

A List of Symbols Used in This Paper

Symbol	Description
$h^{(\ell)}$	Representation at layer ℓ of the network
$\phi^{(\ell)}$	Nonlinear transformation at layer ℓ
$\mathbf{W}^{(\ell)}$	Weight matrix at layer ℓ
$h^{(L)}$	Penultimate-layer representation
$\delta x^{(0)}$	Small perturbation in input
$\delta h^{(\ell)}$	Perturbation at layer ℓ
$\lambda^{(L)}(x)$	Finite-time Lyapunov exponent (FTLE) of input x
λ	Unnormalized FTLE, i.e., $\lambda = L\lambda^{(L)}$
h_i^k	Representation of sample i in class k
\mathbf{w}_k	Classifier weight vector for class k
H^k	Matrix of representations for class k
\mathbf{W}	Classifier weight matrix
Y	Label matrix for all samples
\mathcal{L}	Loss function (typically MSE)
$\lambda_{\mathbf{W}}$	Regularization coefficient for \mathbf{W}
λ_H	Regularization coefficient for H
H_0^\perp	Initial representation orthogonal to \mathbf{W}
α	Stretching factor (inter-class expansion)
β	Compression factor (intra-class collapse)
$\lambda_{\text{trained}}^{k,k'}$	FTLE between class k and k' after training
$\lambda_0^{k,k'}$	FTLE between class k and k' at initialization
\mathcal{T}_t	t -th task in continual learning
θ_t	Network parameters after task t
$\mathcal{D}_{\text{train}}^{(t)}$	Training set of task t
$\mathcal{D}_{\text{test}}^{(t)}$	Test set of task t
m	Mixup interpolation coefficient
x^m, y^m	Interpolated input and target in mixup
M	Confidence amplification factor in Generalized Mixup

B Proof of Theorem 3.1.1

We prove Theorem 3.1.1 in main text that we restate as follows.

Theorem B.1 (Optimal Solution for Binary Classification) *Under UFM and the assumption that both classes contain n samples, $|A| = |B| = n$, consider the following optimization problem:*

$$\min_{\mathbf{W}, H} f(\mathbf{W}, H) := \frac{1}{2} \|\mathbf{W}H + \mathbf{b}\mathbf{1}^\top - Y\|_{\text{MSE}}^2 + \frac{\lambda_{\mathbf{W}}}{2} \|\mathbf{W}\|^2 + \frac{\lambda_H}{2n} \|H - H_0^\perp\|^2, \quad (16)$$

where $\lambda_{\mathbf{W}}$ and λ_H are the coefficients of the two regularization terms that control the magnitude of the classifier weights, \mathbf{W} , and deviation of the learned representations H from the initial representations H_0^\perp orthogonal to \mathbf{W} . The optimal solution obeys:

$$\mathbf{w}_A^* = -\mathbf{w}_B^*, \quad h_i^{k,*} = \sqrt{\lambda_{\mathbf{W}}/\lambda_H} \mathbf{w}_k + h_{i,0}^{k,\perp}, \quad (17)$$

where $k = A$ or B , and $h_{i,0}^{k,\perp}$ is the k -component of H_0^\perp .

We begin by computing the gradients of the loss function with respect to \mathbf{W} and H , and setting them to zero to obtain the optimality conditions.

The gradient of \mathcal{L} with respect to \mathbf{W} is given by

$$\nabla_{\mathbf{W}} \mathcal{L} = (\mathbf{W}H + \mathbf{b}\mathbf{1}^\top - Y)H^\top + \lambda_{\mathbf{W}} \mathbf{W} = 0. \quad (18)$$

The gradient with respect to H is

$$\nabla_H \mathcal{L} = \mathbf{W}^\top (\mathbf{W}H + \mathbf{b}\mathbf{1}^\top - Y) + \frac{\lambda_H}{n} (H - H_0^\perp) = 0. \quad (19)$$

To connect the two optimality conditions, we multiply both sides of Equation (18) on the left by \mathbf{W}^\top , yielding

$$\mathbf{W}^\top (\mathbf{W}H + \mathbf{b}\mathbf{1}^\top - Y)H^\top + \lambda_{\mathbf{W}} \mathbf{W}^\top \mathbf{W} = 0. \quad (20)$$

Substituting Equation (19) into the first term of Equation (20) gives

$$-\frac{\lambda_H}{n} (H - H_0^\perp)H^\top + \lambda_{\mathbf{W}} \mathbf{W}^\top \mathbf{W} = 0. \quad (21)$$

Rearranging terms, we obtain the following relationship between the optimal weight matrix \mathbf{W} and the learned representation matrix H :

$$\lambda_{\mathbf{W}} \mathbf{W}^\top \mathbf{W} = \frac{\lambda_H}{n} (H - H_0^\perp)H^\top. \quad (22)$$

Based on prior results from Neural Collapse theory [36], the optimal classification weight matrix converges to a Simplex Equiangular Tight Frame structure. In the binary classification case, this reduces to a simple antipodal configuration:

$$\mathbf{w}^A = -\mathbf{w}^B. \quad (23)$$

Assuming the classification weights have reached their optimal configuration, the training dynamics of the representation h_i^k under the MSE loss yield the following gradient:

$$-\frac{\partial \mathcal{L}_{\text{MSE}}}{\partial h_i^k} = (1 - \mathbf{w}_k^\top h_i^k) \mathbf{w}_k + (0 - \mathbf{w}_{k'}^\top h_i^k) \mathbf{w}_{k'}, \quad (24)$$

where k is the correct class and k' is the incorrect class index.

This gradient lies entirely in the subspace spanned by \mathbf{w}_k and $\mathbf{w}_{k'}$, implying that the updates to h_i^k only occur in the plane defined by the classification weight vectors. Consequently, the component of h_i^k orthogonal to this plane remains unchanged during training.

Thus, the representation h_i^k can be decomposed as:

$$h_i^k = h_i^\parallel + h_{0,i}^\perp,$$

where $h_i^\parallel \in \text{span}\{\mathbf{w}_k, \mathbf{w}_{k'}\}$ and $h_{0,i}^\perp \perp \text{span}\{\mathbf{w}_k, \mathbf{w}_{k'}\}$. The term $h_{0,i}^\perp$ denotes the component of the initial representation that is orthogonal to the subspace spanned by the classification vectors \mathbf{w}_k and $\mathbf{w}_{k'}$.

Then Equation 22 becomes:

$$\lambda_{\mathbf{W}} \mathbf{W}^\top \mathbf{W} = \frac{\lambda_H}{n} H^\parallel H^{\parallel,\top},$$

where H_0^\parallel denotes the component of the initial representations that is parallel to the subspace spanned by the classification weight vectors.

This constraint provides a closed-form relationship between the classifier weights and the representation geometry at optimality.

For simplicity, we will denote $\|\mathbf{W}\|_F^2 = E$ and thus $\|H^\parallel\|_F^2 = \frac{n\lambda_{\mathbf{W}}}{\lambda_H} E$.

We now continue the proof by deriving a lower bound of the MSE loss. Our goal is to express the lower bound in terms of the Frobenius norm $E = \|\mathbf{W}\|_F^2$.

Recall the MSE loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2} \sum_i \|\mathbf{W}h_i + \mathbf{b}\mathbf{1}^\top - y_i\|^2.$$

Since the orthogonal component $h_{0,i}^\perp$ remains unchanged with $\mathbf{W}h_{0,i}^\perp = 0$, we can express the MSE loss as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2} \sum_i \left\| \mathbf{W}h_i^\parallel + \mathbf{b} - y_i \right\|^2.$$

To enable a tight application of the reverse triangle inequality, we re-center the target labels by defining:

$$b := \frac{1}{2}(y_A + y_B),$$

and rewrite the labels as:

$$\tilde{y}_i := y_i - b \in \{\pm(y_A - b)\}.$$

Now, the MSE loss becomes:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2} \sum_i \left\| \mathbf{W}h_i^\parallel - \tilde{y}_i \right\|^2.$$

We now apply the reverse triangle inequality:

$$\left\| \mathbf{W}h_i^\parallel - \tilde{y}_i \right\| \geq \left| \|\tilde{y}_i\| - \|\mathbf{W}h_i^\parallel\| \right|,$$

which becomes tight when $\mathbf{W}h_i^\parallel \parallel \tilde{y}_i$, and squaring both sides yields:

$$\left\| \mathbf{W}h_i^\parallel - \tilde{y}_i \right\|^2 \geq \left(\|\tilde{y}_i\| - \|\mathbf{W}h_i^\parallel\| \right)^2.$$

Next, since h_i^\parallel is aligned with the classification direction \mathbf{w}_k , we can write (see a brief proof in Section B.1):

$$\|\mathbf{W}h_i^\parallel\| = \|\mathbf{W}\|_F \cdot \|h_i^\parallel\|.$$

Therefore, we obtain:

$$\left\| \mathbf{W}h_i^\parallel - \tilde{y}_i \right\|^2 \geq \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \|h_i^\parallel\| \right)^2.$$

Substituting into the MSE loss gives:

$$\mathcal{L}_{\text{MSE}} \geq \frac{1}{2} \sum_i \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \|h_i^\parallel\| \right)^2.$$

From earlier, we have $\|H^\parallel\|_F^2 = \sum_i \|h_i^\parallel\|^2 = \frac{n\lambda\mathbf{w}}{\lambda_H} E$, where $E = \|\mathbf{W}\|_F^2$. Let n denote the total number of samples. Define the average squared norm:

$$\frac{1}{n} \sum_i \|h_i^\parallel\|^2 = \frac{1}{n} \cdot \frac{n\lambda\mathbf{w}}{\lambda_H} E.$$

Now apply Jensen's inequality to the convex function $f(x) = (b - a\sqrt{x})^2$ for $a > 0$, to get:

$$\frac{1}{n} \sum_i \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \|h_i^\parallel\| \right)^2 \geq \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \sqrt{\frac{1}{n} \sum_i \|h_i^\parallel\|^2} \right)^2.$$

Substitute the average squared norm:

$$\frac{1}{n} \sum_i \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \|h_i^\parallel\| \right)^2 \geq \left(\|\tilde{y}_i\| - \|\mathbf{W}\|_F \cdot \sqrt{\frac{\lambda\mathbf{w}}{\lambda_H} E} \right)^2.$$

Recall $\|\mathbf{W}\|_F = \sqrt{E}$, so we obtain:

$$\mathcal{L}_{\text{MSE}} \geq \frac{n}{2} \left(\|\tilde{y}_i\| - \sqrt{\frac{\lambda\mathbf{w}}{\lambda_H} E} \right)^2.$$

Thus, the original optimization objective becomes:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \frac{\lambda_{\mathbf{W}}}{2} \|\mathbf{W}\|_F^2 + \frac{\lambda_H}{2n} \|H - H_0^\perp\|_F^2 \quad (25)$$

$$\geq \frac{n}{2} \left(\|\tilde{y}_i\| - \sqrt{\frac{\lambda_{\mathbf{W}}}{\lambda_H}} \cdot E \right)^2 + \frac{\lambda_{\mathbf{W}}}{2} E + \frac{\lambda_H}{2n} \cdot \frac{n\lambda_{\mathbf{W}}}{\lambda_H} E \quad (26)$$

$$= \frac{n}{2} \left(\|\tilde{y}_i\| - \sqrt{\frac{\lambda_{\mathbf{W}}}{\lambda_H}} \cdot E \right)^2 + \lambda_{\mathbf{W}} E. \quad (27)$$

This gives a clean lower bound entirely expressed as a function of $E = \|\mathbf{W}\|_F^2$. Since the loss function is continuous and coercive in E , the minimum is achieved at some finite E^* .

The conditions for the inequality to hold require representations h_i^\parallel have equal norm (which can also be intuitively obtained from the symmetry conditions) :

$$\|h_i^\parallel\| = \text{const}, \quad \forall i.$$

Also we have the following constrain equation 22:

$$\lambda_{\mathbf{W}} \|\mathbf{W}\|_F^2 = \frac{\lambda_H}{n} \|H\|_F^2 \Rightarrow \frac{1}{n} \sum_i \|h_i^\parallel\|^2 = \frac{\lambda_{\mathbf{W}}}{\lambda_H} \|\mathbf{W}\|_F^2.$$

Therefore, we have:

$$\|h_i^\parallel\| = \sqrt{\frac{\lambda_{\mathbf{W}}}{\lambda_H}} \cdot \|\mathbf{w}_k\|, \quad \text{for all } i \text{ in class } k.$$

Thus the representation becomes:

$$h_i^{k,*} = \sqrt{\frac{\lambda_{\mathbf{W}}}{\lambda_H}} \mathbf{w}_k + h_{i,0}^{k,\perp},$$

where $h_{i,0}^{k,\perp} \in \text{span}(\mathbf{W})^\perp$ is the fixed orthogonal component inherited from the initial representation.

B.1 A brief proof of $\|\mathbf{W}h_i^\parallel\| = \|\mathbf{W}\|_F \cdot \|h_i^\parallel\|$.

Under the neural collapse assumption 23 that the classification weight matrix is structured as

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_A^\top \\ -\mathbf{w}_A^\top \end{bmatrix},$$

and we define

$$h_i^\parallel := \tau_i \mathbf{w}_A, \quad \text{for some } \tau_i \in \mathbb{R},$$

to emphasize that the representation lies along the direction of \mathbf{w}_A , and τ_i is arbitrary.

We have:

$$\mathbf{W}h_i^\parallel = \tau_i \begin{bmatrix} \mathbf{w}_A^\top \mathbf{w}_A \\ -\mathbf{w}_A^\top \mathbf{w}_A \end{bmatrix} = \tau_i \|\mathbf{w}_A\|^2 \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Thus, the squared norm becomes:

$$\|\mathbf{W}h_i^\parallel\|^2 = 2\tau_i^2 \|\mathbf{w}_A\|^4.$$

On the other hand:

$$\|h_i^\parallel\| = |\tau_i| \cdot \|\mathbf{w}_A\|, \quad \Rightarrow \quad \|\mathbf{W}\|_F^2 = \|\mathbf{w}_A\|^2 + \|-\mathbf{w}_A\|^2 = 2\|\mathbf{w}_A\|^2.$$

Therefore,

$$\|\mathbf{W}h_i^\parallel\| = \sqrt{2} \cdot |\tau_i| \cdot \|\mathbf{w}_A\|^2 = \sqrt{2} \cdot \|\mathbf{w}_A\| \cdot |\tau_i| \cdot \|\mathbf{w}_A\| = \|\mathbf{W}\|_F \cdot \|h_i^\parallel\|.$$

This confirms that the equality

$$\|\mathbf{W}h_i^\parallel\| = \|\mathbf{W}\|_F \cdot \|h_i^\parallel\|$$

holds exactly under this structural assumption.

C Verification of High-Dimensional Data Properties in ImageNet

In the main text, we assume that real-world data distributions exhibit two critical characteristics of high-dimensional spaces: the *measure of concentration* and the *empty space phenomenon*. To verify these assumptions, we conducted the following experiments on the ImageNet dataset:

1. **Measure of Concentration:** We calculated the average pairwise distance between samples in the representation space, along with the variance of these distances. This analysis helps to verify whether the distances concentrate around their mean, as expected in high-dimensional spaces.
2. **Empty Space Phenomenon:** To assess this property, we randomly sampled 1000 points in the representation space and computed the minimum distance between each sample point and these randomly generated points. This test evaluates the relative sparsity of data in high-dimensional spaces.

The results, shown in figure 6, confirm that the ImageNet dataset adheres to both the *measure of concentration* and the *empty space phenomenon*. These findings validate our assumption about the high-dimensional nature of real-world data distributions.

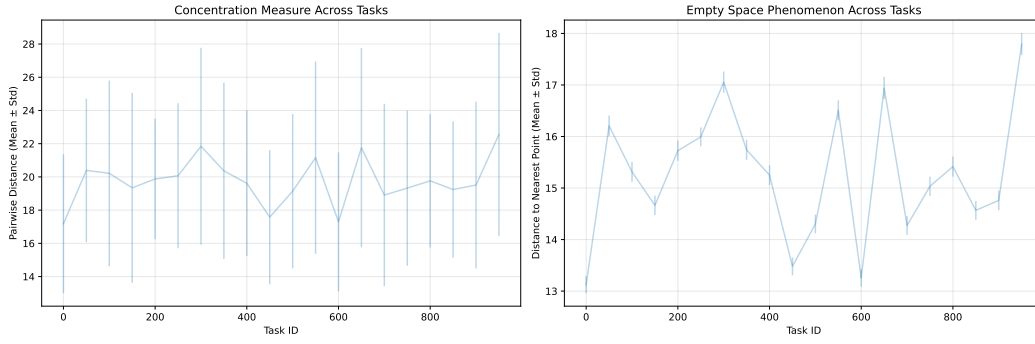


Figure 6: Verification of High-Dimensional Data Properties in ImageNet.

D Proof of Theorem 3.1.2

We prove Theorem 3.1.2 from the main text that we restate as follows.

Theorem D.1 (FTLE in Representation Learning) *The distance between two samples i and j from classes k and k' in representation space, $\Delta h_{ij}^{k,k'} \equiv h_i^k - h_j^{k'}$, can be quantified by FTLE, λ_0 and λ_{trained} , each denotes its values before and after training, respectively.*

(1) *For samples from different classes ($k \neq k'$), training produces a positive FTLE ridge, stretching the space in-between, and the magnitude of stretch is captured by a stretching factor, α :*

$$\log \alpha = \lambda_{\text{trained}}^{k,k'} - \lambda_0^{k,k'} > 0, \quad (28)$$

(2) *For samples of the same class ($k = k'$), training carves out a valley of negative FTLE, compressing the space in-between, and the degree of compression is given by a compressing factor, β :*

$$\log \beta = \lambda_{\text{trained}}^{k,k} - \lambda_0^{k,k} < 0. \quad (29)$$

According to Theorem 3.1.1, the optimal representations satisfy:

$$h_i^{k,*} = \sqrt{\frac{\lambda_{\mathbf{w}}}{\lambda_H}} \mathbf{w}_k + h_{i,0}^{k,\perp},$$

where the trainable component lies entirely in the classification plane spanned by $\{\mathbf{w}_A, \mathbf{w}_B\}$, and the orthogonal component $h_{i,0}^{k,\perp}$ remains unchanged from initialization.

The trained representations become:

$$h_i^{A,*} = \sqrt{\frac{\lambda \mathbf{w}}{\lambda_H}} \mathbf{w}_A + h_{i,0}^{A,\perp}, \quad h_j^{B,*} = \sqrt{\frac{\lambda \mathbf{w}}{\lambda_H}} \mathbf{w}_B + h_{j,0}^{B,\perp}.$$

Hence, the inter-class representational difference after training becomes:

$$\Delta_{\text{trained}} h_{ij}^{A,B} = h_i^{A,*} - h_j^{B,*} = \left(h_{i,0}^{A,\perp} - h_{j,0}^{B,\perp} \right) + \sqrt{\frac{\lambda \mathbf{w}}{\lambda_H}} (\mathbf{w}_A - \mathbf{w}_B).$$

In contrast, the inter-class representational difference before training is given by:

$$\Delta_0 h_{ij}^{A,B} = h_{i,0}^{A,\perp} - h_{j,0}^{B,\perp} + \left\langle h_{i,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} - \left\langle h_{j,0}^B, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|},$$

where the initial representations are projected onto the direction of \mathbf{w}_A .

The change in inter-class representational distance due to training satisfies:

$$\left\| \Delta_{\text{trained}} h_{ij}^{A,B} \right\|^2 - \left\| \Delta_0 h_{ij}^{A,B} \right\|^2 \quad (30)$$

$$= \frac{\lambda \mathbf{w}}{\lambda_H} \cdot \|\mathbf{w}_A - \mathbf{w}_B\|^2 - \left\| \left\langle h_{i,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} - \left\langle h_{j,0}^B, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2 \quad (31)$$

$$= \left(4 \frac{\lambda \mathbf{w}}{\lambda_H} - \left| \frac{\langle h_{i,0}^A - h_{j,0}^B, \mathbf{w}_A \rangle}{\|\mathbf{w}_A\|^2} \right|^2 \right) \cdot \|\mathbf{w}_A\|^2 \quad (32)$$

$$= \left(4 \frac{\lambda \mathbf{w}}{\lambda_H} \|\mathbf{w}_A\|^2 - \left| \left\langle h_{i,0}^A - h_{j,0}^B, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right|^2 \right) \cdot \left\| \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2. \quad (33)$$

Similarly, the change in intra-class representational distance due to training is given by:

$$\left\| \Delta_{\text{trained}} h_{i,j}^{A,A} \right\|^2 - \left\| \Delta_0 h_{i,j}^{A,A} \right\|^2 \quad (34)$$

$$= \left\| h_i^{A,*} - h_j^{A,*} \right\|^2 - \left\| h_{i,0}^A - h_{j,0}^A \right\|^2 \quad (35)$$

$$= \left\| \left(h_{i,0}^{A,\perp} - h_{j,0}^{A,\perp} \right) + \sqrt{\frac{\lambda \mathbf{w}}{\lambda_H}} (\mathbf{w}_A - \mathbf{w}_A) \right\|^2 \quad (36)$$

$$= \left\| \left(h_{i,0}^{A,\perp} - h_{j,0}^{A,\perp} \right) + \left(\left\langle h_{i,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle - \left\langle h_{j,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right) \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2 \quad (37)$$

$$= - \left\| \left(\left\langle h_{i,0}^A - h_{j,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right) \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2 \quad (38)$$

$$= - \left\| \left(\frac{\langle h_{i,0}^A - h_{j,0}^A, \mathbf{w}_A \rangle}{\|\mathbf{w}_A\|^2} \right) \mathbf{w}_A \right\|^2 \quad (39)$$

$$= - \left\| \left\langle h_{i,0}^A - h_{j,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2 \quad (40)$$

$$(41)$$

Thus, the key is to evaluate the following projection term:

$$\left| \left\langle h_{i,0}^K - h_{j,0}^K, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right|^2.$$

This can be decomposed as:

$$\left| \left\langle h_{i,0}^K, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle - \left\langle h_{j,0}^K, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right|^2.$$

Note that this expression captures the difference in the initial projections of the representations $h_{i,0}^K$ and $h_{j,0}^K$ onto the normalized classification direction $\mathbf{w}_A/\|\mathbf{w}_A\|$.

Under this assumption, the original expression reduces to a geometric comparison between three random unit vectors in high-dimensional space: two initial representations z_1, z_2 , and one classification direction w . Let us define:

$$\cos \theta_1 = \cos(\angle(z_1, w)), \quad \cos \theta_2 = \cos(\angle(z_2, w)).$$

We are interested in the expected deviation:

$$z = |\cos \theta_1 - \cos \theta_2|^2.$$

According to Lemma D.1, the distribution of $\cos \theta$ between two independent unit vectors in d -dimensional Euclidean space follows the density:

$$p(\cos \theta) = \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2}) \sqrt{\pi}} (1 - \cos^2 \theta)^{(d-3)/2}.$$

Let $\cos \theta_1 \sim p(x)$, $\cos \theta_2 \sim p(y)$. The distribution of their absolute difference $z = |x - y|^2$ can be written as:

$$p_z(z) = \int_{-1}^1 \int_{-1}^1 p(x)p(y) \delta(z - |x - y|^2) dx dy.$$

By symmetry, this simplifies to:

$$p_z(z) = \frac{C^2}{\sqrt{z}} \int_{-1}^{1-\sqrt{z}} (1 - x^2)^{(d-3)/2} \cdot (1 - (x + \sqrt{z})^2)^{(d-3)/2} dx,$$

where C is the normalization constant ensuring $\int_{-1}^1 p(x) dx = 1$.

Although the exact form of $p_z(z)$ is analytically intractable, it can be efficiently evaluated numerically. In the following figure, we plot the cumulative distribution function (CDF) of z . Here, the dimension d refers to the width of the representation space, i.e., the dimensionality of the representation $h \in \mathbb{R}^d$. As d increases, the distribution becomes increasingly concentrated near zero. Conversely, when the dimension d is small, the distribution of z becomes more dispersed and exhibits heavier mass at larger nonzero values.

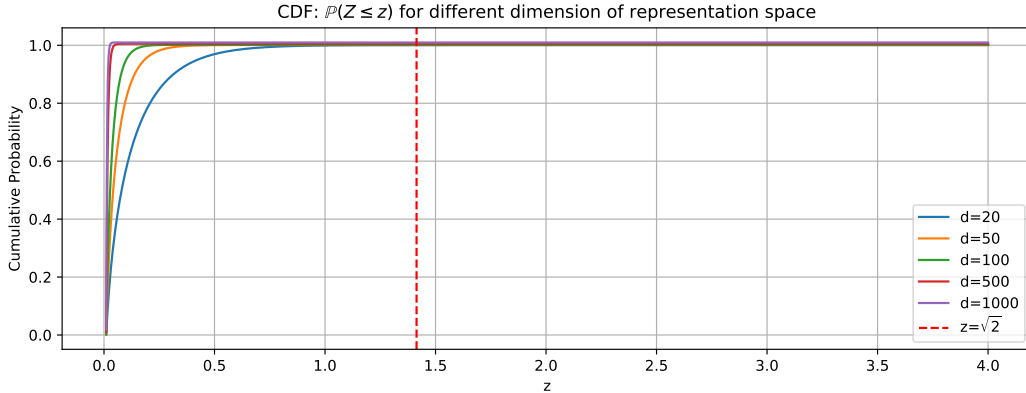


Figure 7: Numerical solution of the CDF of z .

Within-Class Representation Collapse. For the within-class region, in Eq 40, we have already shown that training changes the pairwise distance between representations as:

$$\left\| \Delta_{\text{trained}} h_{ij}^{K,K} \right\|^2 - \left\| \Delta_0 h_{ij}^{K,K} \right\|^2 = - \left\| \left\langle h_{i,0}^A - h_{j,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\|^2.$$

As we have analyzed earlier, the scalar projection term

$$z = \left\langle h_{i,0}^A - h_{j,0}^A, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle$$

follows a distribution that becomes increasingly concentrated near zero as the representation dimensionality d grows, and conversely, spreads more widely when d is small.

Define the within-class shrinkage factor as:

$$\left\| \Delta_{\text{trained}} h_{ij}^{K,K} \right\| = \beta \cdot \left\| \Delta_0 h_{ij}^{K,K} \right\|, \quad \text{with } \beta < 1.$$

Then the FTLE after training is given by:

$$\lambda_{\text{trained}}^{K,K}(x) = \lambda_0^{K,K}(x) + \log \beta(x).$$

This implies that the FTLE will slightly decrease or remain nearly unchanged $\beta \lesssim 1$, when d is large, but may shrink significantly when d is small, due to larger projection fluctuations.

Importantly, while this within-task shrinkage effect may appear mild for a single task, in the continual learning setting it can accumulate over time, especially when repeated over multiple tasks. This accumulated shrinkage can lead to a significant compression of the local representation space, i.e., representation space collapse. This analysis is consistent with our empirical observations: in low-dimensional settings, collapse occurs reliably even for a small number of tasks, while in higher-dimensional spaces, collapse may or may not occur depending on the specific training setup and number of tasks.

Boundary Space Over-Stretched. For the inter-class region, Eq. 40 shows that training modifies the pairwise distance between representations of different classes as:

$$\left\| \Delta_{\text{trained}} h_{ij}^{A,B} \right\|^2 - \left\| \Delta_0 h_{ij}^{A,B} \right\|^2 = \left(4 \cdot \frac{\lambda \mathbf{w}}{\lambda_H} \|\mathbf{w}_A\|^2 - \left| \left\langle h_{i,0}^A - h_{j,0}^B, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right|^2 \right).$$

To estimate the first term, we recall from the optimization analysis of Equation 27 in Section D that the total loss satisfies:

$$\mathcal{L}(E) \geq \frac{n}{2} \left(\|\tilde{y}_i\| - \sqrt{\frac{\lambda \mathbf{w}}{\lambda_H}} \cdot E \right)^2 + \lambda \mathbf{w} E,$$

where the minimum occurs approximately at

$$E^* = \sqrt{\frac{\lambda_H}{\lambda \mathbf{w}}} \cdot \|\tilde{y}_i\| - \frac{\lambda_H}{n} \approx \sqrt{\frac{\lambda_H}{\lambda \mathbf{w}}} \cdot \|\tilde{y}_i\|.$$

This is because n is typically large compared to λ_H , and $\lambda \mathbf{w} \sim \lambda_H$ in magnitude. Hence,

$$\|\mathbf{w}_A\|^2 \approx \frac{1}{2} \cdot \sqrt{\frac{\lambda_H}{\lambda \mathbf{w}}} \cdot \|\tilde{y}_i\| \quad \Rightarrow \quad 4 \cdot \frac{\lambda \mathbf{w}}{\lambda_H} \|\mathbf{w}_A\|^2 \approx \sqrt{2}.$$

In Figure 7, we overlay this estimate (red dashed line) on the empirical CDF of the squared projection term

$$z = \left| \left\langle h_{i,0}^A - h_{j,0}^B, \frac{\mathbf{w}_A}{\|\mathbf{w}_A\|} \right\rangle \right|^2,$$

we observe that in both low- and high-dimensional spaces, most values of z fall below this estimated threshold. This indicates a net expansion of the representation space between classes during training.

Moreover, the distribution of the projection term z becomes more concentrated around zero as the representation dimensionality d increases, while it spreads more widely when d is small. Therefore, although expansion dominates across dimensions, it is more significant in high-dimensional spaces, where the lower baseline projection leads to faster growth.

We formalize this expansion using a stretch factor:

$$\|\Delta_{\text{trained}} h_{ij}^{A,B}\| = \alpha \cdot \|\Delta_0 h_{ij}^{A,B}\|, \quad \alpha > 1.$$

The resulting FTLE becomes:

$$\lambda_{\text{trained}}^{K,K'}(x) = \lambda_0^{K,K'}(x) + \log \alpha(x),$$

indicating a positive FTLE ridge across inter-class boundaries, and reflecting a local "stretching effect" that enhances class separability.

While the effect of class boundary expansion is immediately pronounced in high-dimensional settings, its manifestation in low-dimensional networks can be subtle within a single task. However, under continual learning, where multiple tasks accumulate over time, the expansion effect may progressively intensify and eventually give rise to chaotic behavior. Consequently, we consistently observe clear boundary stretching in high-dimensional models, whereas in low-dimensional models, such dynamics may emerge only after sufficient task accumulation. This observation aligns with our findings reported in the main text.

Lemma D.1 *In high-dimensional spaces, the angle θ between two randomly sampled vectors $x, y \in \mathbb{R}^d$ follows the distribution:*

$$p_d(\theta) = \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}} \sin^{d-2} \theta,$$

where $\Gamma(\cdot)$ is the gamma function[32].

E Experimental Setup for Verifying LoP in Real-World Datasets

To validate our discovery of two different types of LoP across realistic datasets, we use the experimental setup described in Supplementary H, using identical architectures, training protocols, and hyperparameters. Across all settings, we consistently observe the two types of LoP as well as hybrid cases where both types coexist, differing only in their onset speed and severity.

In the main text Fig. 4, we visualize model performance under four representative single-run configurations :

- (a) ReLU + CE with the architecture in Table 5
- (b) Tanh + MSE with the architecture in Table 6
- (c) Tanh + MSE with the architecture in Table 5
- (d) ReLU + CE with the architecture in Table 6

All other experimental conditions remain identical across these runs.

F Generalized Mixup Pseudo Code

Generalized Mixup (G-Mixup) is designed as a simple, plug-and-play module that can be seamlessly integrated into existing training pipelines. It minimally modifies the classical *Mixup* [35] by adapting the label interpolation rule depending on whether the two samples belong to the same class (intra-class) or different classes (inter-class). It supports both Mean Squared Error (MSE) and Negative Log-Likelihood (NLL) loss.

Key idea:

- For **inter-class mixup**, standard Mixup applies: convex interpolation of both input and label.

- For **intra-class mixup**, only the dominant class label is amplified by a factor depending on $|0.5 - m|$ to encourage intra-class diversity while conserving class identity.

In this work, we set the intra-class amplification factor to $M = 0.2$. To prevent label probabilities from exceeding 1, we pre-process all labels by assigning the dominant class a value of $1 - M$, and the remaining M is equally distributed across the other $C - 1$ classes.

Algorithm 1 Generalized Mixup (G-Mixup) for a Mini-Batch

Require: Mini-batch $\{(x_i, y_i^{\text{raw}})\}_{i=1}^N$, mixup parameter α , amplification factor M , number of classes C

- 1: **Preprocess labels:** for each y_i^{raw} , construct y_i as:
 - $y_i[k] = 1 - M$, where $k = \arg \max(y_i^{\text{raw}})$
 - $y_i[\text{others}] = \frac{M}{C-1}$
 - 2: Sample $m \sim \text{Beta}(\alpha, \alpha)$
 - 3: Shuffle mini-batch to get (x'_i, y'_i)
 - 4: Compute mixed inputs: $x_i^m = mx_i + (1 - m)x'_i$
 - 5: **if** $\arg \max(y_i) == \arg \max(y'_i)$ (**intra-class**) **then**
 - 6: $y_i^m[k] = y_i[k] + \frac{M}{2} + M \cdot |0.5 - m|$
 - 7: $y_i^m[\text{others}] = \frac{1 - y_i^m[k]}{C-1}$
 - 8: **else**
 - 9: $y_i^m = my_i + (1 - m)y'_i$ {Use M -adjusted labels}
 - 10: **end if**
 - 11: Compute prediction $\hat{y}_i = f(x_i^m)$
 - 12: **if** loss is MSE **then**
 - 13: $\mathcal{L} = \frac{1}{N} \sum_i \|\hat{y}_i - y_i^m\|^2$
 - 14: **else if** loss is NLL **then**
 - 15: $\mathcal{L} = \frac{1}{N} \sum_i \text{CrossEntropy}(\hat{y}_i, y_i^m)$
 - 16: **end if**
 - 17: Backpropagate \mathcal{L}
-

G Experimental Setup for Continual ImageNet

We evaluate *G-Mixup* on the Continual ImageNet benchmark, following a widely adopted binary task formulation. The full dataset used in this task consists of 1,000 classes, each with 700 images. For each class, 600 samples are used for training and 100 for testing. A continual learning sequence is constructed by randomly pairing classes to create binary classification tasks. Each task comprises 1,200 training samples and 200 test samples, drawn from two classes. Models are trained for 250 epochs using mini-batches of size 100. All images are downsampled to 32×32 resolution to reduce computational cost.

For all experiments, we adopt a convolutional neural network consisting of three convolutional layers with max-pooling, followed by three fully connected layers. To study the impact of representation dimensionality, we employ two variants: a wide network with larger hidden dimensions, identical to the one used in [7] (see Table 6), representing the high-dimensional regime; and a narrow network with reduced layer widths (see Table 5), corresponding to the low-dimensional case. The final output layer has two units representing the current task’s classes. At the start of each task, the output heads are reset to zero—a common practice in continual learning benchmarks—although we note that this introduces privileged information about task boundaries. All methods are trained using SGD with momentum (set to 0.9), and models are initialized once at the beginning of the entire continual learning sequence. Learning rates are set as 0.01; results reported are averaged over 5 independent runs.

We compare *G-Mixup* with multiple baselines and LoP mitigation strategies. Two reference points are included: (1) **Reinit**, which re-initializes the model for every new task, and (2) **No Intervention (No Intv.)**, which applies naive continual training without any LoP countermeasures. For fair comparison,

We also include representative methods from three major strategies for mitigating LoP: **L2 Init** [17] (a regularization-based method, with the weight decay coefficient fixed at 5×10^{-4}), **LayerNorm** [21] (an architecture-based approach, where LayerNorm is applied before the activation function at every layer—including both convolutional and fully connected layers—except for the final output layer), and **CBP** [7] (a reset-based method).

Unless otherwise specified, all methods use the ELU activation function and MSE loss. The only exception is **CBP**, which uses ReLU as in its original implementation [7]. This choice is motivated by the fact that CBP relies on tracking the utility of individual neurons, and ELU—due to its non-zero minimum—can distort utility estimation by introducing persistent low-level activation. We further confirmed that CBP performs better with ReLU than with ELU under our experimental settings. All other hyperparameters and training schedules are kept consistent across methods to ensure fair comparison.

Table 5: SmallConvNet Architecture Details.

Layer 1: Convolutional + Max-Pooling			
Number of Filters	16	Activation	ELU/Tanh/ReLU
Convolutional Filter Shape	(5,5)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(2,2)
Layer 2: Convolutional + Max-Pooling			
Number of Filters	32	Activation	ELU/Tanh/ReLU
Convolutional Filter Shape	(3,3)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(2,2)
Layer 3: Convolutional + Max-Pooling			
Number of Filters	64	Activation	ELU/Tanh/ReLU
Convolutional Filter Shape	(3,3)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(2,2)
Layer 4: Fully Connected			
Output Size	16	Activation	ELU/Tanh/ReLU
Layer 5: Fully Connected			
Output Size	16	Activation	ELU/Tanh/ReLU
Layer 6: Fully Connected			
Output Size	2	Activation	None (Linear)

Table 6: ConvNet Architecture Details.

Layer 1: Convolutional + Max-Pooling			
Number of Filters	32	Activation	ELU/Tanh/ReLU
Conv. Filter Shape	(5,5)	Conv. Filter Stride	(1,1)
Max-Pool Filter Shape	(2,2)	Max-Pool Filter Stride	(2,2)
Layer 2: Convolutional + Max-Pooling			
Number of Filters	64	Activation	ELU/Tanh/ReLU
Conv. Filter Shape	(3,3)	Conv. Filter Stride	(1,1)
Max-Pool Filter Shape	(2,2)	Max-Pool Filter Stride	(2,2)
Layer 3: Convolutional + Max-Pooling			
Number of Filters	128	Activation	ELU/Tanh/ReLU
Conv. Filter Shape	(3,3)	Conv. Filter Stride	(1,1)
Max-Pool Filter Shape	(2,2)	Max-Pool Filter Stride	(2,2)
Layer 4: Fully Connected			
Output Size	128	Activation	ELU/Tanh/ReLU
Layer 5: Fully Connected			
Output Size	128	Activation	ELU/Tanh/ReLU
Layer 6: Fully Connected			
Output Size	2	Activation	None (Linear)

H Continual ImageNet Results

The performance of wide and narrow networks on Continual ImageNet is summarized in Table 7 and Table 8.

Table 7: SmallConv Test Accuracy Results on Continual ImageNet

Task ($\times 1000$)	0-1	1-2	2-3	3-4	4-5
No Intv.	0.817 (± 0.070)	0.805 (± 0.085)	0.562 (± 0.089)	0.500 (± 0.000)	0.500 (± 0.000)
Retrained	0.853 (± 0.065)	0.845 (± 0.066)	0.845 (± 0.068)	0.840 (± 0.067)	0.840 (± 0.072)
L2 init	0.804 (± 0.055)	0.796 (± 0.059)	0.786 (± 0.054)	0.785 (± 0.052)	0.788 (± 0.055)
Layernorm	0.753 (± 0.056)	0.760 (± 0.052)	0.759 (± 0.051)	0.751 (± 0.049)	0.751 (± 0.056)
CBP	0.834 (± 0.052)	0.847 (± 0.052)	0.846 (± 0.050)	0.847 (± 0.050)	0.857 (± 0.052)
G-mixup	0.866 (± 0.052)	0.881 (± 0.048)	0.885 (± 0.048)	0.880 (± 0.050)	0.879 (± 0.054)

Table 8: ConvNet Test Accuracy Results on Continual ImageNet

Task ($\times 1000$)	0-1	1-2	2-3	3-4	4-5
No Intv.	0.794 (± 0.063)	0.778 (± 0.073)	0.604 (± 0.135)	0.537 (± 0.074)	0.500 (± 0.000)
Retrained	0.857 (± 0.060)	0.851 (± 0.060)	0.850 (± 0.060)	0.849 (± 0.060)	0.846 (± 0.064)
L2 init	0.814 (± 0.047)	0.805 (± 0.050)	0.800 (± 0.052)	0.803 (± 0.052)	0.807 (± 0.054)
Layernorm	0.782 (± 0.052)	0.768 (± 0.056)	0.752 (± 0.049)	0.749 (± 0.050)	0.755 (± 0.055)
CBP	0.848 (± 0.053)	0.867 (± 0.049)	0.864 (± 0.046)	0.863 (± 0.048)	0.878 (± 0.047)
G-mixup	0.875 (± 0.049)	0.896 (± 0.043)	0.899 (± 0.042)	0.894 (± 0.045)	0.896 (± 0.047)

I FTLE Analysis of G-Mixup

We analyze the evolution of the FTLE distribution under Generalized Mixup at different stages of continual training. Our results reveal that G-Mixup consistently maintains the FTLE within a stable and moderate range throughout training. This controlled trajectory of representational dynamics effectively prevents both representation collapse and boundary chaotic behavior—two different types of LoP. The robustness of G-Mixup in sculpting the representation space is thus supported by its capacity to regulate FTLE evolution. The corresponding results are presented in Fig. 8.

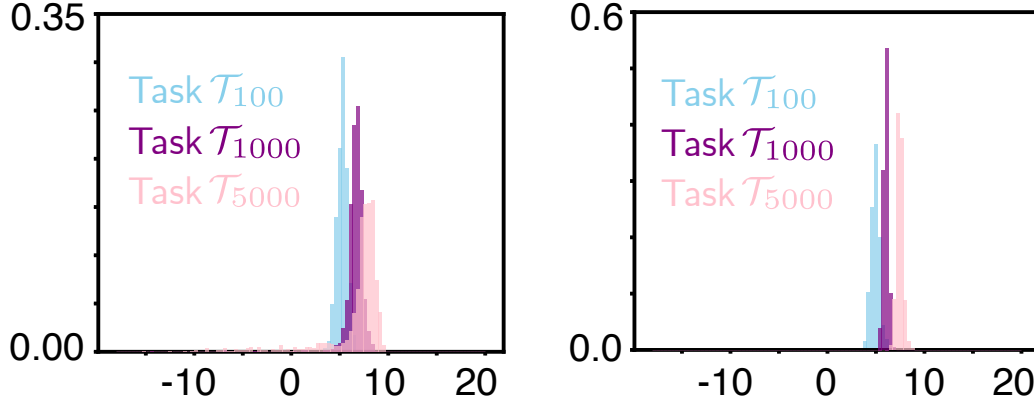


Figure 8: Analysis of FTLE distributions at different training stages of Generalized-mixup.

J Ablation Study

As discussed in the main text, Mixup encourages smoother transitions in the representation space by interpolating inter-class samples and labels, which helps mitigate Type-2 LoP (over-stretch). However, it leaves intra-class labels unchanged (fixed one-hot targets), limiting its ability to prevent

Type-1 LoP (collapse). In contrast, G-Mixup introduces controlled intra-class label variation, thereby regularizing both inter- and intra-class geometry.

To isolate and highlight this intra-class effect, we perform an ablation study in which both Mixup and G-Mixup are restricted to intra-class sample pairs only, removing inter-class interpolation entirely. This setup ensures that both methods generate the same level of sample diversity while differing only in their treatment of intra-class label variation. To accelerate the onset of collapse and better expose representational instability, we slightly increase the learning rate to 0.02.

As shown in Table 9, standard Mixup rapidly leads to collapsed representations, while G-Mixup continues to support progressive adaptation across tasks.

Table 9: Test Accuracy Results for Ablation Study

Task ($\times 500$)	0-1	1-2	2-3	3-4
G-Mixup (SmallConv)	0.864	0.871	0.879	0.878
Mixup (SmallConv)	0.864	0.581	0.500	0.500

K Class-Incremental CIFAR-100

We evaluate all methods under the class-incremental setting of CIFAR-100, where the model progressively learns 100 classes over 20 tasks, with 5 new classes introduced at each task. The model is trained on all accumulated classes at every stage and evaluated on the full set of seen classes, simulating task-agnostic continual learning. Each class contains 600 images, partitioned into 450 for training, 50 for validation, and 100 for testing.

For each increment, the model is trained for 200 epochs using SGD with a momentum of 0.9, weight decay of 0.0005, and a mini-batch size of 100. The learning rate is reset at the beginning of each increment and follows a decaying schedule: 0.1 for the first 60 epochs, 0.02 for the next 60, 0.004 for the following 40, and 0.0008 for the final 40 epochs. For G-Mixup, as the number of classes increases, inter-class interpolations become increasingly dense, making the representation space more entangled and the optimization landscape more complex. To stabilize training under this challenging regime, all learning rates are further reduced by a factor of 0.2 once the number of classes exceeds 10. The same learning-rate schedule is applied to all other baseline methods for fairness. However, under this schedule, their performance deteriorates markedly. Therefore, we present the results obtained under their best-performing configurations. During training, the best-performing model on the validation set is saved at each increment, and training for the next increment starts from this checkpoint, effectively applying early stopping across increments.

We adopt ResNet-18 as the backbone architecture and evaluate two variants: a standard high-dimensional version (ResNet-18) and a compressed low-dimensional variant (0.25 \times ResNet-18), where all channel widths and the final fully connected layer are reduced proportionally. Additionally, a fully connected layer with a width of 16 is appended as the representation layer in the low-dimensional variant. The output layer grows dynamically by adding 5 new units at each increment, with newly added weights initialized via Kaiming initialization and biases set to zero. Convolutional and linear layers follow Kaiming initialization, while batch normalization weights are set to 1.

All input images are normalized to $[0, 1]$, channel-wise standardized using dataset statistics, and augmented during training with random horizontal flips, 4-pixel padding followed by random crops, and random rotations between $[-15^\circ, 15^\circ]$. These augmentations are applied only to the training set.

We compare *G-Mixup* against multiple baselines and LoP mitigation strategies under the class-incremental CIFAR-100 setting. Two primary baselines are considered: (1) **No Intervention (No Intv.)**, which performs standard continual training without any mechanisms to counteract LoP, and (2) **Reinit**, which re-initializes the model from scratch at each task.

To ensure a comprehensive evaluation, we further include representative methods from three major categories of LoP mitigation: **L2 Init** [17], a regularization-based method with a fixed weight decay of 5×10^{-4} and no additional L2 regularization on the weight; **LayerNorm** [21], an architecture-based approach where BatchNorm is replaced with LayerNorm in all layers; and **CBP** [7], a reset-based approach targeting underutilized neurons. For fair comparison, and to rule out the possibility that

G-Mixup’s improvement originates from label softening rather than its geometric regularization effect, all baseline methods are trained with the same soft-label smoothing coefficient of 0.1.

All methods are evaluated under identical training protocols as described above. The comparative results are presented below.

Table 10: 0.25x Resnet-18 Acc. on CIFAR100

Task ($\times 4$)	0-1	1-2	2-3	3-4	4-5
No Intv.	0.927 (± 0.027)	0.812 (± 0.018)	0.754 (± 0.015)	0.708 (± 0.014)	0.661 (± 0.008)
Retrained	0.926 (± 0.035)	0.800 (± 0.020)	0.745 (± 0.013)	0.702 (± 0.012)	0.666 (± 0.007)
L2 init	0.925 (± 0.032)	0.788 (± 0.020)	0.724 (± 0.013)	0.682 (± 0.014)	0.649 (± 0.007)
Layernorm	0.851 (± 0.051)	0.755 (± 0.028)	0.723 (± 0.016)	0.674 (± 0.015)	0.641 (± 0.008)
CBP	0.923 (± 0.038)	0.812 (± 0.020)	0.760 (± 0.017)	0.713 (± 0.011)	0.678 (± 0.006)
G-mixup[ours]	0.932 (± 0.030)	0.825 (± 0.019)	0.772 (± 0.014)	0.732 (± 0.014)	0.697 (± 0.007)

Table 11: Resnet-18 Acc. on CIFAR100

Task ($\times 4$)	0-1	1-2	2-3	3-4	4-5
No Intv.	0.907 (± 0.029)	0.847 (± 0.014)	0.812 (± 0.013)	0.778 (± 0.013)	0.743 (± 0.008)
Retrained	0.901 (± 0.037)	0.842 (± 0.017)	0.815 (± 0.012)	0.788 (± 0.013)	0.767 (± 0.005)
L2 init	0.922 (± 0.027)	0.829 (± 0.016)	0.785 (± 0.013)	0.752 (± 0.012)	0.723 (± 0.006)
Layernorm	0.847 (± 0.065)	0.782 (± 0.035)	0.763 (± 0.019)	0.728 (± 0.019)	0.697 (± 0.019)
CBP	0.907 (± 0.027)	0.847 (± 0.015)	0.816 (± 0.013)	0.788 (± 0.011)	0.768 (± 0.008)
G-mixup[ours]	0.928 (± 0.025)	0.864 (± 0.016)	0.832 (± 0.012)	0.800 (± 0.009)	0.768 (± 0.006)

L Computing Infrastructure

Table 12: Computing infrastructure of the primary server

CPU	AMD EPYC 9654 96-Core Processor, 2 sockets (384 threads total)
GPU	NVIDIA RTX 4090
Memory	512 GB
Operating system	Ubuntu 20.04.6 LTS
Simulation platform	Python 3.11 with PyTorch 2.1.1

Table 13: Computing infrastructure of the secondary server

CPU	AMD EPYC 9354
GPU	NVIDIA RTX 4090
Memory	512 GB
Operating system	Ubuntu 20.04.6 LTS
Simulation platform	Python 3.11 with PyTorch 2.1.1

The simulations and analyses in this study are conducted on two high-performance computing servers equipped with AMD EPYC processors and NVIDIA RTX 4090 GPUs. The primary server features dual AMD EPYC 9654 CPUs (each with 96 cores), totaling 384 threads, and supports simultaneous multithreading. The secondary server is equipped with an AMD EPYC 9354 CPU and identical GPU and software configuration. Both systems are configured with 512 GB of memory, enabling efficient execution of memory-intensive simulations and deep learning workloads. Experiments were run under Ubuntu 20.04.6 LTS using Python 3.11 and PyTorch 2.1.1, providing a robust and up-to-date scientific computing environment.