This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Feedback Alignment Meets Low-Rank Manifolds: A Structured Recipe for Local Learning

Arani Roy
Purdue University
roy173@purdue.edu

Marco P. Apolinario
Purdue University
mapolina@purdue.edu

Shristi Das Biswas
Purdue University
sdasbisw@purdue.edu

Kaushik Roy
Purdue University
kaushik@purdue.edu

Abstract

Training deep neural networks (DNNs) with backpropagation (BP) achieves state-of-the-art accuracy but requires global error propagation and full parameterization, leading to substantial memory and computational overhead. Direct Feedback Alignment (DFA) enables local, parallelizable updates with lower memory requirements but is limited by unstructured feedback and poor scalability in deeper architectures, specially convolutional neural networks. To address these limitations, we propose a structured local learning framework that operates directly on low-rank manifolds defined by the Singular Value Decomposition (SVD) of weight matrices. Each layer is trained in its decomposed form, with updates applied to the SVD components using a composite loss that integrates cross-entropy, subspace alignment, and orthogonality regularization. Feedback matrices are constructed to match the SVD structure, ensuring consistent alignment between forward and feedback pathways. Our method reduces the number of trainable parameters relative to the original DFA model, without relying on pruning or post hoc compression. Experiments on CIFAR-10, CIFAR-100, and ImageNet show that our method achieves accuracy comparable to that of BP. Ablation studies confirm the importance of each loss term in the low-rank setting. These results establish local learning on low-rank manifolds as a principled and scalable alternative to full-rank gradient-based training.

1. Introduction

The growing size and complexity of deep neural networks (DNN) have made their memory and computational demands a critical bottleneck, particularly in resource-constrained environments. Backpropagation (BP), today's dominant training method, amplifies these challenges due to its reliance on a global loss objective, the need to store intermediate activations, and the update-locking problem [18], where the layer updates depend on the completion of both forward and backward passes. BP also suffers from the biologically implausible weight transport problem [2, 21], which requires the transport of symmetric weights during the backward pass. These limitations highlight the need for alternative learning paradigms that offer parallelization, memory efficiency and hardware-compatibility.

Feedback Alignment (FA) [17] replaces the symmetric feedback weights with fixed random ones, showing that the forward weights align with them during training. Direct Feedback Alignment (DFA) [21] extends this idea by removing both the weight symmetry requirement and update-locking constraint, transmitting output errors directly to each hidden layer. Unlike BP, DFA does not require the storage of intermediate activations or backward weight transport, significantly reducing memory and computational overhead. These characteristics make DFA particularly attractive for deployment on resource-constrained and neuromorphic hardware platforms [6, 16, 20, 21]. Figure 1 summarizes the core differences in the update flow between BP and DFA. Despite its promise, DFA shows poor scalability to deeper architectures and convolutional networks. While some success has been observed in fully-connected layers [16], DFA fails on complex datasets such as CIFAR-100 and ImageNet without auxiliary strategies like transfer learning [6]. More recent improvements, such as the Direct Kolen–Pollack (DKP) algorithm [28], have improved alignment through learned symmetric feedback, but often at the cost of greater memory or training complexity. A key insight from [23] is that DFA's effectiveness depends on the alignment between forward and feedback weights, an alignment that deteriorates with depth, especially in convolutional networks. Structural mismatches between fixed feedback paths and convolutional kernels further impede performance. Whether this alignment can be improved efficiently in low-rank subspaces remains an open question. ***Could low-rank representations of weight and feedback spaces enable better alignment and extend DFA to convolutional layers? Could structured local updates make DFA scalable to deeper layers?*** We hypothesize that modeling both forward and feedback signals within low-rank

subspaces can increase alignment, improve inference efficiency, and enable scalable local learning in deep architectures, addressing key limitations of DFA.

We propose SVD-Space Alignment (SSA), a novel framework that combines the structural benefits of Singular Value Decomposition (SVD) with the efficiency of DFA. By operating directly on decomposed weight components (U, S, V^T), SSA enables structured updates within low-rank subspaces, aligning forward and feedback pathways to reduce randomness and improve convergence. A custom loss enforces task accuracy, alignment, and orthogonality. Furthermore, dynamic rank-scheduling scheme progressively compresses the model during training, reducing parameters below DFA levels without sacrificing performance. This synergy makes SSA a scalable and resource-efficient solution for training in constrained environments. In this work, we focus on demonstrating how low-rank constraints improve the scalability and stability of local learning. Our local learning recipe offers the following key contributions:

- We propose SVD-Space Alignment (SSA), a novel local learning framework that combines Direct Feedback Alignment (DFA) with Singular Value Decomposition (SVD) to enable structured updates in low-rank subspaces, addressing DFA's random feedback limitations.
- We design a custom loss function combining feedback error, alignment loss, orthogonality regularization, and sparsity, to ensure efficient, convergent updates in the SVD-space.
- We extend SSA to deep convolutional networks, demonstrating its ability to preserve spatial structure and hierarchical information while enabling scalable local learning.
- Empirical results show that SSA achieves classification accuracy on par with BP, converges faster than DFA, and incurs lower inference cost than DFA on benchmarks such as CIFAR-100 and ImageNet.

2. Background

We consider a deep fully connected neural network with N layers. Each layer $i \in \{1, \dots, N\}$ has a weight matrix $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$, a pre-activation vector a_i, and a post-activation output h_i. Let $f_i(\cdot)$ be the elementwise nonlinearity at layer i. The input to the network is $h_0 = X$, and the final output is $\hat{y} = h_N$. The **forward pass** is defined recursively as:

$$\begin{aligned} a_i &= W_i h_{i-1}, \\ h_i &= f_i(a_i), \quad \text{for } i = 1, \dots, N. \end{aligned} \tag{1}$$

The task-specific loss $\mathcal{L}(\hat{y}, y)$ quantifies the mismatch between the network's predictions $\hat{y}$ and the true labels y.

Backward Pass with Backpropagation (BP): In standard BP, gradients are computed via recursive application of the chain rule. The error signal $\delta a_i = \frac{\partial \mathcal{L}}{\partial a_i}$ is propagated backward from the top layer. Neglecting optimizer-specific details and biases, the gradient update for W_i is given by:

$$\delta W_i = -\frac{\partial \mathcal{L}}{\partial W_i} = -\left[\left(W_{i+1}^\top \delta a_{i+1}\right) \circ f_i'(a_i)\right] h_{i-1}^\top, \tag{2}$$

where $\circ$ denotes elementwise (Hadamard) product and $f_i'(a_i)$ is the derivative of the activation function. This recursive dependence on W_{i+1} breaks locality and requires symmetric weights, limiting parallelization and efficiency.

Backward Pass with Direct Feedback Alignment (DFA): DFA replaces the non-local BP gradient signal $W_{i+1}^\top \delta a_{i+1}$ with a random fixed projection of the final-layer error signal δa_N. Each layer i uses a feedback matrix $B_i \in \mathbb{R}^{d_i \times d_N}$, fixed at initialization, to compute its local update:

$$\delta W_i = -\left[\left(B_i \delta a_N\right) \circ f_i'(a_i)\right] h_{i-1}^\top, \tag{3}$$

where $\delta a_N = \frac{\partial \mathcal{L}}{\partial a_N} = \hat{y} - y$ for standard classification losses like mean squared error or cross-entropy. DFA removes upstream weight dependency, enabling local, parallel updates with lower memory and compute overhead.

Alignment in DFA Although DFA uses fixed random feedback matrices, it has been empirically observed that the forward weights gradually adapt to align with the feedback

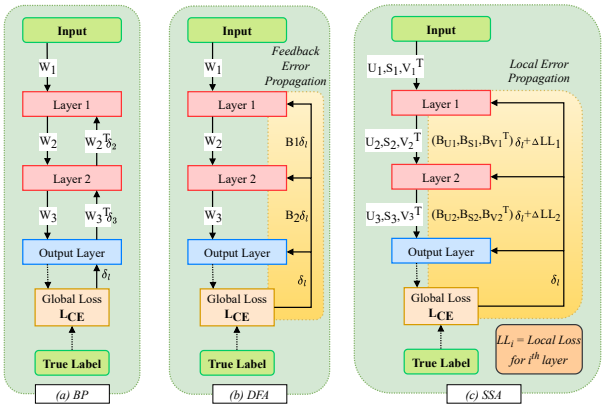


Figure 1. A comparison of neural network training methods. Notations: W = forward weights, $Layer$ = Layer Activations, L_{CE} = cross-entropy loss, B = random feedback weights, δ = gradients, δ_l = gradient of L_{CE} = feedback error, U, S, V^T = SVD components of forward weights, B_U, B_S, B_{V^T} = SVD components of feedback weights, Local Loss (LL) has components: L_{align} = Alignment loss, L_{ortho} = Singular Vector Orthogonality regularizer. **(a) Backpropagation (BP):** Global gradient updates through each layer. **(b) Direct Feedback Alignment (DFA):** Local updates with random feedback **(c) SVD-space Alignment (SSA) (ours):** Decomposes weights into SVD components before training and aligns feedback on those decomposed components itself with local losses. Note: LL + L_{CE} is the total local loss objective.

pathways during training, such that $W_i \approx \lambda B_i^\top$ for some scalar λ. This emergent behavior, termed *weight alignment*, improves the directional agreement between DFA's pseudo-gradients and the true gradients computed via BP. More precisely, alignment refers to the similarity between the pseudo-gradient δa_i^{DFA} and the BP gradient δa_i^{BP}. A common alignment metric is the cosine similarity between the DFA and BP weight gradients:

$$\cos \angle(\nabla_W \mathcal{L}_{\text{BP}}, \nabla_W \mathcal{L}_{\text{DFA}}) = \frac{\langle \nabla_W \mathcal{L}_{\text{BP}}, \nabla_W \mathcal{L}_{\text{DFA}} \rangle}{\|\nabla_W \mathcal{L}_{\text{BP}}\|_F \cdot \|\nabla_W \mathcal{L}_{\text{DFA}}\|_F}. \tag{4}$$

Initially, due to random initialization, this alignment is weak. However, as training progresses, the forward weights rotate toward the subspace defined by the feedback matrices, leading to stronger alignment and more effective updates [23]. This phenomenon provides a key explanation for DFA's empirical success despite bypassing gradient symmetry. Nevertheless, [23] observe that this alignment tends to deteriorate in deeper networks, specially when structural mismatches exist between the fixed feedback paths and the convolutional or deep hierarchical structures of the forward weights. To address this, our method introduces structured forward and feedback paths and SVD-based loss terms that improve alignment and training stability, especially in convolutional and deep architectures.

3. Related Works

Our work builds on prior research in DFA, SVD-based representations in neural networks, and local layerwise training. This section reviews these foundations.

DFA and its variants: DFA [21] offers a lightweight alternative to BP by replacing symmetric feedback with fixed random matrices, enabling local updates without backward weight transport. Despite its efficiency, DFA struggles with poor gradient alignment in deep and convolutional networks, limiting its scalability [24, 28]. Enhancements such as auxiliary classifiers [22] and partial integration into transformer models [16] improve stability but retain BP in key components. To improve alignment, Uniform Sign-concordant Feedbacks (uSF) define $B_i = \text{sign}(W_i^\top)$, while Batchwise Random Magnitude Sign-concordant Feedbacks (brSF) set $B_i = |R_i| \cdot \text{sign}(W_i^\top)$ with resampled $|R_i|$. Our approach constrains our parameter space into low-rank manifolds, and adds custom local losses to the standard DFA errors, to improve alignment and scalability.

SVD in Neural Networks: SVD has been widely used to compress neural networks by reducing the dimensionality of weight matrices, lowering parameter count and FLOPs with minimal accuracy loss [8]. Subsequent work [9, 29] applied SVD to CNNs for inference acceleration, with [29] introducing orthogonality regularization to stabilize training. However, these methods rely on backpropagation, which can disrupt the orthogonality of the decomposed weights. In contrast, SSA integrates SVD with DFA to preserve orthogonality and enables rank reduction during training, yielding compact models without losing accuracy.

Local Layerwise Training: Local learning rules, which update layers independently of global gradients, are gaining attention for their scalability and efficiency. Methods like greedy layerwise training [3], DRTP [10], and AugLocal [19], reduce BP's memory costs using local losses or auxiliary networks. Forward-only methods, such as Hinton's Forward-Forward (FF) [12] and PEPITA [7], avoid backprop entirely but often fall short in accuracy, especially in deeper networks, and struggle with global alignment. Our work focuses on improving local loss methods, as forward-only approaches struggle with classification accuracy in deeper networks. We enhance local layerwise training by applying SVD to layer weights and aligning representations with feedback matrices using DFA.

4. Methodology

We now present the details of SVD-Space Alignment (SSA), a local learning method that updates network weights in their SVD-decomposed form. Each layer's orthogonal and singular components are optimized using a custom loss combining feedback error, alignment, and orthogonality regularization. Gradients are projected onto the tangent spaces of the Stiefel manifolds during training. The following subsections outline the structured forward and backward paths, loss formulation, SSA update rules, and dynamic rank scheduling. A theoretical perspective on our method is provided in Section 5.

4.1. Embedding Structural Priors: Breaking Weights into SVD-Space

Given a weight matrix $W_i \in \mathbb{R}^{m \times n}$ for layer i, we decompose it into its SVD form:

$$W_i = U_i S_i V_i^T \tag{5}$$

where $U_i \in \mathbb{R}^{m \times r}$ and $V_i \in \mathbb{R}^{n \times r}$ are orthogonal matrices, and $S_i \in \mathbb{R}^{r \times r}$ is a diagonal matrix of singular values, with rank $r \leq \min(m, n)$. Decomposing weights prior to training allows updates to be applied directly in the SVD-space, preserving the orthogonality of U_i and V_i^T throughout the training process. This approach avoids the computational overhead associated with performing SVD at every epoch, while facilitating efficient rank reduction during training.

SVD-Space for Convolutional Layers: To adapt the SVD-space approach for convolutional layers, we employ a spatial decomposition strategy inspired by [29]. A convolutional kernel $K \in \mathbb{R}^{N \times C \times H \times W}$, with N as the number of filters, C as the number of input channels, and $H \times W$ as the spatial dimensions, is first reshaped into a 2D matrix $K' \in \mathbb{R}^{NW \times CH}$. K' is then decomposed using SVD:

$K' = U\Sigma V^T$ where $U \in \mathbb{R}^{NW\times r}$ and $V \in \mathbb{R}^{CH\times r}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix of singular values, with rank $r = \min(NW, CH)$. The decomposed components are reshaped into convolutional layers as follows: $U\sqrt{\Sigma}$ is reshaped into a convolutional kernel $K_1 \in \mathbb{R}^{r\times C\times H\times 1}$, $\sqrt{\Sigma}V^T$ is reshaped into a kernel $K_2 \in \mathbb{R}^{N\times r\times 1\times W}$. This decomposition effectively factorizes the original convolution into two consecutive operations. The first convolution, with K_1, reduces the input channels and captures spatial information across the height dimension, while the second convolution, with K_2, restores the original output dimensionality and captures spatial information across the width dimension. This hierarchical decomposition effectively preserves spatial and hierarchical features. We retain the decomposed kernels for inference. For SSA, we form our feedback matrices to be of the same dimensions as our convolutional kernels.

4.2. Local Loss Objective Design

The local loss function at layer i is designed to ensure efficient and structured learning directly in the SVD-space. It consists of three key components:

$$LL_i(\theta_i) = \alpha L_{\text{CE}}(\theta_i) + \beta L_{\text{align}}(\theta_i) + \gamma L_{\text{ortho}}(\theta_i) \tag{6}$$

where $\theta_i = (U_i, S_i, V_i^T)$, and each term contributes uniquely to the training process. We provide the gradients of this loss to the layer as shown in Fig. 1 (c).

Cross-Entropy Loss (L_{CE}): This term ensures the model learns the primary task like classification. In the DFA framework, the propagated feedback error is derived directly from the gradient of the model cross-entropy loss:

$$\Delta L_{\text{CE}} = y_{\text{predict}} - y_{\text{label}} \tag{7}$$

Here, ΔL_{CE} *i.e.* δ_l guides the local updates in DFA and aligns predictions with the target outputs.

Alignment Loss (L_{align}): This term ensures alignment between the forward SVD components (U_i, S_i, V_i^T) and the feedback matrices $(B_{U_i}, B_{S_i}, B_{V_i^T})$ used for DFA updates, reducing mismatch in SVD-space. The loss is defined as:

$$L_{\text{align}}(\theta_i) = \|U_i - B_{U_i}\|_F^2 + \|S_i - B_{S_i}\|_F^2 + \|V_i^T - B_{V_i^T}\|_F^2 \tag{8}$$

Singular Vector Orthogonality Regularizer (L_{ortho}): To ensure the numerical stability of the SVD decomposition and a well-structured representation during training, this regularizer promotes orthogonality in the singular vectors U_i and V_i^T. It is formulated as:

$$L_{\text{ortho}}(\theta_i) = \|U_i^T U_i - I\|_F^2 + \|V_i^T V_i - I\|_F^2 \tag{9}$$

4.3. SSA Updates in SVD-Space

In SSA, DFA is adapted to operate directly in the SVD-space of the weight matrices (5). For each layer i, we first sample a fixed random feedback matrix $B_i \in \mathbb{R}^{d_i\times d_N}$ from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, followed by column-wise normalization to unit ℓ_2 norm. To impose structure, we compute a truncated randomized SVD of B_i: $B_i \approx B_{U_i} B_{S_i} B_{V_i}^\top$, where the retained rank r is chosen to match the forward decomposition. These structured factors $(B_{U_i}, B_{S_i}, B_{V_i})$ serve as explicit alignment targets in the local loss objective. The update rules for each component are as follows:

$$\begin{aligned} U_i^{(t+1)} &= U_i^{(t)} - \eta \nabla_{U_i} LL_i(U_i, S_i, V_i^T) \\ S_i^{(t+1)} &= S_i^{(t)} - \eta \nabla_{S_i} LL_i(U_i, S_i, V_i^T) \\ V_i^{T(t+1)} &= V_i^{T(t)} - \eta \nabla_{V_i^T} LL_i(U_i, S_i, V_i^T) \end{aligned} \tag{10}$$

where η is the learning rate, and $LL_i(U_i, S_i, V_i^T)$ is the local loss for layer i. These updates are fully local and parallelizable, requiring no global error propagation. The gradients for each SVD component are derived from the local loss LL_i and are detailed below.

Singular Values S_i:

$$\nabla_{S_i} LL_i = U_i^\top \nabla_{W_i} L_{\text{CE}} V_i + 2\beta(S_i - B_{S_i}) \tag{11}$$

Left Singular Vectors U_i:

$$\begin{aligned} \nabla_{U_i} LL_i = (I - U_i U_i^\top)\big[&\nabla_{W_i} L_{\text{CE}} V_i S_i^{-1} \\ &+ 2\beta(U_i - B_{U_i}) \\ &+ 4\gamma U_i (U_i^\top U_i - I)\big] \end{aligned} \tag{12}$$

Right Singular Vectors $V_i^\top$:

$$\begin{aligned} \nabla_{V_i^\top} LL_i = (I - V_i^\top V_i)\big[&\nabla_{W_i} L_{\text{CE}}^\top U_i S_i^{-1} \\ &+ 2\beta(V_i^\top - B_{V_i^\top}) \\ &+ 4\gamma V_i^\top (V_i V_i^\top - I)\big] \end{aligned} \tag{13}$$

The projection terms $(I - U_i U_i^\top)$ and $(I - V_i^\top V_i)$ ensure updates remain on the Stiefel manifold (details in 5.2), preserving orthogonality during optimization.

4.4. Dynamic Rank Reduction Strategy

SVD-based parameterization increases the model size from mn to approximately $r(m + n)$, effectively doubling the parameter count when $r = \min(m, n)$. To ensure SSA remains compact and comparable to the original model, we employ a hybrid rank reduction strategy, where the effective rank r is constrained to satisfy: $r < \frac{mn}{m+n}$ Training begins with full rank r_0. Over the first 30% of epochs, we apply an **epoch-based schedule** that reduces r to $0.7r_0$. This is guided by the Hoyer regularizer, applied every 10 epochs to promote sparsity in singular values: $\mathcal{L}_{\text{Hoyer}}(S_i) = \frac{\|S_i\|_1}{\|S_i\|_2}$. In later epochs, we adopt a **threshold-based schedule** that retains the top singular values accounting for 95% of the

spectral energy, preventing over-pruning near convergence. While the epoch-based schedule enables rapid early compression, reducing the rank below 30% of r_0 often destabilizes training. Our hybrid strategy balances early aggressive pruning with stable, energy-aware refinement. Notably, SSA trains predominantly and always infers at reduced rank, demonstrating its efficiency in both convergence and deployment.

5. Theoretical Foundations and Alignment Properties in SSA

In this section, we examine the theoretical foundations and alignment characteristics of our method. We begin by analyzing the properties of the local loss, followed by a detailed explanation of the update procedures on the Stiefel manifold. We then discuss the relationship between SSA and BP gradients, providing evidence that SSA achieves stronger gradient alignment than DFA. Finally, we study the dynamics of the matrix alignment angle across training epochs and network depth, illustrating how SSA maintains more stable alignment in deeper architectures.

5.1. SSA Local Loss Properties

Convexity and smoothness of the loss function are essential for stable and efficient optimization. The cross-entropy loss L_{CE} is convex and L-smooth with Lipschitz constant $L = 0.5$, while the alignment loss L_{align}, composed of squared Frobenius norms, is also convex and 2-smooth [5, 26]. In contrast, the orthogonality regularizer L_{ortho}, which involves quartic terms such as $\|U^\top U - I\|_F^2$, is non-convex. To address this, we constrain the updates of U and V to lie on the Stiefel manifold [25], projecting gradients onto its tangent space to preserve orthogonality and improve optimization stability. While the overall objective is non-convex due to these constraints, the dominance of convex and smooth terms in the local layerwise losses results in empirically stable and efficient convergence. A detailed analysis of convexity and smoothness properties is provided in Supplementary Section 1. Although a global convergence guarantee is difficult in this setting, we observe that maintaining local Lipschitz continuity and smoothness strongly correlates with stable training and improved final performance in practice.

5.2. Orthogonality-Preserving Updates

The Stiefel manifold $\mathbb{V}_r(\mathbb{R}^n)$ comprises all $n \times r$ matrices with orthonormal columns, i.e., $U^\top U = I$. To preserve this structure in the singular matrices U_i and V_i, SSA employs a two-fold strategy: (1) projection-based gradient updates and (2) soft orthogonality regularization. We project the gradients onto the tangent space of the Stiefel manifold as:

$$\nabla_{U_i}\mathcal{L} \leftarrow (I - U_i U_i^\top)\nabla_{U_i}\mathcal{L}, \quad \nabla_{V_i^T}\mathcal{L} \leftarrow (I - V_i^T V_i)\nabla_{V_i^T}\mathcal{L}.$$

This lightweight operation avoids costly retractions (QR or Cayley transforms), while preserving local orthogonality constraints. It aligns with recent findings [1, 25] that such projection-only methods can match full Riemannian optimizers in convergence and constraint satisfaction. To mitigate long-term drift due to stochastic updates or numerical imprecision, we additionally include a soft penalty $\|U_i^\top U_i - I\|_F^2$ and $\|V_i V_i^\top - I\|_F^2$ in the loss function. This complementary design enables stable, structure-aware optimization in the SVD space throughout training.

5.3. Alignment Between SSA and BP Gradients

As per [21] and [4], alignment refers to the degree of similarity between BP gradient and FA techniques based pseudo-gradients. High alignment (alignment angle $> 0°$) suggests a better approximation of the true direction of descent of the gradient. To demonstrate that SSA gradients align with BP gradients, we analyze the cosine similarity between the true gradient $\nabla_W L_{\text{true}}$ and the SSA gradient $\nabla_W L_{\text{SSA}}$. Expanding $\nabla_W L_{\text{SSA}}$:

$$\nabla_W L_{\text{SSA}} = \nabla_W L_{\text{CE}} + 2\beta\nabla_W L_{\text{align}} + 4\gamma\nabla_W L_{\text{ortho}} \quad (14)$$

We analyze the contributions of each term:

Contribution from L_{CE}: The primary descent direction is dictated by $\nabla_W L_{\text{CE}}$ (original DFA feedback error), which matches $\nabla_W L_{\text{true}}$ as shown in DFA prior works [21].

Contribution from L_{align}: The gradient of the alignment loss is:

$$\nabla_W L_{\text{align}} = 2(W - B) \quad (15)$$

Since B approximates W^T for the backward gradient pathway, this term complements $\nabla_W L_{\text{CE}}$, ensuring positive alignment. This assumes that W and B are reasonably aligned over time, and the difference between them is small, given proper initialization of B.

Contribution from L_{ortho}: The orthogonality regularizer ensures stability by constraining the singular vectors to the Stiefel manifold. The gradient of L_{ortho} is tangent to this manifold, so:

$$\langle \nabla_W L_{\text{true}}, \nabla_W L_{\text{ortho}} \rangle \approx 0 \quad (16)$$

Also, the orthogonality-preserving projections contribute to stability without altering descent direction.

The inner product between $\nabla_W L_{\text{true}}$ and $\nabla_W L_{\text{SSA}}$ is:

$$\langle \nabla_W L_{\text{true}}, \nabla_W L_{\text{SSA}} \rangle = \|\nabla_W L_{\text{true}}\|^2 + 2\beta\langle \nabla_W L_{\text{true}}, W - B \rangle \quad (17)$$

This is guaranteed to be positive because $\|\nabla_W L_{\text{true}}\|^2 > 0$ and $\langle \nabla_W L_{\text{true}}, W - B \rangle > 0$.

Thus:

$$\cos(\nabla_W L_{\text{true}}, \nabla_W L_{\text{SSA}}) > 0 \quad (18)$$

ensuring the angle between the gradients is $\leq 90°$. This alignment guarantees that SSA updates descend along a similar direction to the true gradient of BP.

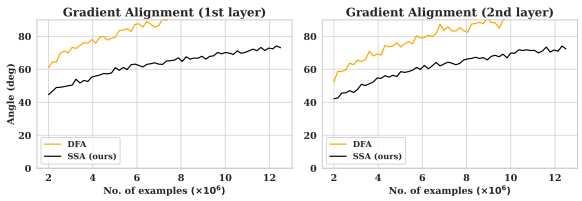


Figure 2. Gradient alignment (angle in degrees) between SSA/DFA and BP gradients across epochs on a 3-layer MLP. Lower angles indicate better alignment.

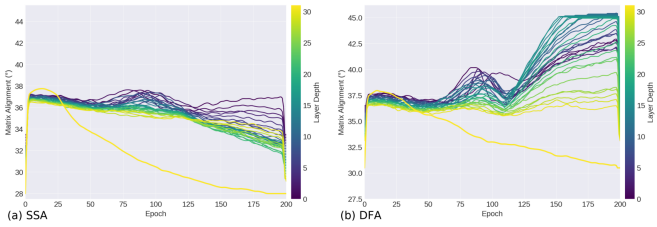


Figure 3. Matrix alignment angles across ResNet-32 layers over training epochs, between (a) SSA (b) DFA, and BP

5.4. Gradient Alignment Analysis

To assess whether SSA produces updates more aligned with the true supervised gradient than DFA, we compute the alignment angle (in degrees) between each method's update and the corresponding backpropagation (BP) gradient. We run a 3-layer MLP on CIFAR-10, and log alignment angles for each layer for different sample sizes. We compare against DFA to quantify how incorporating structure and Stiefel-regularized updates in SSA affects alignment. As shown in Figure 2, SSA consistently achieves lower alignment angles across layers compared to DFA, indicating that SSA updates are better aligned with the true BP direction.

5.5. Layerwise Alignment Analysis over Time

Additionally, we evaluate the *matrix alignment angle*, defined as the geometric angle between the column spaces of $W^\top$ and B, with smaller angles indicating stronger alignment. Tracking this metric across training epochs and network depth in figure 3 reveals how feedback alignment evolves during learning. As expected, the *last layer closest to the output* achieves the best alignment, since error information is directly available at the output. In deeper layers, the alignment varies more gradually across epochs, reflecting the increasing complexity of propagating structured signals through the network. Crucially, SSA outperforms DFA by maintaining lower alignment angles in deeper layers, yielding greater stability and reduced error. This highlights SSA as a principled and more reliable alternative to DFA in deep networks.

6. Experimental Setup

We evaluate the performance of our framework by outlining the datasets, architectures, baselines, and evaluation metrics for computer vision tasks.

Datasets. We conduct experiments on CIFAR-10, CIFAR-100 [14], and ImageNet (ILSVRC-2012) [15], using CIFAR for small-scale benchmarks and ImageNet for large-scale evaluation.

Network Architectures. SSA is evaluated on SmallConv (conv96-pool-conv192-pool-conv512-pool-fc1024), VGG-13 [27], and ResNet-32 [11], covering diverse depths and sizes to assess flexibility and scalability.

Baselines. We compare SSA against BP, DFA [21], SVD-BP (low-rank BP without feedback alignment), PredSim [22], AugLocal [19], DRTP [10], and PEPITA [7], evaluating accuracy, memory, computational cost, and training stability.

Evaluation Metrics. We measure the following. **Classification Accuracy:** Top-1 and top-5 accuracy for generalization. **Memory Usage:** Training memory footprint for efficiency. **Computational Cost:** FLOPs per training iteration to quantify efficiency gains.

Training Details. Experiments are conducted on NVIDIA A40 GPUs with 48 GB memory, implemented in PyTorch. Training uses the Adam optimizer [13] with a learning rate between $1e{-}4$ and $5e{-}4$. Data augmentation includes random cropping and horizontal flipping. Batch sizes are 128 for CIFAR and 256 for ImageNet. SSA starts with full-rank weight matrices and progressively reduces rank during training as described in Section 4.4.

7. Results and Analysis

In this section, we present comprehensive experimental results to evaluate the effectiveness of the proposed SSA method. We benchmark SSA against baseline methods, including standard DFA and its recent variants, across multiple metrics. An ablation study is conducted to assess the contribution of individual loss components. Beyond accuracy and convergence behavior, we also quantify the memory and computational gains enabled by SSA's structured learning and low-rank optimization.

7.1. Classification Accuracy

We evaluate the classification accuracy of SSA on CIFAR-10 and ImageNet and compare it against a range of baselines. Table 1 reports results for SmallConv, including BP, SVD-BP, and several local or forward-only methods. Here, SVD-BP trains the SVD components using backpropagation, DRTP projects the targets (rather than the global error) to local layers, and PEPITA trains layers via two forward passes. The Forward-Forward method is excluded as it does not extend to convolutional networks. SSA outperforms the

other local and forward-only methods while reaching accuracy comparable to BP.

Method	CIFAR-10 (mean ± std)	CIFAR-100 (mean ± std)
BP	87.57 ± 0.14	62.25 ± 0.21
SVD-BP [29]	87.30 ± 0.18	61.64 ± 0.19
DFA [2]	73.10 ± 0.53	44.93 ± 0.51
DRTP [10]	68.96 ± 0.80	NA
PEPITA [7]	56.34 ± 1.24	27.56 ± 0.67
SSA (ours)	86.21 ± 0.10	60.72 ± 0.15

Table 1. Comparison of classification accuracy (mean ± standard deviation) over 5 independent runs with random inits for CIFAR-10 and CIFAR-100 datasets, with SmallConv network.

Table 2 extends the comparison to larger networks (VGG-13 and ResNet-32) on CIFAR-10 and ImageNet. On VGG-13, SSA matches BP within 0.2% on CIFAR-10 and remains competitive on ImageNet, showing that SSA scales effectively where DFA and other local methods from Table 1 collapse. We include DFA here to highlight this scalability gap, while omitting other baselines that fail entirely at this scale. On ResNet-32, SSA exhibits a larger gap to BP, reflecting a common limitation of local learning: feedback signals weaken as they traverse many layers without gradient information, and residual connections further distort them. Nevertheless, SSA still achieves substantially higher accuracy than DFA, demonstrating that structure-aware learning mitigates, though does not completely resolve, these challenges. For completeness, we note that PredSim combines prediction and similarity matching losses but does not report ImageNet results, while AugLocal attains strong accuracy by embedding later-layer information into earlier layers at the cost of substantial auxiliary overhead. In contrast, SSA achieves comparable accuracy without such additions, underscoring its efficiency and scalability.

7.2. Ablation Study

We perform an ablation study to quantify the contribution of each component in the composite loss function to the overall performance of SSA. Table 3 reports the impact of removing each term: Cross-Entropy Loss, Alignment Loss, and the Orthogonality Regularizer, on classification accuracy and computational efficiency. Additional results, including a sensitivity analysis of loss coefficients α, β, γ, are provided in the supplementary material.

From Table 3, removing the Cross-Entropy Loss results in a significant accuracy drop, as expected. Alignment loss attempts to reduce the loss between the subspaces in forward and feedback weights. Alignment loss aids gradient direction preservation, and therefore, removing these components also decreases accuracy, but less severely than Cross-Entropy Loss. Orthogonality Regularizer maintains the unitary properties of U, Vt. If the unitary properties are maintained, the angular alignment and any angular transformation will be meaningful (preserving lengths and angles). Hence, removing the regularizer negatively impacts both accuracy and computational efficiency, which might worsen in deeper models. Overall, the ablation study demonstrates that each loss component is essential for SSA's performance and efficiency.

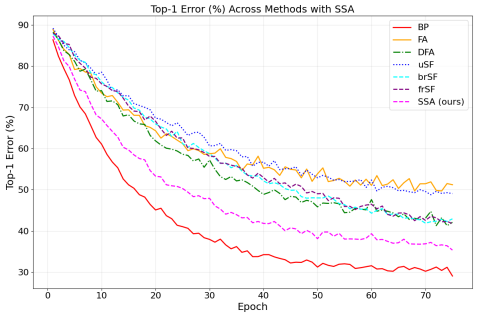


Figure 4. Top-1 Error (%) Across Epochs for a 3-layer MLP.

7.3. Comparison with DFA and its variants

SSA introduces two key distinctions from DFA: the use of a structured weight-space and custom loss components applied directly in this SVD-space. To evaluate these differences, we present comparisons between SSA and DFA, along with its variants [24], in the following tables and figures. The variants are explained in Section 3.

We evaluate SSA and DFA (with its variants) on CIFAR-10 and ImageNet datasets. Tables 4 summarize the test error rates for these methods, including the baseline BP. Our results show that SSA outperforms most variants of DFA across both datasets. To further analyze convergence behavior, we plot the error across epochs for a 3-layer MLP trained with SSA, BP, and DFA (with its variants) in Figure 4. The results illustrate that SSA converges significantly faster than DFA and its variants, showcasing the advantages of structured feedback and custom loss design. While DFA and its variants perform poorly on convolutional layers, or cannot be directly applied to them, SSA achieves robust performance across both fully connected and convolutional architectures. To ensure uniformity in comparisons, the plotted results focus on MLPs. This limitation of DFA on convolutional layers further highlights the versatility of SSA for broader network types. Note, we also achieve alignment angles between SSA and BP gradients better than that of DFA gradients, as seen in Section 5.4.

7.4. Memory and Compute Cost Analysis

SSA performs inference using the decomposed form $W_i = U_i S_i V_i^\top$, avoiding reconstruction of full weight matrices. The forward computation involves: $\mathcal{O}(r \cdot n)$ for $V^\top x$, $\mathcal{O}(r)$ for scaling by S, $\mathcal{O}(m \cdot r)$ for projection by U, leading to an overall inference complexity of $\mathcal{O}(r(n+m+1))$,

Network	Method	CIFAR10 (Top-1, mean ± std)	ImageNet (Top-1, mean ± std)	ImageNet (Top-5, mean ± std)
VGG-13	BP	93.75 ± 0.12	71.59 ± 0.25	90.39 ± 0.14
	SVD-BP [29]	92.80 ± 0.15	71.37 ± 0.20	90.20 ± 0.18
	PredSim [22]	86.49 ± 0.53	NA	NA
	AugLocal [19]	93.72 ± 0.10	70.93 ± 0.22	90.16 ± 0.16
	SSA (ours)	92.70 ± 0.13	69.68 ± 0.24	88.84 ± 0.15
ResNet-32	BP	92.14 ± 0.11	74.28 ± 0.30	91.76 ± 0.12
	SVD-BP [29]	91.77 ± 0.14	72.91 ± 0.27	89.27 ± 0.20
	PredSim [22]	79.31 ± 0.45	NA	NA
	AugLocal [19]	93.47 ± 0.12	73.95 ± 0.25	91.70 ± 0.17
	SSA (ours)	88.02 ± 0.18	69.38 ± 0.23	87.72 ± 0.19

Table 2. Comparison of classification accuracy (mean ± standard deviation) over 5 independent runs for CIFAR-10 and ImageNet datasets.

Table 3. Ablation study on CIFAR-10 showing the impact of each component in the composite loss function.

Component Removed	Accuracy (%)	FLOPs (B)
Full SSA (All Components)	92.70	0.14
No Cross-Entropy Loss	27.05	0.133
No Alignment Loss	83.12	0.119
No Orthogonality Regularizer	85.44	0.112

Method	LeNet	ResNet-20	ResNet-56	ResNet-18 (I)
BP	15.92	10.01	7.83	30.39
FA	40.67	29.59	29.23	85.25
DFA	37.59	32.16	32.02	82.45
uSF	16.34	10.59	9.19	34.97
brSF	17.08	11.08	10.13	37.21
SSA (Ours)	16.27	10.63	9.72	33.41

Table 4. CIFAR-10 test error (%) for LeNet, ResNet-20, and ResNet-56, and ImageNet Top-1 error (%) for ResNet-18 (I). SSA consistently outperforms DFA and FA, narrowing the gap to BP.

where $r \ll \min(m, n)$. During inference, SSA stores only the compact low-rank factors: $\mathcal{O}(m \cdot r) + \mathcal{O}(r) + \mathcal{O}(r \cdot n)$, which is significantly smaller than storing the full dense weight matrix of size $\mathcal{O}(m \cdot n)$. The SVD decomposition itself is performed only once prior to training, with complexity $\mathcal{O}(mn \min(m, n))$, and this cost is negligible relative to the total training time dominated by repeated forward and backward passes.

Empirical Comparison. Figure 5 compares compute and memory costs of SSA, DFA, and BP on LeNet, SmallConv, and VGG-13. DFA is efficient on shallow models but fails to scale, whereas SSA sustains efficiency on deeper networks and achieves lower inference-time cost through structure-aware low-rank training.

8. Conclusion

We introduced *SVD-Space Alignment* (SSA), a novel local learning framework that combines Singular Value Decomposition (SVD) with Direct Feedback Alignment (DFA)

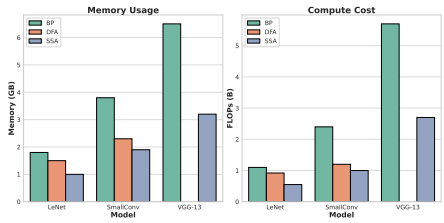


Figure 5. Compute and memory cost comparison of BP, DFA, and SSA across different architectures. SSA achieves lowest inference-time cost through structured low-rank training.

to enable efficient, scalable training of deep neural networks. Unlike prior DFA-based methods, which failed to scale to convolutional or deep architectures, SSA incorporates structure into both the forward and feedback pathways, making local learning viable even for models like ResNet. By performing updates directly in the SVD-space, SSA utilizes a structured local loss function that ensures gradient alignment, stability through orthogonality preservation, and efficient convergence. The projection of gradients onto the Stiefel manifold complements the orthogonality regularizer, maintaining the geometry of the SVD-space while enhancing training robustness. Our theoretical analysis demonstrated that SSA gradients align positively with backpropagation gradients, guaranteeing descent directions, while dynamic rank reduction enables lightweight inference in low-rank subspaces. Experiments show that SSA achieves competitive accuracy on par with backpropagation, with substantial reductions in memory and compute. Ablation studies underscore the importance of each loss component in balancing accuracy and efficiency. In summary, SSA offers a principled and scalable local learning method, paving the way for resource-aware neural network training in real-world applications.

References

[1] Pierre Ablin and Gabriel Peyré. Fast and accurate optimization on the orthogonal manifold without retraction. In *International Conference on Artificial Intelligence and Statistics*, pages 5636–5657. PMLR, 2022.

[2] 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.

[3] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

[4] Blake Bordelon and Cengiz Pehlevan. The influence of learning rule on representation dynamics in wide neural networks. *arXiv preprint arXiv:2210.02157*, 2022.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] Brian Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct feedback alignment with sparse connections for local learning. *Frontiers in neuroscience*, 13: 525, 2019.

[7] Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pages 4937–4955. PMLR, 2022.

[8] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

[9] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[10] 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.

[11] 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.

[12] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[16] Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.

[17] 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.

[18] 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.

[19] Chenxiang Ma, Jibin Wu, Chenyang Si, and Kay Chen Tan. Scaling supervised local learning with augmented auxiliary networks. *arXiv preprint arXiv:2402.17318*, 2024.

[20] Emre O Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience*, 11:324, 2017.

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

[22] Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pages 4839–4850. PMLR, 2019.

[23] 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.

[24] Albert Jiménez Sanfiz and Mohamed Akrout. Benchmarking the accuracy and robustness of feedback alignment algorithms. *arXiv preprint arXiv:2108.13446*, 2021.

[25] Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35:20051–20063, 2022.

[26] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

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

[28] Matthew Bailey Webster, Jonghyun Choi, et al. Learning the connections in direct feedback alignment. 2021.

[29] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 678–679, 2020.