Supplementary Material

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

1. Local Loss Properties: Convexity and Smoothness

The convergence of gradient descent for Lipschitz-continuous functions is well-established. Given that the composite loss function $L_i(\theta_i)$ satisfies Lipschitz smoothness, choosing an appropriate learning rate η ensures convergence to a stationary point:

$$\lim_{t\to\infty} \|\nabla_{\theta_i} L_i(\theta_i^{(t)})\| = 0. \tag{1}$$

This guarantees stable updates for the SVD components.

1.1. Cross Entropy Loss: Convexity and Smoothness

Gradient and Hessian The gradient of the cross-entropy loss with respect to logits z is:

$$\nabla_z L_{\text{CE}} = \hat{y} - y, \tag{2}$$

where $\hat{y}$ is the softmax output. This serves as the **local feedback error** at each layer.

The Hessian matrix is given by:

$$H_{ij} = \hat{y}_i(\delta_{ij} - \hat{y}_j), \tag{3}$$

which is **positive semi-definite**, ensuring convexity of L_{CE}.

Smoothness and Lipschitz Continuity A function is L-smooth if its gradient is Lipschitz continuous:

$$\|\nabla L_{\text{CE}}(z_1) - \nabla L_{\text{CE}}(z_2)\| \leq L\|z_1 - z_2\|. \tag{4}$$

Since the softmax function is $\frac{1}{2}$-Lipschitz, it follows that the cross-entropy loss is smooth with $L = 1/2$, ensuring stable optimization.

1.2. Alignment Loss: Convexity and Smoothness

The alignment loss is defined as:

$$L_{\text{align}}(\theta_i) = \|U_i - B_{U_i}\|_F^2 + \|S_i - B_{S_i}\|_F^2 + \|V_i^T - B_{V_i^T}\|_F^2 \tag{5}$$

where U, S, V^T are the SVD components, and B_U, B_S, B_{V^T} are feedback matrices.

Gradient and Hessian of Alignment Loss Each term follows the form:

$$\|X - B_X\|_F^2 = \sum_{i,j}(X_{ij} - (B_X)_{ij})^2. \tag{6}$$

The gradient with respect to X is:

$$\nabla_X L_{\text{align}} = 2(X - B_X). \tag{7}$$

The Hessian is:

$$H_{ij} = 2\delta_{ij}, \tag{8}$$

which is diagonal and positive semi-definite, ensuring convexity.

Smoothness and Lipschitz Continuity A function is L-smooth if:

$$\|\nabla L_{\text{align}}(X_1) - \nabla L_{\text{align}}(X_2)\|_F \leq L\|X_1 - X_2\|_F. \tag{9}$$

Since:

$$\|\nabla_X L_{\text{align}}(X_1) - \nabla_X L_{\text{align}}(X_2)\|_F = 2\|X_1 - X_2\|_F, \tag{10}$$

L_{align} is L-smooth with $L = 2$.

Boundedness of Alignment Loss If the norms of U, S, V^T and their feedback matrices satisfy $\|U\|_F, \|S\|_F, \|V^T\|_F \leq M$, then:

$$L_{\text{align}}(U, S, V^T, B_U, B_S, B_{V^T}) \leq 3M^2. \tag{11}$$

1.3. Singular Vector Orthogonality Regularizer: Convexity and Smoothness

The singular vector orthogonality regularizer ensures that singular vectors remain orthogonal:

$$L_{\text{ortho}}(U) = \|U^T U - I\|_F^2, \tag{12}$$

where U is the singular vector matrix and I enforces orthogonality.

Gradient and Hessian of L_{ortho} Expanding the Frobenius norm:

$$L_{\text{ortho}}(U) = \text{Tr}(U^T U U^T U - 2U^T U + I). \tag{13}$$

The gradient is:

$$\nabla_U L_{\text{ortho}}(U) = 4U(U^T U - I). \tag{14}$$

The Hessian involves $U(U^T U)$, making it non-trivial and potentially non-positive semi-definite, implying non-convexity.

Smoothness and Lipschitz Continuity For Lipschitz smoothness:

$$\|\nabla_U L_{\text{ortho}}(U_1) - \nabla_U L_{\text{ortho}}(U_2)\|_F \le L\|U_1 - U_2\|_F. \tag{15}$$

Using:

$$\nabla_U L_{\text{ortho}}(U) = 4U(U^T U - I), \tag{16}$$

we obtain a Lipschitz bound:

$$L \le 4(\|U_1\|_F \|U_1^T U_1 - I\|_F + \|U_2\|_F \|U_2^T U_2 - I\|_F). \tag{17}$$

Boundedness and Regularization. If $\|U\|_F \le M$, then:

$$L_{\text{ortho}}(U) = \|U^\top U - I\|_F^2 \le (M^2 - 1)^2, \tag{18}$$

but this bound holds only if $U^\top U \approx I$. Without such structure, the regularizer can grow unbounded.

Improving Smoothness and Stability. To improve smoothness and numerical stability, we augment the orthogonality regularizer with a Frobenius norm penalty:

$$L'_{\text{ortho}}(U) = \|U^\top U - I\|_F^2 + \lambda\|U\|_F^2. \tag{19}$$

Additionally, constraining U to the Stiefel manifold (e.g., via projection or normalization such that $\|U\|_F = 1$) enhances regularity and stabilizes training in practice.

2. Experimental Details

Training Setup Based on observations from [1], biologically plausible methods like DFA perform better with the Adam optimizer. Therefore, all experiments use Adam with initial learning rates adapted from prior works. Learning rates for SSA are dynamically adjusted to accommodate progressive rank reduction.

CIFAR-100 Experiments We use the CIFAR-100 dataset, containing 60,000 images across 100 classes, with 50,000 for training and 10,000 for testing. Images are resized to **32×32 pixels**, and standard data augmentation techniques, including random cropping (with 4-pixel padding) and horizontal flipping, are applied. Training is conducted over **200 epochs** using Adam with an initial learning rate of 1×10^{-4}. A learning rate scheduler reduces the rate by a factor of 10 at the **20th**, **40th**, and **60th epochs**. A batch size of 128 is used, with He initialization for convolutional layers and Xavier initialization for fully connected layers.

For SSA, weight matrices start at full rank and are progressively reduced every 10 epochs while retaining **95% matrix energy**. Loss coefficients are fixed at $(\alpha, \beta, \gamma) = (1, 0.01, 0.1)$, determined through cross-validation.

ImageNet Experiments We evaluate on ImageNet (ILSVRC-2012), a large-scale dataset with 1.28 million training images and 50,000 validation images across 1,000 classes. Images are resized to **224×224 pixels**, and data augmentation includes random cropping, horizontal flipping, and color jittering. Images are normalized using the dataset mean and standard deviation. Training is conducted for **200 epochs** with Adam, using an initial learning rate of 2×10^{-4} for BP and 5×10^{-4} for SSA. The learning rate is decayed every **30 epochs**. A batch size of 256 is used across all experiments.

For SSA, rank reduction begins after the first 20 epochs and proceeds every 10 epochs, retaining **90% matrix energy**. Loss coefficients are kept consistent with CIFAR-100 ($\alpha, \beta, \gamma = 1, 0.01, 0.1$), with adjustments only to the overall learning rates for layerwise updates.

Normalization Layers For batchnorm layers, we use it mostly for the forward process, and do not involve in the layerwise backward process (as the gradient calculation process is not sequential). We also use layernorm as an alternative to batchnorm. From our experiments, we find that layernorm is more suited to our method than batchnorm (empirically determined).

Hyperparameter Tuning We introduce three hyperparameters in our local layerwise loss objective. We put lower values of hyperparameters for singular value orthogonality regularizer, as they are non-convex.

ResNet Local Module Splitting In our experiments, each residual block in ResNet is treated as a fundamental layer. For ResNet-32, this results in a total of 16 fundamental layers, with each block encapsulating key functions like identity mapping and feature transformation.

3. Sensitivity Analysis of Loss Coefficients

We conduct a sensitivity analysis on the weights of the three components in our composite loss: cross-entropy (α), alignment loss (β), and orthogonality regularizer (γ), as shown in Figure 1. The results highlight that SSA is robust across a wide range of values for all three hyperparameters. Accuracy peaks when the CE loss weight is close to 1, which reflects its dominant role in guiding supervised learning. The model maintains stable performance even as the alignment and orthogonality weights vary, indicating that these auxiliary terms contribute effectively without destabilizing training. Notably, performance degrades when alignment and orthogonality weights are either too low (insufficient guidance) or too high (over-regularization), underscoring the importance of balanced loss weighting.

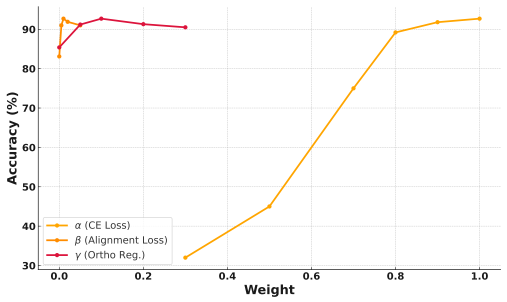

Figure 1. Ablation on loss-component weights

References

[1] Albert Jiménez Sanfiz and Mohamed Akrout. Benchmarking the accuracy and robustness of feedback alignment algorithms. *arXiv preprint arXiv:2108.13446*, 2021. 2