [Paszke et al., 2019] on a workstation equipped with two NVIDIA RTX PRO 4000 Blackwell GPUs, with Hydra [Yadan, 2019] used to manage model, optimiser, data, and training hyperparameters across grid-search runs. The resulting accuracy and loss curves were visualised using Matplotlib [Hunter, 2007]. One full run on the CIFAR10 dataset with 50,000 steps took approximately 10 minutes. One full run on STL-10 took approximately 10 minutes.

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: In the abstract and introduction, we summarized our work and explained its contributions, and did not claim anything further.

   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 discussed the limitations in the discussion section.

   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 provide the full set of assumptions and proofs in the appendix.

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 put the information in 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 provided anonymized code in the supplemental material.

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: Yes, we specified all of these 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 used 5 random seeds for each result, except for ImageNet where we used 3 seeds.

   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 provided this information 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 followed strictly the code of ethics.

   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: [N/A]

    Justification: We do not expect any societal impact from this work. It is mainly about understanding alternatives to backpropagation, which is useful for neuroscience, but will not currently advance machine learning capabilities.

    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: Our work does not pose any risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cited appropriately previous work, and code we used.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification: We did not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: Our work did not involve crowdsourcing experiments.

    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 work did not involve crowdsourcing experiments nor 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: Our work did not involve LLMs used in any novel, important or non standard way.

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.