AdaLMM: Adaptive Layer-wise Model Merging for Continual Learning

Anonymous authors
Paper under double-blind review

Abstract

Replay-based continual learning mitigates catastrophic forgetting by retaining historical examples, but its gains can diminish as the replay memory grows. Parameter-space model merging offers a complementary solution, yet fixed global interpolation coefficients provide limited flexibility in balancing stability and plasticity across the network. To address this limitation, we propose **Ada**ptive **L**ayer-wise **M**odel **M**erging (**AdaLMM**), a task-boundary merging framework for replay-based continual learning. AdaLMM first aligns the historical and updated models through activation-based Optimal Transport matching and architecture-compatible symmetry transformations, and then learns layer-wise or block-wise interpolation coefficients directly from replay data. This allows different parts of the network to retain different proportions of the newly learned update. Experiments on Seq-CIFAR-100 and Seq-Tiny-ImageNet with ResNet-18 and Vision Transformer backbones show that AdaLMM improves the balance between retaining previous knowledge and learning new tasks, leading to higher overall accuracy than standard replay across different memory budgets.

1 Introduction

As machine learning systems are increasingly deployed in dynamic, non-stationary environments, continual learning (CL) has emerged as a critical paradigm that enables models to accumulate knowledge while mitigating catastrophic forgetting (Wang et al., 2024; Kirkpatrick et al., 2017). To navigate the inherent stability-plasticity dilemma, replay-based methods, particularly **Experience Replay (ER)** (Chaudhry et al., 2019; Rolnick et al., 2019), have been widely adopted as a foundational strategy. ER operates by maintaining a limited episodic memory that stores a small subset of representative samples from previous tasks. During training on subsequent tasks, these historical exemplars are interleaved with current data to approximate the overall joint distribution. By constraining the model to optimize the current objective while simultaneously minimizing the loss on the replayed exemplars, ER imposes an explicit gradient-level regularizer. This mechanism helps preserve historical representations, thereby facilitating a practical balance between plasticity for novel concepts and stability for previously acquired knowledge.

Historically, the CL community has predominantly evaluated algorithms under strict memory constraints, often retaining only a minimal subset of past data (Prabhu et al., 2020; Chaudhry et al., 2019). However, this restricted-memory assumption increasingly misaligns with modern machine learning infrastructure. Recent studies emphasize that practical deployments are primarily constrained by computational and training-time budgets rather than storage (Prabhu et al., 2023; Cho et al., 2026; Verghese, 2026). In contemporary architectures, high-performance computing hardware constitutes the primary bottleneck, whereas data storage remains highly scalable and relatively inexpensive (Barth et al., 2025). Transitioning toward an abundant-memory regime, however, does not necessarily yield proportional performance gains for replay-based methods. As the replay buffer grows, historical data are represented more comprehensively during training, which can strengthen stability while making the optimization of newly arriving tasks more challenging. Consequently, the benefit of further increasing the replay budget may diminish, leading to a saturation of overall performance. We refer to this phenomenon as the *plasticity plateau*. As illustrated in Figure 1 (left), continuously enlarging the buffer capacity eventually yields diminishing returns, with the overall accuracy saturating at a suboptimal level. This loss of plasticity may be related to exacerbated gradient interference and an increasingly rugged loss landscape (Wang et al., 2024; Cho et al., 2026).As empirically

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

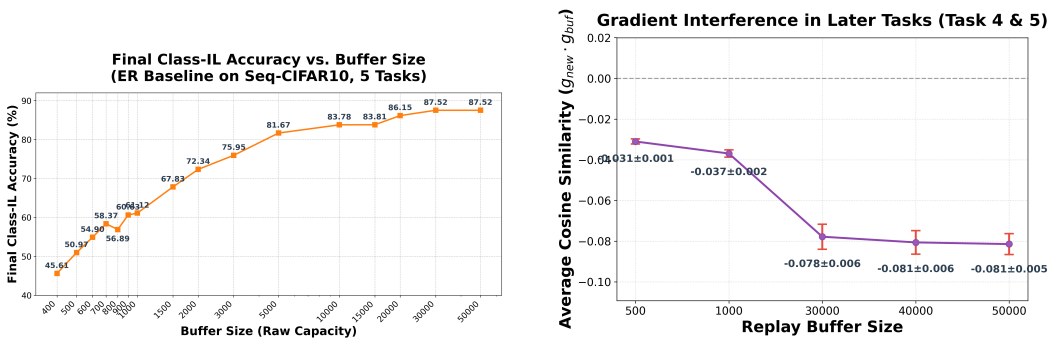

Figure 1: Motivation experiments on Seq-CIFAR-10. **Left:** Final Class-IL accuracy of ER (Chaudhry et al., 2019) across different replay buffer sizes. **Right:** Average cosine similarity between gradients from new-task data (g_{new}) and replay data (g_{buf}).

observed on Seq-CIFAR-10 (Figure 1, right), while the gradients between new and historical data inherently exhibit a negative cosine similarity due to task disparities, expanding the buffer capacity significantly exacerbates this misalignment. This amplified geometric conflict fundamentally restricts the network's capacity to assimilate novel concepts.

To bypass the inherent limitations of data-space replay and address the plasticity plateau, a compelling line of research has shifted focus to directly intervening in parameter space. By resolving permutation symmetries and aligning the hidden units of sequentially trained networks (Ainsworth et al., 2023), model merging can explicitly consolidate learned concepts within a flatter, wider loss basin (Entezari et al., 2022; Wortsman et al., 2022). However, adapting model merging to replay-based continual learning raises two challenges. **First**, although adaptive merging coefficients have been explored in broader multi-task model merging, continual learning requires deciding how much of a sequential task update should be retained at each task boundary using only the information available online. A single fixed coefficient, as commonly used in existing continual-learning interpolation methods, cannot capture heterogeneous responses of different network components to sequential updates. **Second**, integrating model merging under strict total memory budgets introduces a severe storage trade-off. Caching the historical model incurs a substantial memory footprint. Under a strictly aligned memory constraint, this structural overhead significantly diminishes the capacity available for the data replay buffer, often resulting in merging methods underperforming compared to standard ER.

In this paper, we introduce the **Adaptive Layer-wise Model Merge (AdaLMM)** framework, a two-phase parameter-space intervention approach. In the first phase, AdaLMM formulates the structural discrepancy between sequentially trained networks as an Optimal Transport (OT) problem (Kantorovich, 1942; Villani, 2008; Peyré & Cuturi, 2019). By solving for the optimal permutation matrix using activation covariances, we align the cached historical model into the parameter space of the newly updated network, effectively smoothing the interpolation path. In the second phase, AdaLMM replaces the inflexible global scalar with layer-wise learnable vectors. We optimize these merging coefficients by backpropagating the replay loss computed on a proxy memory buffer. This differentiable merging phase allows different structural components to adopt different stability–plasticity trade-offs according to the replay objective, without imposing a manually designed depth-dependent schedule.

The main contributions of our work are summarized as follows:

- **Layer-wise Differentiable Merging via Geometric Alignment:** We propose AdaLMM, a task-boundary model-merging framework that unifies geometric mode connectivity with adaptive knowledge dynamics. It effectively addresses the limitations of uniform global scalars by autonomously optimizing layer-specific merging coefficients to mitigate catastrophic forgetting.
- **Plug-and-Play Scalability to Sequential Tasks:** Operating post-hoc to the task-specific training phase, AdaLMM serves as a universal, plug-and-play module that integrates seamlessly with standard rehearsal algorithms. Furthermore, our framework naturally supports progressive task expansion with a modest computational overhead across both the alignment and merging phases.

- **Consistent Improvements across Architectures and Replay Budgets:** Experiments with ResNet-18 and Vision Transformer backbones on Seq-CIFAR-100 and Seq-Tiny-ImageNet show that AdaLMM improves several representative rehearsal baselines under memory-aligned evaluation, while introducing no additional inference-time cost.

2 PRELIMINARIES

Due to the space limit, we leave the detailed introduction on related works to Appendix A. In this section, we briefly introduce some important concepts and formulations for our problem.

2.1 CONTINUAL LEARNING AND EXPERIENCE REPLAY

We consider a standard continual learning setup where a model learns from a sequence of N tasks, denoted as $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$. Each task T_t provides a training dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{n_t}$ drawn from an underlying, non-stationary data distribution $P_t(X, Y)$. A shared classifier predicts over all seen tasks without task identity. We write the predictor as $f(x; \theta, s)$, distinguishing learnable parameters $\theta \in \mathbb{R}^d$ from non-learnable state s, such as Batch Normalization running statistics. The primary objective is to update f sequentially across tasks without catastrophic forgetting of previous distributions $P_{1:t-1}$.

To mitigate forgetting, Experience Replay (ER) maintains a memory buffer $\mathcal{M}$ with a strictly bounded capacity M. During the training of task T_t, the model optimizes the joint empirical risk over both the current task data and the buffered historical samples. The ER objective function is formulated as:

$$\mathcal{L}_{\mathrm{ER}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(f(x;\theta,s),y)] + \mathbb{E}_{(x,y)\sim\mathcal{M}}[\ell(f(x;\theta,s),y)]\,, \tag{1}$$

where $\ell(\cdot,\cdot)$ denotes the standard cross-entropy loss. The efficacy of ER is sensitive to the buffer capacity M: small memories can leave the model susceptible to catastrophic forgetting, whereas further enlarging the memory does not necessarily yield proportional performance gains. In the abundant-memory regime studied here, we empirically observe diminishing returns together with increased conflict between current-task and replay gradients, motivating complementary intervention in parameter space.

2.2 MODEL MERGING IN PARAMETER SPACE

To address the plasticity bottlenecks inherent to purely data-driven replay, recent research has turned toward direct intervention in parameter space, drawing upon the methodologies from the broader model merging literature (Wortsman et al., 2022; Ilharco et al., 2022; Ainsworth et al., 2023). Let $\theta_{hist} \in \mathbb{R}^d$ denote the cached historical model immediately after convergence on task T_{t-1}, and let $\theta_{new} \in \mathbb{R}^d$ represent the updated model that incorporates the newly learned task knowledge via gradient descent on $\mathcal{D}_t \cup \mathcal{M}$. Standard model merging techniques attempt to consolidate knowledge by linearly interpolating between these two states using a global merging coefficient $\alpha \in [0, 1]$:

$$\theta_{\mathrm{merged}} = (1-\alpha)\theta_{\mathrm{new}} + \alpha\theta_{\mathrm{hist}}, \qquad \alpha \in [0,1]. \tag{2}$$

Despite its computational efficiency, this standard formulation presents two fundamental limitations. Geometrically, due to the permutation symmetries of hidden units, θ_{new} and θ_{hist} may diverge into disparate coordinate systems; direct interpolation across these misaligned weights frequently crosses high-loss barriers. Structurally, applying a globally fixed scalar enforces an inflexible, layer-agnostic merging ratio. This contradicts the hierarchical dynamics of deep neural networks, where shallow and deep layers demand divergent stability-plasticity trade-offs. Our proposed AdaLMM framework is specifically designed to bypass these two bottlenecks.

2.3 OPTIMAL TRANSPORT

Optimal Transport (OT) provides an elegant mathematical framework for finding the most cost-efficient plan to transform one probability distribution into another (Kantorovich, 1942; Villani, 2008; Peyré & Cuturi, 2019). In the context of parameter-space intervention, we focus on the discrete formulation of OT. Consider two discrete probability distributions $\mu = \sum_{i=1}^{n} \mathbf{p}_i \delta_{u_i}$ and $\nu = \sum_{j=1}^{m} \mathbf{q}_j \delta_{v_j}$, where u_i and v_j represent sample points in the source and target distributions, $\mathbf{p} \in \mathbb{R}_+^n$ and $\mathbf{q} \in \mathbb{R}_+^m$ are the probability weight vectors satisfying $\sum_{i=1}^{n} \mathbf{p}_i = \sum_{j=1}^{m} \mathbf{q}_j = 1$, and δ denotes the Dirac delta function, indicating that the probability mass is discretely concentrated at specific sample points.

Let $C \in \mathbb{R}^{n \times m}$ be the cost matrix, where the element $C_{i,j}$ defines the penalty of moving a unit of mass from u_i to v_j. The discrete Kantorovich formulation of OT seeks a transport plan $P \in \mathbb{R}_+^{n \times m}$ that establishes the optimal correspondence by minimizing the total transportation cost:

$$\min_{P \in \Pi(\mathbf{p}, \mathbf{q})} \langle P, C \rangle_F = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{i,j} C_{i,j} \tag{3}$$

where $\Pi(\mathbf{p}, \mathbf{q}) = \{P \in \mathbb{R}_+^{n \times m} \mid P\mathbf{1}_m = \mathbf{p}, P^\top \mathbf{1}_n = \mathbf{q}\}$ is the set of all valid coupling matrices satisfying the marginal distribution constraints, and $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product.

3 Proposed Method: Adaptive Layer-wise Model Merge (AdaLMM)

In this section, we introduce AdaLMM, a two-stage parameter-space consolidation framework for replay-based continual learning. We first formulate the alignment between the historical and updated model as an activation-based optimal transport problem, together with architecture-compatible symmetry transformations, to reduce coordinate mismatch before interpolation (Section 3.1). We then introduce a replay-driven differentiable merging stage that learns layer-wise or block-wise interpolation coefficients, allowing different parts of the network to retain different fractions of the current task update (Section 3.2). Finally, the merged model is equipped with the appropriate normalization state and carried forward for inference and subsequent learning. The overall framework is illustrated in Figure 2, and Algorithm 1 summarizes the full procedure.

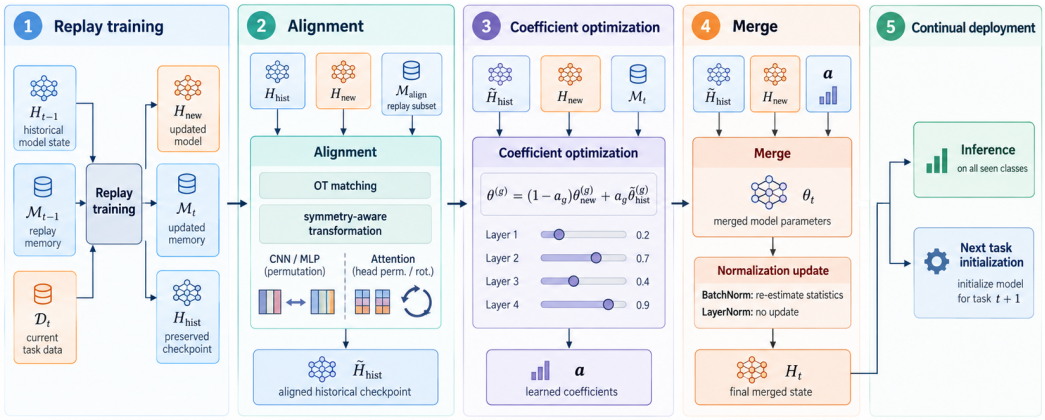

Figure 2: **Overview of AdaLMM framework.** A historical model is saved before replay training. At the task boundary, it is aligned to the updated model, and replay examples are used to learn layer-wise or block-wise interpolation coefficients. The merged model is deployed after normalization-statistics handling and becomes the starting point for the next task.

3.1 Phase 1: Architecture-Aware Alignment

Direct weight interpolation implicitly assumes that corresponding parameters in the two endpoint models refer to compatible hidden representations. This assumption need not hold exactly because neural networks admit function-preserving parameter symmetries: hidden units can be permuted, and attention modules admit additional symmetry transformations, without changing the represented function (Singh & Jaggi, 2020; Ainsworth et al., 2023; Zhang et al., 2025). Interpolating two functionally related models in mismatched coordinates can therefore mix unrelated hidden units. To reduce this ambiguity, AdaLMM first aligns the historical model to the updated model while keeping the latter fixed as the reference.

Activation-based OT matching. A useful correspondence should reflect what a hidden unit represents, therefore we estimate unit similarity from activations on replay examples. Let $\mathcal{M}_{\text{align}} \subseteq \mathcal{M}_t$ be a replay subset used for alignment. At a matching site l with K_l units, evaluating both models on the same inputs gives activation matrices

$$A^{(l)}_{\text{hist}}, A^{(l)}_{\text{new}} \in \mathbb{R}^{n_l \times K_l}, \tag{4}$$

where each column corresponds to one hidden unit and n_l denotes the number of activation samples used to estimate the correlation. For convolutional features, spatial locations are treated as separate samples when computing the correlation. Following correlation-based neuron matching (Jordan et al., 2023), we define the pairwise cost

$$C_{ij}^{(l)} = 1 - \frac{\mathrm{Cov}\Big(A_{\mathrm{hist},:i}^{(l)}, A_{\mathrm{new},:j}^{(l)}\Big)}{\mathrm{std}\Big(A_{\mathrm{hist},:i}^{(l)}\Big)\,\mathrm{std}\Big(A_{\mathrm{new},:j}^{(l)}\Big) + \varepsilon}, \tag{5}$$

so that units with similar normalized activation patterns receive lower matching cost.

Using $C^{(l)}$, we instantiate the discrete OT formulation introduced in Section 2.3. For equal-width layers, we use uniform marginals $\mathbf{p}^{(l)} = \mathbf{q}^{(l)} = \mathbf{1}/K_l$ and solve

$$T_l^\star \in \arg\min_{T \in \Pi(\mathbf{p}^{(l)}, \mathbf{q}^{(l)})} \langle T, C^{(l)} \rangle_F. \tag{6}$$

The OT problem may be solved with a hard assignment method such as the Hungarian algorithm (Kuhn, 1955), or with entropy-regularized transport using Sinkhorn iterations (Cuturi, 2013). In our hard-alignment setting, the resulting correspondence is applied as a function-preserving reparameterization of the historical model.

For fully connected, convolutional, and feed-forward layers, this reparameterization corresponds to permuting hidden units or channels and propagating the same permutation to connected parameters. Bias terms, normalization parameters, and interacting residual branches are transformed consistently. These coupled transformations change the parameter coordinates of the historical checkpoint without changing the function it represents; detailed transformation rules are given in Appendix G.

Attention modules. Transformer attention introduces additional symmetries beyond unit permutations. Following prior work on Transformer fusion and rotation-based matching (Imfeld et al., 2024; Zhang et al., 2025), attention heads can be permuted together with their associated query, key, value, and output-projection blocks. Within a matched head, self-attention also admits compensated orthogonal transformations

$$Q' = QR, \qquad K' = KR, \qquad V' = VS, \qquad W_O' = S^\top W_O, \tag{7}$$

where $R^\top R = I$ and $S^\top S = I$. These transformations preserve the query–key scores and the value–output mapping, allowing the historical attention subspaces to be aligned without changing the represented attention function. Continuous rotations are restricted to these linear attention subspaces; nonlinear feed-forward layers are handled through permutations, while residual-stream and LayerNorm coordinates remain unchanged. We denote the aligned historical parameters by $\widetilde{\theta}_{\mathrm{hist}}$.

3.2 Phase 2: Replay-Based Layer-wise Merging

Alignment makes the two models more compatible for interpolation, but it does not determine how much of the historical model should be retained. A single global coefficient imposes the same interpolation ratio throughout the network, although different layers or blocks can respond differently to sequential updates. AdaLMM therefore assigns an independent coefficient to each predefined layer or architectural block and learns these coefficients from replay data.

From a global ratio to adaptive coefficients. Let $\{\mathcal{I}_l\}_{l=1}^{L}$ be a partition of the learnable parameters, where all parameters in $\mathcal{I}_l$ share one coefficient α_l. For component l, the merged parameters are

$$\theta_{\mathrm{merged}}^{(l)}(\alpha_l) = (1-\alpha_l)\theta_{\mathrm{new}}^{(l)} + \alpha_l \widetilde{\theta}_{\mathrm{hist}}^{(l)}, \qquad \alpha_l \in [0,1]. \tag{8}$$

Here, α_l is the interpolation coefficient that determines the contribution of the historical model to component l. When alignment is the identity, the same expression can be written as

$$\theta_{\mathrm{merged}}^{(l)} = \theta_{\mathrm{hist}}^{(l)} + (1-\alpha_l)\Delta_t^{(l)}, \qquad \Delta_t^{(l)} = \theta_{\mathrm{new}}^{(l)} - \theta_{\mathrm{hist}}^{(l)}. \tag{9}$$

This form gives a more direct interpretation of the coefficient: $1-\alpha_l$ is the fraction of the net task update retained in component l. Thus, learning different α_l values amounts to calibrating how strongly different parts of the network inherit the current task update, rather than prescribing one ratio for the entire model.

Algorithm 1 Adaptive Layer-wise Model Merging (AdaLMM)

Input: Tasks $\{\mathcal{D}_t\}_{t=1}^{N}$, initial state H_0, bounded replay memory $\mathcal{M}_0$, parameter partition $\{\mathcal{I}_l\}_{l=1}^{L}$, coefficient-fitting epochs E_{merge}
Output: Final model state H_N

for $t = 1, \ldots, N$ **do**
  $H_{\text{hist}} = (\theta_{\text{hist}}, s_{\text{hist}}) \leftarrow \text{Copy}(H_{t-1})$
  $(H_{\text{new}}, \mathcal{M}_t) \leftarrow \text{ReplayTrain}(\text{Copy}(H_{t-1}), \mathcal{D}_t, \mathcal{M}_{t-1})$
  Let $H_{\text{new}} = (\theta_{\text{new}}, s_{\text{new}})$
  if $t = 1$ **then**
    $H_t \leftarrow H_{\text{new}}$
    continue
  end if
  Select replay support $\mathcal{M}_{\text{align}} \subseteq \mathcal{M}_t$
  $\widetilde{H}_{\text{hist}} \leftarrow \text{Align}(H_{\text{hist}}, H_{\text{new}}; \mathcal{M}_{\text{align}})$
  Let $\widetilde{H}_{\text{hist}} = (\widetilde{\theta}_{\text{hist}}, \widetilde{s}_{\text{hist}})$
  Define $\theta_{\text{merged}}(\boldsymbol{\alpha})$ by Eq. (8)
  Choose and fix normalization state s_{fit} according to the normalization policy
  Initialize coefficient variables $\mathbf{w}$
  for $e = 1, \ldots, E_{\text{merge}}$ **do**
    for minibatch $\mathcal{B}$ from $\mathcal{M}_t$ **do**
      $\boldsymbol{\alpha} \leftarrow \sigma(\mathbf{w})$
      Compute a minibatch estimate of $\mathcal{L}_{\mathcal{M}_t}$ using $\mathcal{B}$
      Update only $\mathbf{w}$ using $\nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{M}_t}$ /* *Eq.* (11) */
    end for
  end for
  $\theta_t \leftarrow \theta_{\text{merged}}(\sigma(\mathbf{w}))$
  $s_t \leftarrow s_{\text{new}}$
  if Batch Normalization is used **then**
    Re-estimate BN running statistics in s_t using replay data
  end if
  $H_t \leftarrow (\theta_t, s_t)$
end for
return H_N

Learning the coefficients from replay. The replay memory already contains information from previously observed tasks and the current task, making it a natural signal for selecting the interpolation point without introducing an additional validation set.We parameterize $\alpha_l = \sigma(w_l)$, where $\sigma(\cdot)$ is the sigmoid function that maps the unconstrained variable $w_l \in \mathbb{R}$ to a valid interpolation coefficient $\alpha_l \in (0, 1)$. We then optimize $\mathbf{w} = [w_1, \ldots, w_L]^\top$ by minimizing the empirical replay loss:

$$\mathcal{L}_{\mathcal{M}_t}(\mathbf{w}) = \frac{1}{|\mathcal{M}_t|} \sum_{(x,y)\in\mathcal{M}_t} \ell(f(x; \theta_{\text{merged}}(\mathbf{w}), s), y), \qquad \alpha_l = \sigma(w_l), \tag{10}$$

where the endpoint parameters, their alignment, and the normalization state s are held fixed during coefficient fitting. Optimization therefore adjusts only the interpolation coefficients, leaving the underlying replay learner unchanged.

The chain rule gives

$$\frac{\partial \mathcal{L}_{\mathcal{M}_t}}{\partial w_l} = \alpha_l(1-\alpha_l) \left\langle \nabla_{\theta^{(l)}} \mathcal{L}_{\mathcal{M}_t}, \widetilde{\theta}^{(l)}_{\text{hist}} - \theta^{(l)}_{\text{new}} \right\rangle, \tag{11}$$

with the inner product taken over all parameters assigned to component l. The term inside the inner product is the directional derivative of the replay loss from the updated checkpoint toward the aligned historical checkpoint. A positive value pushes α_l downward and retains more of the current update, whereas a negative value pushes α_l upward and increases the historical contribution. Appendix I provides the full derivation.

Why allow different coefficients? Layer-wise or block-wise coefficients enlarge the admissible interpolation space by allowing different parts of the network to adjust their interpolation ratios

independently. If $J(\boldsymbol{\alpha})$ denotes the replay objective expressed directly in the coefficients, a shared coefficient restricts the search to $\boldsymbol{\alpha} = c\mathbf{1}$, whose directional derivative is $\mathbf{1}^\top \nabla_{\boldsymbol{\alpha}} J$. Contributions from different components can therefore cancel along the global direction even when the full coefficient-space gradient is nonzero. Allowing the coefficients to vary independently can expose a local descent direction that is unavailable to global interpolation; a formal local analysis is given in Appendix H.

3.3 Normalization and Continual Deployment

Weight interpolation changes the network parameters and can consequently alter the activation statistics of the merged model. This is particularly relevant for Batch Normalization (BN), whose running moments are stored as state variables rather than optimized together with the interpolation coefficients (Ioffe & Szegedy, 2015; Jordan et al., 2023). We therefore keep the BN running statistics fixed during coefficient optimization and re-estimate them from replay data after the final merged parameters θ_t are obtained, so that the normalization state is consistent with the merged network. The learnable BN affine parameters remain part of the interpolation.

LayerNorm does not maintain running statistics and therefore requires no such recalibration; its affine parameters are interpolated together with the other learnable parameters (Ba et al., 2016). The resulting model state $H_t = (\theta_t, s_t)$ is then used for inference over all seen classes and serves as the initialization for task $t + 1$.

4 Experiments

We evaluate the empirical performance and computational efficiency of the proposed AdaLMM framework across various exemplar memory budgets.

4.1 Experimental Setup

Datasets and Scenarios. We evaluate our method under the rigorous Class-Incremental Learning (Class-IL) paradigm, where task identities are strictly unavailable during inference. We utilize two standard benchmarks: Seq-CIFAR-100 (Krizhevsky et al., 2009) (10 tasks, 10 classes per task) and Seq-Tiny-ImageNet (Le & Yang, 2015) (20 tasks, 10 classes per task). Detailed dataset statistics and preprocessing pipelines are deferred to Appendix C.

Evaluation Metrics. We measure performance using two standard continual learning metrics (Chaudhry et al., 2018): **Average Accuracy (A_N)**, which represents the mean test accuracy across all N tasks after training on the final task (higher is better), and **Average Forgetting (F_N)**, which denotes the average accuracy drop for each task from its peak performance to its state at the end of the training stream (lower is better).

4.2 Baselines and Memory-Aligned Evaluation

To rigorously evaluate the efficacy of AdaLMM, we benchmark our framework against representative continual learning methodologies, encompassing regularization-based methods (oEWC (Schwarz et al., 2018), SI (Zenke et al., 2017)), prominent rehearsal strategies (ER (Chaudhry et al., 2019), DER++ (Buzzega et al., 2020), and ER-ACE (Caccia et al., 2022)), and weight interpolation approach, CLEWI (Kozal et al., 2024). Because AdaLMM operates strictly post-hoc to the gradient descent phase, it functions independently of the underlying buffer sampling mechanics and loss formulations. Consequently, it serves as an orthogonal plug-in module. We specifically evaluate its integration with three prominent rehearsal strategies, denoted as **AdaLMM+ER**, **AdaLMM+DER++**, and **AdaLMM+ER-ACE**.

To establish empirical performance bounds, we include two reference baselines: **Joint Training** (offline optimization on the cumulative dataset, serving as the upper bound) and **Naive SGD** (standard sequential optimization without forgetting mitigation, serving as the lower bound).

Strict Memory Alignment. To ensure an equitable comparison, all evaluated methods are constrained to a strictly identical total memory footprint. Because AdaLMM necessitates caching the historical model θ_{hist}, we directly deduct the equivalent storage overhead of these parameters from its allotted exemplar buffer capacity. Consequently, AdaLMM operates with a reduced number of rehearsed samples compared to standard baselines to exactly offset this structural memory cost. Details of this memory equivalence are in Appendix F.

Table 1: **Main results with ResNet-18.** We report Average Accuracy ($A_N \uparrow$) and Average Forgetting ($F_N \downarrow$) after the final task. All methods are evaluated under the strictly memory-aligned setting. "±" denotes standard deviation across 5 random seeds.

Method	Seq-CIFAR-100 (10 Tasks)		Seq-Tiny-ImageNet (20 Tasks)	
	$A_N(\%) \uparrow$	$F_N(\%) \downarrow$	$A_N(\%) \uparrow$	$F_N(\%) \downarrow$
Joint Training	70.70 ± 0.32	–	57.07 ± 0.35	–
Naive SGD	8.80 ± 0.45	89.37 ± 0.94	4.00 ± 0.22	78.15 ± 1.88
EWC-On	8.47 ± 0.57	86.72 ± 0.96	3.76 ± 0.12	74.78 ± 3.28
SI	8.67 ± 0.43	88.01 ± 1.08	3.57 ± 0.22	68.15 ± 3.90
Buffer Capacity: 60% of Historical Data (30,000 for Seq-CIFAR-100, 60,000 for Seq-Tiny-ImageNet)				
ER	65.78 ± 0.64	17.39 ± 0.43	44.68 ± 0.86	25.27 ± 0.66
+ CLeWI	65.14 ± 0.49	1.61 ± 0.69	46.92 ± 1.24	9.16 ± 0.57
+ **AdaLMM** (Ours)	67.18 ± 0.17 (+1.40)	8.38 ± 0.29 (-9.01)	47.46 ± 0.91 (+2.78)	12.26 ± 0.64 (-13.01)
DER++	67.64 ± 0.60	14.09 ± 1.00	48.32 ± 3.40	19.21 ± 3.22
+ CLeWI	66.85 ± 0.74	1.84 ± 0.67	51.03 ± 1.05	$\mathbf{5.91 \pm 0.83}$
+ **AdaLMM** (Ours)	$\mathbf{68.10 \pm 0.48}$ (+0.46)	4.60 ± 0.81 (-9.49)	$\mathbf{51.14 \pm 1.98}$ (+2.82)	7.46 ± 1.54 (-11.75)
ER-ACE	66.16 ± 0.54	9.44 ± 0.68	44.89 ± 0.80	13.75 ± 0.76
+ CLeWI	62.08 ± 0.71	$\mathbf{-2.01 \pm 0.29}$	44.87 ± 0.72	7.17 ± 0.67
+ **AdaLMM** (Ours)	65.88 ± 0.64 (-0.28)	6.51 ± 0.34 (-2.93)	46.93 ± 1.01 (+2.04)	10.48 ± 0.36 (-3.27)
Buffer Capacity: 80% of Historical Data (40,000 for Seq-CIFAR-100, 80,000 for Seq-Tiny-ImageNet)				
ER	66.88 ± 0.30	15.77 ± 0.64	45.61 ± 1.10	23.12 ± 0.32
+ CLeWI	65.95 ± 0.39	0.62 ± 0.56	47.39 ± 1.40	8.01 ± 0.73
+ **AdaLMM** (Ours)	67.81 ± 0.56 (+0.93)	7.23 ± 0.25 (-8.54)	48.51 ± 0.65 (+2.90)	10.72 ± 0.25 (-12.40)
DER++	68.49 ± 0.43	12.99 ± 1.34	48.31 ± 3.67	19.28 ± 3.71
+ CLeWI	67.35 ± 0.63	1.26 ± 0.51	50.52 ± 1.24	7.20 ± 1.10
+ **AdaLMM** (Ours)	$\mathbf{68.54 \pm 0.49}$ (+0.05)	3.07 ± 1.14 (-9.92)	$\mathbf{52.52 \pm 0.62}$ (+4.21)	$\mathbf{5.63 \pm 0.74}$ (-13.65)
ER-ACE	66.92 ± 0.57	8.26 ± 0.74	45.45 ± 0.88	12.79 ± 0.82
+ CLeWI	63.19 ± 0.73	$\mathbf{-2.82 \pm 0.28}$	45.75 ± 0.64	6.53 ± 0.62
+ **AdaLMM** (Ours)	67.16 ± 0.29 (+0.24)	5.74 ± 0.52 (-2.52)	47.49 ± 0.84 (+2.04)	9.75 ± 1.08 (-3.04)

4.3 Main Results

We evaluate AdaLMM on Seq-CIFAR-100 and Seq-Tiny-ImageNet with both convolutional and Transformer backbones. Table 1 reports the results with ResNet-18, while Table 2 evaluates AdaLMM with a Vision Transformer backbone.

Results with ResNet-18. AdaLMM improves the underlying rehearsal methods across replay budgets on both benchmarks. For example, with an 80% buffer on Seq-Tiny-ImageNet, AdaLMM increases DER++ accuracy from 48.31% to 52.52% while reducing forgetting from 19.28% to 5.63%. CLeWI often achieves lower forgetting through stronger historical weighting, but this can come with lower final accuracy; AdaLMM generally provides a better balance between the two metrics.

Table 2: **Main results with a Vision Transformer backbone.** A_N (↑) and F_N (↓) are reported (%).

Method	Buffer	$A_N \uparrow$	$F_N \downarrow$
Seq-CIFAR-100			
Joint Training	–	88.54 ± 0.32	–
ER	60%	84.23 ± 0.27	11.40 ± 0.41
CLeWI	60%	85.29 ± 0.06	6.70 ± 0.09
AdaLMM	60%	$\mathbf{86.40 \pm 0.25}$ (+2.17)	$\mathbf{6.27 \pm 0.04}$ (-5.13)
ER	80%	84.64 ± 0.28	10.88 ± 0.37
CLeWI	80%	85.24 ± 0.06	6.57 ± 0.06
AdaLMM	80%	$\mathbf{86.60 \pm 0.11}$ (+1.96)	$\mathbf{5.79 \pm 0.35}$ (-5.09)
Seq-Tiny-ImageNet			
Joint Training	–	85.12 ± 0.30	–
ER	60%	79.79 ± 0.21	13.67 ± 0.14
CLeWI	60%	81.41 ± 0.14	8.36 ± 0.32
AdaLMM	60%	$\mathbf{83.94 \pm 0.15}$ (+4.15)	$\mathbf{6.34 \pm 0.18}$ (-7.33)
ER	80%	80.28 ± 0.11	13.04 ± 0.29
CLeWI	80%	81.53 ± 0.13	7.98 ± 0.03
AdaLMM	80%	$\mathbf{84.11 \pm 0.12}$ (+3.83)	$\mathbf{5.91 \pm 0.24}$ (-7.13)

Results with Vision Transformers. AdaLMM also consistently improves both ER and CLeWI with the Transformer backbone. The gains are particularly clear on Seq-Tiny-ImageNet, where AdaLMM improves ER by 4.15 and 3.83 percentage points in accuracy under the 60% and 80% buffers, while reducing forgetting by 7.33 and 7.13 points, respectively. These results show that the benefit of adaptive merging extends beyond convolutional architectures.

4.4 Ablation Study

We ablate AdaLMM's core components by evaluating two degraded variants (Table 3): *w/o OT* (omitting geometric alignment) and *Global-*α (replacing layer-wise vectors with a globally optimized scalar using the identical proxy loss framework).

Table 3: Ablation study on the core components of AdaLMM. A_N (↑) and F_N (↓) are reported (%).

Method	60% Buffer Ratio		80% Buffer Ratio	
	A_N (↑)	F_N (↓)	A_N (↑)	F_N (↓)
Seq-CIFAR-100				
w/o OT	$66.72_{\pm 0.23}$	$8.45_{\pm 0.49}$	$67.01_{\pm 0.33}$	$7.74_{\pm 0.55}$
Global-α	$66.97_{\pm 0.13}$	$8.84_{\pm 0.16}$	$67.54_{\pm 0.55}$	$7.71_{\pm 0.96}$
AdaLMM	$\mathbf{67.18}_{\pm 0.17}$	$\mathbf{8.38}_{\pm 0.29}$	$\mathbf{67.81}_{\pm 0.56}$	$\mathbf{7.23}_{\pm 0.25}$
Seq-Tiny-ImageNet				
w/o OT	$46.89_{\pm 1.11}$	$14.32_{\pm 0.28}$	$47.12_{\pm 1.90}$	$13.46_{\pm 1.38}$
Global-α	$47.01_{\pm 0.53}$	$14.03_{\pm 0.49}$	$48.08_{\pm 0.57}$	$12.78_{\pm 0.73}$
AdaLMM	$\mathbf{47.46}_{\pm 0.91}$	$\mathbf{12.26}_{\pm 0.64}$	$\mathbf{48.51}_{\pm 0.65}$	$\mathbf{10.72}_{\pm 0.25}$

Effect of OT Alignment. Removing the alignment stage increases forgetting, particularly on Seq-Tiny-ImageNet (e.g., F_N increases from 12.26% to 14.32% at the 60% buffer setting). This result shows that accounting for possible coordinate mismatch before interpolation can improve the merged model, especially in the more challenging setting.

Effect of Layer-wise Coefficients. Using a single learned coefficient (*Global-*α) consistently yields lower accuracy than AdaLMM under the same replay-based optimization objective. This result indicates that constraining all layers to share the same interpolation ratio can limit the flexibility of model merging. Allowing layer-wise coefficients enlarges the interpolation space and enables different parts of the network to respond independently to the replay objective. We further visualize the learned coefficients across network depth and sequential tasks in Appendix J.

4.5 Analysis of Computational Overhead

We evaluate the computational overhead of AdaLMM by measuring the total training time over the complete continual learning sequence. As shown in Figure 3, AdaLMM introduces an additional 7.5%–19.4% training cost with DeiT-S/16 across the evaluated datasets and memory budgets. The additional computation comes from architecture-aware alignment and replay-based coefficient optimization, while the standard rehearsal training remains unchanged. Results with ResNet-18 are provided in Appendix L. Overall, AdaLMM incurs moderate task-boundary computation without additional inference-time cost, since only the final merged model is deployed.

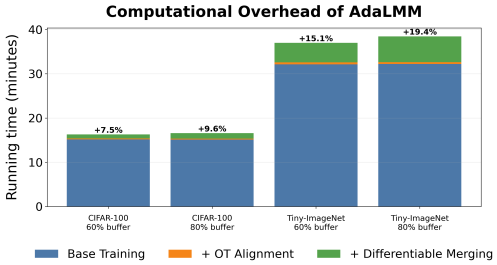

Figure 3: Computational overhead of AdaLMM with DeiT-S/16. Percentages denote the additional execution time relative to the corresponding ER baseline.

5 Conclusion

In this paper, we introduced AdaLMM, a task-boundary model merging framework for replay-based continual learning. AdaLMM first aligns the historical and updated models through architecture-aware matching, and then learns layer-wise interpolation coefficients from replay data, allowing different parts of the network to retain different proportions of the current task update while preserving a single deployable model. Experiments on Seq-CIFAR-100 and Seq-Tiny-ImageNet with both ResNet-18 and Vision Transformer backbones show that AdaLMM improves the balance between retaining previous knowledge and learning new tasks, resulting in higher overall accuracy across the tasks learned so far. Future work will investigate more efficient alignment strategies and replay-efficient coefficient optimization for larger models and longer task sequences.

AI Use Statement

Generative AI tools were used to assist with language editing, organization and presentation of the manuscript, and the preparation of scientific figures. Codex was additionally used to assist with experimental execution, code inspection, and result organization. All AI-assisted outputs, experimental results, analyses, and manuscript content were reviewed and verified by the authors, who take full responsibility for the final work.

REFERENCES

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

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

Adam Barth et al. The Data Center Dividend: How US States Can Capitalize on the AI Boom. McKinsey & Company Featured Insights, Oct 2025. https://www.mckinsey.com/featured-insights/week-in-charts/the-data-center-dividend.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in neural information processing systems*, volume 33, pp. 15920–15930, 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

Dongkyu Cho, Taesup Moon, Rumi Chunara, Kyunghyun Cho, and Sungmin Cha. Forget forgetting: Continual learning in a world of abundant memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fvL8IIEPxG.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

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*, pp. 770–778, 2016.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Colin Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, et al. Patching open-vocabulary models by interpolating weights. In *Advances in Neural Information Processing Systems*, 2022.

Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. Transformer fusion with optimal transport. In *International Conference on Learning Representations*, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, 2015.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gU5sJ6ZggcX.

Leonid Vitalievich Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk SSSR*, volume 37, pp. 199–201, 1942.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Jędrzej Kozal, Jan Wasilewski, Bartosz Krawczyk, and Michał Woźniak. Continual learning with weight interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4187–4195, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24246–24255, 2023.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 379–395, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73013-9.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5–6):355–607, February 2019. ISSN 1935-8237. doi: 10.1561/2200000073. URL https://doi.org/10.1561/2200000073.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019. doi: 10.1561/2200000073.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 524–540, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58536-5.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3698–3707, 2023.

Zihuan Qiu, Yi Xu, Chiyuan He, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. MINGLE: Mixture of null-space gated low-rank experts for test-time continual model merging. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8DCyv8x58O.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. *Experience replay for continual learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22045–22055. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb2697869f56484404c8ceee2985b01d-Paper.pdf.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021.

Akhil Verghese. Is AI compute becoming the next bottleneck? *RTInsights*, Apr 2026. URL https://www.rtinsights.com/is-ai-compute-becoming-the-next-bottleneck/.

Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

Haitao Wen, Lili Pan, Yu Dai, Heqian Qiu, Lanxiao Wang, Qingbo Wu, and Hongliang Li. Class incremental learning with multi-teacher distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28443–28452. IEEE Computer Society, 2024.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Michał Zając, Tinne Tuytelaars, and Gido M van de Ven. Prediction error-based classification for class-incremental learning. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

Han Zang, Yongfeng Dong, Linhao Li, Liang Yang, and Yu Wang. Topology-aware knowledge preservation for class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 28059–28067, 2026.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77090–77106. PMLR, 2025.

A Extended Related Work

Continual Learning. To mitigate catastrophic forgetting in Class-Incremental Learning, regularization-based approaches constrain updates to parameters that are important for previously learned tasks, e.g., through Bayesian penalties such as oEWC (Schwarz et al., 2018) or path-integral based importance estimates such as SI (Zenke et al., 2017). Rehearsal-based methods instead retain a subset of previously observed examples and jointly train on current and replayed data. Standard Experience Replay (ER) (Chaudhry et al., 2019) interleaves buffered samples with current-task examples, while subsequent methods improve rehearsal through nearest-mean classification (Rebuffi et al., 2017), knowledge distillation (Buzzega et al., 2020), asymmetric objectives (Caccia et al., 2022), contrastive proxies (Lin et al., 2023), prediction-error based classification (Zając et al., 2024), multi-teacher distillation (Wen et al., 2024), and topology-aware knowledge preservation (Zang et al., 2026).

Recent work has also revisited continual learning under an *abundant-memory* regime, where storage is less restrictive and the dominant challenge can shift from retaining historical information toward maintaining sufficient plasticity for learning new tasks. In particular, Cho et al. (2026) show that increasing replay memory does not necessarily lead to monotonic improvements and propose Weight Space Consolidation, which restores plasticity through parameter-space interventions followed by weight averaging. This perspective is closely related to our motivation. In contrast, AdaLMM focuses on adaptively selecting how much of the task-induced parameter update should be retained at different layers or blocks, using replay data to directly optimize these interpolation coefficients after task training.

Model Merging. Model merging aims to combine multiple trained models or task-specific updates into a single parameter set without requiring full retraining (Wortsman et al., 2022; Ilharco et al., 2022). A central challenge is that neural networks can represent similar functions in different parameter coordinate systems due to permutation and other function-preserving symmetries. Prior work therefore studies geometric alignment and mode connectivity between independently optimized networks, including Git-Re-Basin (Ainsworth et al., 2023), REPAIR (Jordan et al., 2023), and optimal-transport based model fusion (Singh & Jaggi, 2020). These methods provide important tools for improving compatibility between model endpoints before interpolation.

Beyond using a single fixed interpolation coefficient, recent model-merging approaches have also explored adaptive coefficient selection. In particular, AdaMerging (Yang et al., 2024) learns task-wise or layer-wise merging coefficients for multi-task model merging, demonstrating that different components of a network may benefit from different interpolation strengths. Our work is complementary to this line of research: rather than merging a collection of independently fine-tuned task models, AdaLMM considers the sequential continual-learning setting in which the historical checkpoint and the newly updated checkpoint are connected by task-wise training dynamics. AdaLMM uses replay data available at each task boundary to optimize layer-wise or block-wise historical-retention coefficients and combines this adaptive interpolation with architecture-aware alignment between the sequential endpoints.

Model merging has also been introduced into continual learning as a post-hoc mechanism for consolidating sequential knowledge. CLeWI (Kozal et al., 2024) combines rehearsal with weight interpolation between historical and updated models, while methods such as MagMax (Marczak et al., 2025) and MINGLE (Qiu et al., 2025) mitigate parameter interference through magnitude-based selection or structured routing in weight space. These works establish the effectiveness of parameter-space consolidation for continual learning, but differ in how the merging rule is determined. AdaLMM specifically studies *replay-driven adaptive interpolation*: instead of prescribing a fixed merging ratio across the network, it learns component-specific coefficients from the replay objective while explicitly accounting for the compatibility of the two sequential model endpoints.

B Variance Collapse and Statistics Reset

As introduced in Section 3.3, directly interpolating the parameters of neural networks—even when their hidden units are geometrically aligned—often leads to severe performance degradation. This is primarily caused by a phenomenon known as *variance collapse* (Jordan et al., 2023).

Specifically, consider a hidden unit in the interpolated network whose preactivation is denoted by $X_\alpha = (1-\alpha)X_{old}+\alpha X_{new}$. Because the correlation between aligned units across two independently trained networks is typically less than 1 (empirically around 0.4), the variance of the interpolated unit is strictly less than the linear interpolation of their individual variances. For a midpoint interpolation ($\alpha = 0.5$), the variance shrinks by approximately 30% relative to the endpoint networks (Jordan et al., 2023). This reduction compounds multiplicatively through the layers of deep architectures (e.g., ResNets), causing the activations in the final layers to become nearly constant, thereby destroying the network's ability to differentiate inputs.

To resolve this issue efficiently without requiring complex layer-by-layer affine transformations, we leverage the existing Batch Normalization (BN) layers in the architecture. By setting the newly interpolated network to `train()` mode and passing a small subset of samples (e.g., B instances) from the replay buffer $\mathcal{M}$ through it, we force the BN layers to re-estimate their running mean and variance. This brief statistics reset effectively renormalizes the decayed preactivations back to a standard normal distribution, completely mitigating the variance collapse and successfully restoring the representational capacity of the merged model (Jordan et al., 2023).

C Experimental Datasets and Preprocessing Details

In this section, we provide detailed statistics, configuration settings, and data preprocessing pipelines for the three standard continual learning benchmarks utilized in our evaluation. For all datasets, we adopt the Class-Incremental Learning (Class-IL) paradigm, where the model must classify test samples from all seen classes without access to task boundaries or task identifiers during inference.

Seq-CIFAR-10 (Krizhevsky et al., 2009). The CIFAR-10 dataset consists of 60,000 RGB images of size 32×32 pixels, uniformly distributed across 10 classes. The dataset is partitioned into 50,000 training images and 10,000 testing images. In our continual learning setup (Seq-CIFAR-10), the dataset is split into 5 sequential tasks, with each task introducing 2 disjoint classes to the model.

Seq-CIFAR-100 (Krizhevsky et al., 2009). The CIFAR-100 dataset shares the same scale and resolution as CIFAR-10 (i.e., 32×32 RGB images, 50,000 training and 10,000 testing samples) but contains 100 finer-grained classes. For the Seq-CIFAR-100 benchmark, these classes are divided into 10 incremental tasks, with each task comprising exactly 10 distinct classes.

Seq-Tiny-ImageNet (Le & Yang, 2015). Tiny-ImageNet is a scaled-down subset of the large-scale ImageNet dataset, designed to be more computationally tractable while preserving task difficulty. It contains 200 distinct classes, with each class comprising 500 training images, 50 validation images, and 50 testing images. The images are resized to a resolution of 64×64 pixels. In the Seq-Tiny-ImageNet configuration, the 200 classes are evenly partitioned into 20 sequential tasks, with 10 classes arriving per task.

Data Preprocessing and Augmentation. Following standard continual learning protocols, we apply a unified data augmentation and preprocessing pipeline across all tasks to prevent overfitting and improve generalization.

For ResNet-18 on Seq-CIFAR-10 and Seq-CIFAR-100, training images are augmented with random horizontal flips and random crops of size 32×32 after zero-padding by 4 pixels on each side. For Seq-Tiny-ImageNet, we use random horizontal flips and random crops of size 64×64 with 4-pixel padding. Images are converted to tensors, which scales the original pixel values to $[0, 1]$, and are then standardized using the channel-wise mean and standard deviation computed for each dataset.

For pretrained DeiT-S/16, the same crop-and-flip augmentation is applied at the original image resolution, followed by bicubic resizing to 224×224. The images are then converted to tensors and standardized using the ImageNet mean and standard deviation $(0.485, 0.456, 0.406)$ and $(0.229, 0.224, 0.225)$, respectively. During evaluation, random augmentation is disabled and only bicubic resizing, tensor conversion, and normalization are applied.

D Detailed Formulation of Evaluation Metrics

In this section, we provide the formal mathematical definitions for the evaluation metrics used in our experiments: Average Accuracy (A_N) and Average Forgetting (F_N). Following the standard

continual learning evaluation protocol (Chaudhry et al., 2018), we denote N as the total number of tasks sequentially learned by the model. Let $a_{i,j}$ represent the test accuracy of the model evaluated on the test set of task j immediately after the network has finished training on task i.

Average Accuracy (A_N). This metric measures the overall performance of the model across all seen tasks after the entire continual learning stream is completed. It is defined as the mean of the test accuracies on all N tasks after training on the final task N:

$$A_N = \frac{1}{N} \sum_{j=1}^{N} a_{N,j} \tag{12}$$

A higher A_N indicates that the model retains a better overall generalization capacity across the sequentially learned data distribution.

Average Forgetting (F_N). Forgetting quantifies the performance degradation on a specific task caused by learning subsequent new tasks. The forgetting for a specific task j after the model has been trained up to task N is defined as the difference between its peak accuracy (typically achieved right after training on task j) and its final accuracy at the end of the training stream. The Average Forgetting is the mean of these accuracy drops over all previous tasks:

$$F_N = \frac{1}{N-1} \sum_{j=1}^{N-1} \max_{l \in \{1,\cdots,N-1\}} (a_{l,j} - a_{N,j}) \tag{13}$$

Note that the average is computed over $N-1$ tasks because the final task N has not yet been subjected to any subsequent learning, and thus its forgetting is intrinsically zero. A lower F_N indicates that the model experiences less catastrophic forgetting and possesses higher stability.

E Implementation Details

E.1 Experimental Framework and General Settings

All experiments are implemented using the PyTorch framework and are built upon Mammoth Buzzega et al. (2020), a highly recognized and comprehensive library for continual learning. To ensure a strictly fair comparison, all baselines and our proposed AdaLMM share the exact same data augmentations, data loaders, and evaluation protocols inherently provided by the framework. All experiments are executed on a server equipped with Intel Xeon Gold 6326 processors and NVIDIA RTX 6000 Ada Generation GPUs (48GB VRAM). To ensure rigorous statistical significance, all reported metrics are averaged over 5 independent runs with different random seeds.

Backbone Architectures and Optimization. We evaluate two backbone architectures. For the ResNet-based experiments, we use a modified ResNet-18 (He et al., 2016) adapted to the image resolutions of Seq-CIFAR-10, Seq-CIFAR-100, and Seq-Tiny-ImageNet. During task learning, the model is optimized with stochastic gradient descent (SGD) using a learning rate of 0.1, momentum of 0.0, and weight decay of 0.0. We train for 20 epochs per task on Seq-CIFAR-10 and 50 epochs per task on both Seq-CIFAR-100 and Seq-Tiny-ImageNet.

For the ViT-based experiments, we use a pretrained DeiT-S/16 (Touvron et al., 2021) initialized from an ImageNet-1K pretrained checkpoint. The entire network is fine-tuned during continual learning. Unlike the ResNet experiments, the ViT model is optimized with AdamW using a learning rate of 1×10^{-4} and weight decay of 0.01. We use a cosine-annealing learning-rate schedule without warm-up, a batch size of 32 current samples and 32 replay samples, and train for 5 epochs per task on Seq-CIFAR-100 and Seq-Tiny-ImageNet. Input images are resized to 224×224 before being processed by the DeiT-S/16 backbone.

E.2 Baseline Hyperparameters

For the baseline methods, we use the optimal hyperparameters reported in their respective original papers:

- **DER++**: We set the distillation weight $\alpha = 0.1$ and the classification weight $\beta = 0.5$. To calculate its memory footprint accurately under fair constraints, we account for the extra memory required to store the continuous logits (Float32) in its specialized buffer.

- **CLeWI**: Following the optimal settings reported by the original authors, we employ specific static interpolation coefficients depending on the underlying rehearsal strategy. We set the static $\alpha = 0.3$ when paired with ER and ER-ACE, and $\alpha = 0.2$ when paired with DER++.

E.3 AdaLMM Hyperparameters

Our proposed AdaLMM utilizes an adaptive layer-wise gradient-driven optimization process to dynamically determine the plasticity-stability trade-off. For the differentiable merging phase, the learning rate for the merging coefficients is initialized at 0.1 with a batch size of 256. This proxy optimization is performed for 1 epoch using the Adam optimizer coupled with a Cosine Annealing learning rate scheduler, as the data maintained in the proxy memory buffer is sufficient to rapidly guide the adaptive merging.

F Strict Memory Alignment and Buffer Deduction

To guarantee a rigorously fair comparison, we enforce a strict total memory budget across all evaluated methods. In our AdaLMM framework, caching the historical model parameters (θ_{hist}) incurs additional memory overhead. Rather than ignoring this cost, we explicitly convert the model size into an equivalent number of training samples and deduct them from the replay buffer $\mathcal{M}$.

Sample Memory Footprint Calculation. During training, rehearsal buffers typically store images as `float32` tensors alongside `int64` labels. The memory footprint in bytes for a single sample is mathematically formulated as:

$$\text{Bytes}_{sample} = (C \times H \times W \times 4) + 8 \tag{14}$$

For CIFAR-10 and CIFAR-100 ($3 \times 32 \times 32$), a single sample strictly occupies $12,296$ bytes. For Tiny-ImageNet ($3 \times 64 \times 64$), the higher resolution increases the single-sample footprint to $49,160$ bytes.

Quantitative Deduction. Let M_{model} denote the exact memory size (in bytes) of the cached historical network state dictionary. The penalized buffer size B_{AdaLMM} is calculated by subtracting the equivalent sample capacity from the baseline buffer size B_{base}:

$$B_{AdaLMM} = B_{base} - \lfloor \frac{M_{model}}{\text{Bytes}_{sample}} \rfloor \tag{15}$$

In our main experiments, we evaluate the abundant memory regime under 2 dataset capacity settings: 60% and 80%. Because the cached model size (M_{model}) and the single-sample footprint (Bytes_{sample}) are fixed constants for a given dataset, the absolute number of deducted samples remains exactly the same across 2 settings. Therefore, to avoid redundancy, Table 4 provides the exact quantitative breakdown using the 60% dataset capacity setting as a representative baseline.

Notably, while the ResNet-18 model size remains relatively consistent (∼42-43 MB) across datasets, the number of deducted samples varies significantly. For instance, in Tiny-ImageNet, despite the slightly larger model capacity required for the expanded classifier head, the sample deduction (917) is substantially smaller than in CIFAR-10 (3638). This is because each 64×64 Tiny-ImageNet image consumes exactly four times the memory of a CIFAR image, making each sample "more expensive" in the memory budget. By applying this exact byte-for-byte compensation, we ensure that AdaLMM strictly operates under the exact same memory constraints as standard baseline methods across all capacity settings.

Table 4: Exact Memory Alignment Breakdown for AdaLMM across Datasets (illustrated under the 60% capacity setting).

Dataset	Resolution	Bytes / Sample	Cached Model Size	Base Buffer (B_{base}, 60%)	Deducted Samples	AdaLMM Buffer (B_{AdaLMM})
Seq-CIFAR-10	32×32	12,296	42.66 MB	30,000	-3,638	26,362
Seq-CIFAR-100	32×32	12,296	42.84 MB	30,000	-3,653	26,347
Seq-Tiny-ImageNet	64×64	49,160	43.03 MB	60,000	-917	59,083

G Details of Architecture-Aware Alignment

We use row-vector notation throughout this section, so a linear projection is written as

$$XW + \mathbf{1}b. \tag{16}$$

The tensor layout used by a software implementation may be transposed relative to this notation.

Uniform OT and hard permutation matching. Consider a matching site with K units and uniform marginals. The feasible transport set is

$$\Pi_K = \left\{ T \in \mathbb{R}_+^{K\times K} : T\mathbf{1} = \frac{1}{K}\mathbf{1}, \quad T^\top \mathbf{1} = \frac{1}{K}\mathbf{1} \right\}. \tag{17}$$

For any $T \in \Pi_K$, the matrix KT is doubly stochastic. By the Birkhoff–von Neumann theorem, the extreme points of this set are of the form P/K, where P is a permutation matrix. Since the unregularized OT objective is linear in T, it admits an optimal extreme-point solution

$$T^\star = \frac{1}{K}P^\star. \tag{18}$$

Thus, when a hard one-to-one alignment is desired, the corresponding permutation can be obtained by solving the induced linear assignment problem (Peyré & Cuturi, 2019; Kuhn, 1955). If several permutations attain the same minimum cost, non-extreme optimal couplings may also exist as convex combinations of optimal permutation solutions.

Entropy-regularized OT generally produces a soft coupling rather than a permutation. When an exact function-preserving reparameterization is required, the resulting correspondence must therefore be converted into an architecture-valid transformation before it is applied to the network.

Fully connected and convolutional units. Let P be a permutation applied to the output units of a fully connected or convolutional layer. To preserve the represented function, the same reordering must be propagated to every parameter that consumes that feature dimension. For convolutional layers, an output-channel permutation is therefore compensated by the corresponding input-channel permutation in subsequent convolutions. Bias terms, Batch Normalization affine parameters, and running statistics follow the same channel ordering.

Residual additions impose an additional consistency constraint: all branches participating in the same element-wise addition must use compatible output orderings. In particular, identity and projection shortcuts cannot be permuted independently when their outputs are combined. Applying these coupled transformations preserves the represented endpoint function (Ainsworth et al., 2023).

Feed-forward hidden units. Consider a feed-forward block

$$F(X) = \phi(XW_1 + \mathbf{1}b_1)W_2 + \mathbf{1}b_2, \tag{19}$$

where ϕ is applied elementwise. For a permutation matrix P, define

$$W_1' = W_1P, \qquad b_1' = b_1P, \qquad W_2' = P^\top W_2, \qquad b_2' = b_2. \tag{20}$$

Because an elementwise nonlinearity commutes with permutation,

$$\phi(ZP) = \phi(Z)P, \tag{21}$$

we obtain

$$\begin{aligned} F'(X) &= \phi(XW_1P + \mathbf{1}b_1P)P^\top W_2 + \mathbf{1}b_2 && (22)\\ &= \phi(XW_1 + \mathbf{1}b_1)PP^\top W_2 + \mathbf{1}b_2 && (23)\\ &= F(X). && (24) \end{aligned}$$

Hence, FFN hidden units can be permuted without changing the block output, provided that the adjacent linear projections are transformed consistently (Imfeld et al., 2024; Zhang et al., 2025). The residual-stream coordinates and surrounding normalization layers remain unchanged.

Attention-space orthogonal symmetries. After the correspondence between attention heads has been fixed, additional function-preserving transformations are available within each matched head (Zhang et al., 2025). Let W_Q, W_K, and W_V denote the query, key, and value projections, and let W_O denote the corresponding block of the output projection.

For the query–key subspace, let R be an orthogonal matrix satisfying

$$R^\top R = I. \tag{25}$$

We transform the corresponding parameters as

$$W'_Q = W_Q R, \qquad b'_Q = b_Q R, \qquad W'_K = W_K R, \qquad b'_K = b_K R. \tag{26}$$

This gives

$$Q' = QR, \qquad K' = KR, \tag{27}$$

and therefore

$$Q'K'^\top = QRR^\top K^\top = QK^\top. \tag{28}$$

Thus, the query–key attention scores are preserved.

Similarly, for the value–output subspace, let S be orthogonal:

$$S^\top S = I. \tag{29}$$

We apply

$$W'_V = W_V S, \qquad b'_V = b_V S, \qquad W'_O = S^\top W_O. \tag{30}$$

Then

$$V' = VS, \tag{31}$$

and

$$V'W'_O = VSS^\top W_O = VW_O. \tag{32}$$

Hence, the value–output contribution of the head is unchanged.

Orthogonal matching. The orthogonal transformations are estimated by matching corresponding historical and updated parameter blocks. For a generic stacked parameter matrix Z, we solve the orthogonal Procrustes problem

$$U^\star \in \arg\min_{U^\top U = I} \|Z_{\text{hist}} U - Z_{\text{new}}\|_F^2 . \tag{33}$$

If

$$Z_{\text{hist}}^\top Z_{\text{new}} = L\Sigma V^\top \tag{34}$$

is a singular-value decomposition, an orthogonal minimizer is

$$U^\star = LV^\top. \tag{35}$$

We apply this construction separately to the query–key and value–output parameter blocks to obtain the corresponding orthogonal transformations R and S (Zhang et al., 2025).

These transformations are restricted to attention subspaces in which the compensating parameter changes preserve the network function. They are not propagated through nonlinear FFN activations, residual-stream coordinates, or LayerNorm feature dimensions. Head permutations, when used, are applied consistently to the corresponding query, key, value, and output-projection blocks, while the output bias remains unchanged.

H Local Analysis of Layer-wise and Block-wise Interpolation

This section provides a local analysis of the difference between global interpolation and layer-wise or block-wise coefficient optimization. We fix the updated checkpoint, the aligned historical checkpoint, the replay objective, and the normalization state used during coefficient optimization.

For each component g, let $\mathbf{d}_g \in \mathbb{R}^d$ denote the displacement from the updated checkpoint toward the aligned historical checkpoint, with nonzero entries only for the parameters belonging to that component. Collecting these directions gives

$$D = [\mathbf{d}_1, \ldots, \mathbf{d}_G]. \tag{36}$$

The merged parameters can then be written as

$$\theta(\mathbf{a}) = \theta_{\text{new}} + D\mathbf{a}, \qquad \mathbf{a} \in [0,1]^G, \tag{37}$$

where $\mathbf{a} = [a_1, \ldots, a_G]^\top$ collects the interpolation coefficients.

Let $r(\theta)$ denote the empirical replay loss for a fixed normalization state, and define the replay objective in coefficient space as

$$\mathcal{J}(\mathbf{a}) = r(\theta_{\text{new}} + D\mathbf{a})\,. \tag{38}$$

Since D is independent of $\mathbf{a}$, the chain rule gives

$$\nabla_{\mathbf{a}}\mathcal{J} = D^\top \nabla_\theta r. \tag{39}$$

Thus, the gradient with respect to each interpolation coefficient measures the change in replay loss along the corresponding displacement from the updated checkpoint toward the aligned historical checkpoint.

Global versus layer-wise or block-wise interpolation. A single global interpolation coefficient restricts the coefficient vector to

$$\mathbf{a} = c\mathbf{1}, \qquad c \in [0,1]. \tag{40}$$

Let

$$j(c) = \mathcal{J}(c\mathbf{1}) \tag{41}$$

denote the replay loss along this one-dimensional global interpolation path. Its derivative is

$$j'(c) = \mathbf{1}^\top \nabla_{\mathbf{a}}\mathcal{J}(c\mathbf{1}). \tag{42}$$

Therefore, coefficient gradients from different components can cancel along the global direction even when the full coefficient-space gradient remains nonzero.

Proposition (Local descent beyond a stationary global coefficient). Let $\mathbf{a}_0 = c\mathbf{1}$ with $c \in (0,1)$, and define

$$\mathbf{g} = \nabla_{\mathbf{a}}\mathcal{J}(\mathbf{a}_0). \tag{43}$$

Suppose that $\mathcal{J}$ has an $L_{\mathcal{J}}$-Lipschitz continuous gradient in a neighborhood of $\mathbf{a}_0$. If

$$\mathbf{1}^\top \mathbf{g} = 0, \qquad \mathbf{g} \neq \mathbf{0}, \tag{44}$$

then the global interpolation path is stationary at c, while a sufficiently small step in the direction $-\mathbf{g}$ strictly decreases the replay objective.

Proof. From Eq. 42,

$$j'(c) = \mathbf{1}^\top \mathbf{g} = 0, \tag{45}$$

so the global interpolation path is stationary at c. However, $\mathbf{g} \neq \mathbf{0}$ implies that $\mathbf{a}_0$ is not stationary in the full coefficient space.

By the standard descent lemma for functions with Lipschitz-continuous gradients (Bottou et al., 2018), for any displacement $\mathbf{v}$ whose connecting segment remains in the neighborhood where the Lipschitz-gradient condition holds,

$$\mathcal{J}(\mathbf{a}_0 + \mathbf{v}) \leq \mathcal{J}(\mathbf{a}_0) + \mathbf{g}^\top \mathbf{v} + \frac{L_{\mathcal{J}}}{2}\|\mathbf{v}\|_2^2. \tag{46}$$

Taking $\mathbf{v} = -\eta\mathbf{g}$ gives

$$\mathcal{J}(\mathbf{a}_0 - \eta\mathbf{g}) \leq \mathcal{J}(\mathbf{a}_0) - \eta\left(1 - \frac{L_{\mathcal{J}}\eta}{2}\right)\|\mathbf{g}\|_2^2. \tag{47}$$

For

$$0 < \eta < \frac{2}{L_{\mathcal{J}}}, \tag{48}$$

the coefficient multiplying $\|\mathbf{g}\|_2^2$ is strictly positive. Since $c \in (0,1)$, a sufficiently small η ensures that the entire segment from $\mathbf{a}_0$ to $\mathbf{a}_0 - \eta\mathbf{g}$ remains both inside $[0,1]^G$ and within the neighborhood where the Lipschitz-gradient assumption holds. Therefore,

$$\mathcal{J}(\mathbf{a}_0 - \eta\mathbf{g}) < \mathcal{J}(\mathbf{a}_0), \tag{49}$$

which proves the claim. □

AdaLMM optimizes unconstrained variables $\mathbf{w}$ through the sigmoid parameterization $\mathbf{a} = \sigma(\mathbf{w})$. At any interior coefficient vector $\mathbf{a} \in (0,1)^G$, the sigmoid Jacobian

$$\frac{\partial \mathbf{a}}{\partial \mathbf{w}} = \operatorname{diag}\left(\mathbf{a} \odot (1 - \mathbf{a})\right) \tag{50}$$

has strictly positive diagonal entries. Hence, the local distinction between the one-dimensional global interpolation path and the larger layer-wise or block-wise coefficient space is preserved under the optimization variables $\mathbf{w}$.

This result shows that stationarity along the one-dimensional global interpolation path does not necessarily imply stationarity in the larger layer-wise or block-wise coefficient space. The statement is local and applies to the fixed replay objective used during coefficient optimization.

I Derivation of the Coefficient Gradient

For component l, AdaLMM parameterizes the merged parameters as

$$\theta^{(l)}(\alpha_l) = (1 - \alpha_l)\theta^{(l)}_{\text{new}} + \alpha_l \widetilde{\theta}^{(l)}_{\text{hist}}, \tag{51}$$

where

$$\alpha_l = \sigma(w_l). \tag{52}$$

Rewriting the interpolation gives

$$\theta^{(l)}(\alpha_l) = \theta^{(l)}_{\text{new}} + \alpha_l \left(\widetilde{\theta}^{(l)}_{\text{hist}} - \theta^{(l)}_{\text{new}}\right). \tag{53}$$

Hence,

$$\frac{\partial \theta^{(l)}}{\partial \alpha_l} = \widetilde{\theta}^{(l)}_{\text{hist}} - \theta^{(l)}_{\text{new}}. \tag{54}$$

Since $\alpha_l = \sigma(w_l)$,

$$\frac{\partial \alpha_l}{\partial w_l} = \sigma(w_l)\big(1 - \sigma(w_l)\big) = \alpha_l(1 - \alpha_l). \tag{55}$$

Therefore,

$$\frac{\partial \theta^{(l)}}{\partial w_l} = \alpha_l(1 - \alpha_l)\left(\widetilde{\theta}^{(l)}_{\text{hist}} - \theta^{(l)}_{\text{new}}\right). \tag{56}$$

The replay objective $\mathcal{J}_t$ depends on w_l through the interpolated parameters assigned to component l. Applying the chain rule yields

$$\frac{\partial \mathcal{J}_t}{\partial w_l} = \left\langle \nabla_{\theta^{(l)}} \mathcal{J}_t, \frac{\partial \theta^{(l)}}{\partial w_l} \right\rangle \tag{57}$$

$$= \alpha_l(1 - \alpha_l)\left\langle \nabla_{\theta^{(l)}} \mathcal{J}_t, \widetilde{\theta}^{(l)}_{\text{hist}} - \theta^{(l)}_{\text{new}} \right\rangle, \tag{58}$$

which gives Eq. 11.

When a coefficient is shared by multiple parameter tensors within the same layer or block, their gradient contributions are summed. Let $\mathcal{I}_l$ denote the set of parameter tensors assigned to component l. Then

$$\frac{\partial \mathcal{J}_t}{\partial w_l} = \alpha_l(1 - \alpha_l) \sum_{p \in \mathcal{I}_l} \left\langle \nabla_{\theta_p} \mathcal{J}_t, \widetilde{\theta}_{\text{hist},p} - \theta_{\text{new},p} \right\rangle. \tag{59}$$

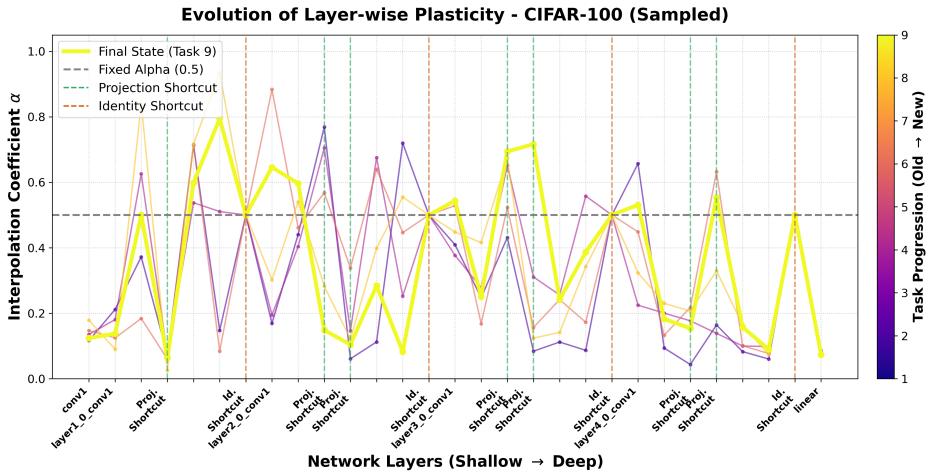

Figure 4: Evolution of the learned historical-retention coefficients α_l on Seq-CIFAR-100 with an 80% buffer. Smaller α_l corresponds to retaining a larger fraction of the current-task update. The color gradient represents task progression.

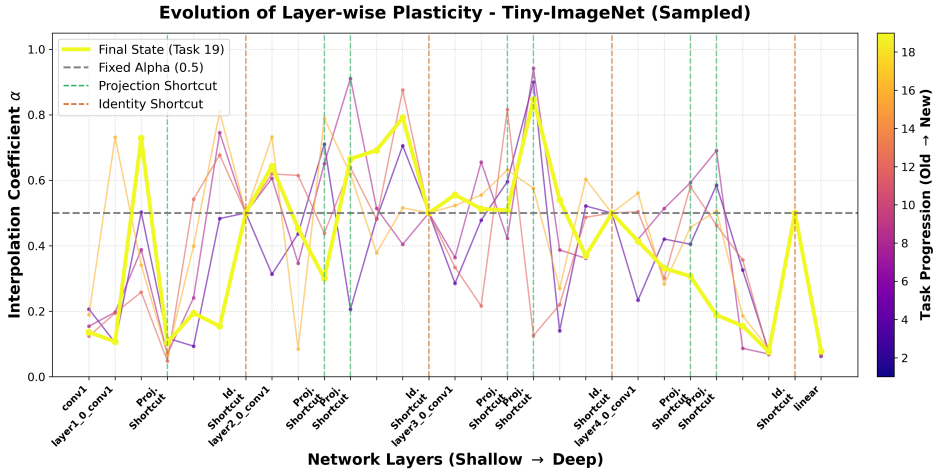

Figure 5: Evolution of the learned historical-retention coefficients α_l on Seq-Tiny-ImageNet with an 80% buffer. Smaller α_l corresponds to retaining a larger fraction of the current-task update. The color gradient represents task progression.

J Fine-grained Analysis on Layer-wise Merging Dynamics

To dissect AdaLMM's inner workings, we track the learned layer-wise merging coefficients $\alpha_l = \sigma(w_l)$ for **AdaLMM+ER** (80% buffer) across sequential tasks (Figure 4, 5). The resulting heterogeneous distributions validate that uniform merging fails to capture hierarchical demands, revealing two key insights:

Heterogeneous Layer-wise Plasticity. Stability-plasticity demands are highly non-linear. The extremities (`conv1` and `linear`) frequently converge to high plasticity ($\alpha \approx 0$) to accommodate low-level pixel shifts and novel semantics. Crucially, intermediate residual blocks exhibit divergent stabilization patterns with alternating peaks and valleys, where adjacent layers often stabilize at opposite extremes. This confirms that mid-level representations selectively consolidate based on specific functional demands—intricate nuances completely overlooked by uniform scalar merging or rigid heuristics.

Dichotomy in Shortcut Connections. The $\boldsymbol{\alpha}$ distribution exposes a rigorous dichotomy mapping exactly to residual topologies. Explicit *projection* shortcuts actively shift their α to accommodate structural transformations. Conversely, pure *identity* shortcuts remain anchored at $\alpha = 0.5$; lacking learnable parameters across tasks, their proxy gradients intrinsically vanish. This gracefully verifies that AdaLMM exclusively adapts layers with tangible parametric conflicts, leaving parameter-free pathways undisturbed.

K Performance under Extreme Data Scarcity

Table 5: Evaluation under Strictly Data-Aligned. All methods are restricted to store the exact same number of historical exemplars (500 and 1000). We report Average Accuracy ($A_N \uparrow$) and Average Forgetting ($F_N \downarrow$). "±" denotes standard deviation across 5 different random seeds.

Method	Seq-CIFAR-10 (5 Tasks)		Seq-CIFAR-100 (10 Tasks)		Seq-Tiny-ImageNet (20 Tasks)	
	$A_N(\%) \uparrow$	$F_N(\%) \downarrow$	$A_N(\%) \uparrow$	$F_N(\%) \downarrow$	$A_N(\%) \uparrow$	$F_N(\%) \downarrow$
Buffer Capacity: 500 Exemplars Total						
iCaRL	62.05 ± 0.96	22.97 ± 2.19	**45.16** ± 0.49	24.95 ± 0.97	**22.39** ± 0.59	25.21 ± 1.05
ER	54.03 ± 4.62	53.64 ± 6.22	22.88 ± 1.49	72.16 ± 1.25	6.54 ± 0.58	79.72 ± 0.90
+ **AdaLMM** (Ours)	66.32 ± 4.50 (+12.29)	29.54 ± 7.20 (-24.10)	40.18 ± 1.11 (+17.30)	34.92 ± 1.35 (-37.24)	16.16 ± 0.45 (+9.62)	56.97 ± 0.85 (-22.75)
DER++	69.56 ± 1.90	25.00 ± 2.17	34.86 ± 0.38	54.78 ± 0.59	10.82 ± 1.94	67.47 ± 2.14
+ **AdaLMM** (Ours)	**70.88** ± 0.20 (+1.32)	6.92 ± 1.80 (-18.08)	41.41 ± 1.03 (+6.55)	16.08 ± 1.53 (-38.70)	17.98 ± 1.27 (+7.16)	24.24 ± 1.45 (-43.23)
ER-ACE	68.03 ± 1.72	13.69 ± 2.35	37.66 ± 1.13	38.05 ± 1.55	14.87 ± 0.64	42.68 ± 1.40
+ **AdaLMM** (Ours)	58.20 ± 2.09 (-9.83)	**-7.47** ± 0.99 (-21.16)	37.92 ± 0.48 (+0.26)	**11.34** ± 0.45 (-26.71)	17.61 ± 0.31 (+2.74)	**19.26** ± 0.89 (-23.42)
Buffer Capacity: 1000 Exemplars Total						
iCaRL	66.08 ± 1.47	23.11 ± 2.60	47.80 ± 0.33	23.21 ± 0.86	**27.15** ± 0.21	**21.72** ± 0.87
ER	64.26 ± 5.29	39.70 ± 7.77	30.71 ± 0.94	63.15 ± 0.50	9.54 ± 0.59	76.11 ± 0.72
+ **AdaLMM** (Ours)	71.28 ± 3.90 (+7.02)	25.76 ± 4.99 (-13.94)	43.23 ± 0.70 (+12.52)	40.64 ± 0.97 (-22.51)	18.04 ± 0.46 (+8.50)	60.87 ± 1.25 (-15.24)
DER++	71.31 ± 7.66	22.16 ± 9.22	43.59 ± 1.02	43.98 ± 1.25	17.71 ± 2.21	58.92 ± 3.31
+ **AdaLMM** (Ours)	**74.97** ± 1.23 (+3.66)	6.99 ± 3.33 (-15.17)	**48.60** ± 1.91 (+5.01)	19.80 ± 0.35 (-24.18)	23.08 ± 1.63 (+5.37)	37.37 ± 2.27 (-21.55)
ER-ACE	73.05 ± 1.44	11.22 ± 1.76	43.81 ± 0.63	32.68 ± 1.20	20.25 ± 1.08	40.80 ± 0.88
+ **AdaLMM** (Ours)	65.49 ± 2.86 (-7.56)	**-1.21** ± 2.28 (-12.43)	45.15 ± 0.48 (+1.34)	**15.08** ± 0.48 (-17.60)	21.81 ± 1.02 (+1.56)	23.12 ± 1.28 (-17.68)

In some practical Continual Learning deployments, the primary limiting factor preventing the historical buffer from acquiring sufficient replay data is often not the physical hardware storage capacity, but rather the strict unavailability of historical data. Due to rigorous privacy regulations or proprietary data access restrictions, practitioners are frequently permitted to retain only a minimal number of historical samples, regardless of the available disk space. To address this realistic scenario, we evaluate the AdaLMM framework under extreme data scarcity, restricting the replay buffer to exceptionally small sizes (500 and 1000 exemplars in total).

Strictly Data-Aligned Setting. Because the core motivation here is driven by strict data unavailability rather than hardware memory efficiency, this evaluation does not employ the memory-equivalent deduction used in the main text. Instead, we enforce a **Strictly Data-Aligned** setting. All evaluated methods, including our AdaLMM variants, are granted access to the exact same number of historical exemplars, allowing us to purely assess the algorithms' resilience against severe data scarcity.

Analysis of Results. The quantitative results are comprehensively reported in Table 5. Under these strictly limited exemplar budgets, standard rehearsal methods struggle significantly to maintain historical knowledge, suffering from catastrophic forgetting. Integrating AdaLMM yields substantial and consistent improvements:

- **Massive Forgetting Mitigation:** AdaLMM dramatically suppresses forgetting across complex datasets. For instance, on Seq-CIFAR-100 with only 500 exemplars, AdaLMM reduces the F_N of ER from 72.16% to 34.92% (a massive $\mathbf{37.24}\%$ absolute reduction), and DER++ from 54.78% to 16.08% (a $\mathbf{38.70}\%$ reduction). Similar dominant trends are observed on Seq-Tiny-ImageNet.
- **Breaking the Capacity Trade-off:** As the task complexity scales up (Seq-CIFAR-100 and Seq-Tiny-ImageNet), standard methods hit severe bottlenecks due to insufficient rehearsal signals. AdaLMM effectively compensates for this lack of data through its parameter-space geometric alignment, delivering simultaneous and significant gains in both average accuracy ($A_N \uparrow$) and average forgetting ($F_N \downarrow$).

L COMPUTATIONAL OVERHEAD WITH RESNET-18

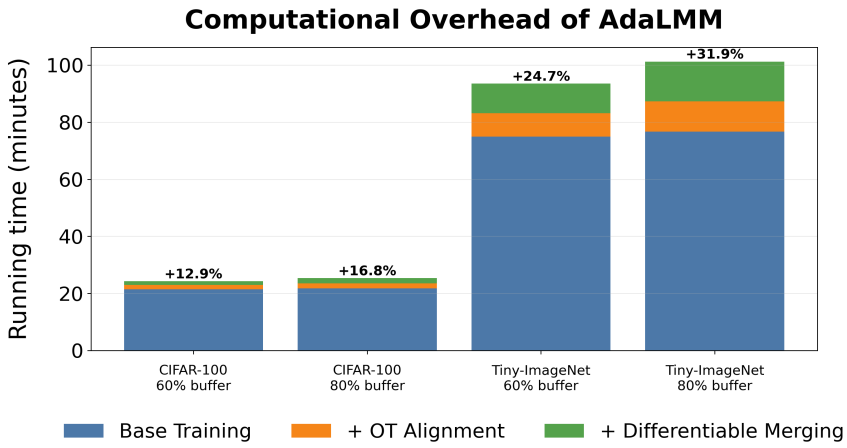

Figure 6: Computational overhead of AdaLMM with ResNet-18. Percentages denote the additional execution time relative to the corresponding ER baseline.