Latent-LoRA: Compact Latent-Space Adapters with Gradient-Free Routing for Continual Learning

Reza Rahimi Azghan[1], Gautham Krishna Gudur[2], Giulia Pedrielli[3], Pavan Turaga[4], Hassan Ghasemzadeh[1]

[1]College of Health Solutions, Arizona State University, Phoenix, Arizona, USA
[2]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas, USA
[3]School of Computing and Augmented Intelligence, Arizona State University, Tempe, Arizona, USA
[4]The GAME School, Arizona State University, Tempe, Arizona, USA

Correspondence: rrahimia@asu.edu

Abstract

Large language models generalize well to individual tasks but lack an inherent mechanism for learning them sequentially, leading to catastrophic forgetting. To mitigate this, LoRA-based continual learning methods allocate a separate low-rank adapter per task, yet existing approaches either require task identity at inference or sum all adapters indiscriminately, letting irrelevant branches distort the output. Recent gating-based solutions route inputs to the correct adapter but introduce trainable parameters that themselves need protection against forgetting. In this work, we observe that pooled token embeddings from a frozen LLM embedding layer already separate task distributions throughout the learning sequence. A Gaussian mixture model fitted on these embeddings, without any gradient-based training, is sufficient for task-agnostic adapter selection at test time. This eliminates the need for a learned gating module. On the adapter side, constraining each task's parameters to the principal subspace of the pretrained weights via SVD yields a compact latent-space parameterization. Within this subspace, orthogonal regularization directly controls inter-task interference. The resulting system, Latent-LoRA, is replay-free, requires no trainable routing component, and uses substantially fewer parameters per task. Experiments across five model scales and two established continual learning benchmarks show state-of-the-art performance with near-zero forgetting.

1 Introduction

Continual learning (CL), which requires a model to learn a sequence of tasks without forgetting previously acquired knowledge, is a fundamental challenge for large language models (LLMs). While LLMs achieve strong performance across a wide range of tasks through pre-training and fine-tuning, they tend to overwrite earlier knowledge when trained on new tasks, a well-known phenomenon called catastrophic forgetting (McCloskey and Cohen, 1989; Kemker et al., 2018; French, 1999). This problem is particularly pronounced in LLMs due to their massive parameter counts, which give them high capacity for new tasks but little capacity for preserving old ones (Shi et al., 2024; Luo et al., 2023; De Lange et al., 2021; Rebuffi et al., 2017).

Low-rank adaptation (LoRA) (Hu et al., 2022; Dettmers et al., 2023) has become a popular foundation for CL in LLMs. By reparameterizing weight updates as low-rank matrices, LoRA enables parameter-efficient fine-tuning that fits naturally into the CL setting (Wang et al., 2024): each new task receives its own LoRA branch while previous branches remain frozen, preventing direct interference with earlier knowledge. Methods such as O-LoRA (Wang et al., 2023) further constrain new branches to lie in subspaces orthogonal to previous ones to reduce cross-task overlap. However, a critical problem persists at inference. Since task identities are unavailable in the task-agnostic setting, O-LoRA integrates all branches by simple summation, forcing every branch to influence the output on every input. This indiscriminate aggregation allows branches trained for one task to distort predictions on another and reintroduces forgetting despite the frozen parameters and orthogonal training.

Recent work has attempted to address this by inserting a learned gating module between the input and each branch. GainLoRA (Liang and Li, 2025), for instance, introduces a per-task MLP gate trained to activate the corresponding branch while suppressing others. While effective, this approach has its own drawbacks. The number of gating parameters grows linearly with the number of tasks, adding significant overhead. More fundamentally, the gates themselves are susceptible to forgetting, since each new gate's training can interfere with the subspaces used by previous gates. To prevent this, GainLoRA applies Gradient Projection Memory (Saha et al., 2021) to constrain gate updates

orthogonally. It essentially introduces a second continual learning mechanism to protect the first.

In this work, we take a different approach. We observe that pooled token embeddings produced by an LLM's own frozen embedding layer already separate task distributions cleanly enough to be routed by a simple probabilistic model. Concretely, after each task, we fit a Gaussian mixture over that task's training-data embeddings; at inference, the resulting mixtures yield a posterior distribution over tasks for any input, which is used to weight the contribution of each task's adapter. The router has stored parameters (means and covariance) but no trainable ones, and cannot be corrupted by subsequent training. We pair this router with compact adapters based on LoRA-XS (Bałazy et al., 2025), which constrain each task's weight update to the principal subspace of the pretrained weights via SVD, reducing per-task storage to a small $r \times r$ matrix, where r is the adapter rank, building on ideas explored in SVFT (Lingam et al., 2024). Each adapter is saved as an independent snapshot that is never modified after training, so no subsequent task can corrupt a previously learned adapter.

The scientific contributions in this work are:

- We propose a training-free Gaussian mixture router that operates on pooled token embeddings from the base LLM's frozen embedding layer. The router has no trainable parameters, requires no protection against forgetting, and adds negligible storage per task.
- To the best of our knowledge, this is the first application of low-rank latent-space adapters to continual learning. By training only a small square matrix in the SVD-induced latent subspace of each pretrained weight, we substantially reduce per-task storage compared to the standard LoRA branches.
- Our method, LatentLoRA, achieves state-of-the-art average performance on standard CL benchmarks in the task-agnostic, exemplar-free setting, without any trainable routing component.

2 Background

2.1 Continual Learning Setup

Continual learning considers a setting where a model is trained on a sequence of tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, arriving one at a time (Parisi et al., 2019). In the supervised setting, each task consists of input-label pairs $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_t}$ drawn from a task-specific distribution $P_t(\boldsymbol{x}, y)$, where n_t is the number of examples in task t. The goal is to find parameters $\boldsymbol{\Theta}$ for a single model that performs well across all tasks seen so far:

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \sum_{t=1}^{T} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_t} \log\, p(y \mid \boldsymbol{x}; \boldsymbol{\Theta}) \quad (1)$$

The central challenge is that when training on task t, the model no longer has access to the data of previous tasks $\{\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}\}$, which leads to catastrophic forgetting of earlier knowledge. In this work, we operate under the task-agnostic, exemplar-free setting (Aljundi et al., 2019): the model trains on each task sequentially without storing any data from prior tasks, and at inference time, no task identity is provided.

2.2 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a parameter-efficient fine-tuning method (Houlsby et al., 2019) that exploits the observation that weight updates in pre-trained models tend to lie in a low-dimensional subspace (Aghajanyan et al., 2021). For a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, LoRA constrains the update $\Delta\boldsymbol{W}$ to a low-rank decomposition $\Delta\boldsymbol{W} = \boldsymbol{BA}$, where $\boldsymbol{B} \in \mathbb{R}^{m\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times n}$, and $r \ll \min(m, n)$. The pre-trained weight $\boldsymbol{W}$ remains frozen during training, and the forward pass of the adapted layer becomes:

$$\boldsymbol{e} = (\boldsymbol{W} + \boldsymbol{BA})\boldsymbol{h} \quad (2)$$

where $\boldsymbol{h}$ and $\boldsymbol{e}$ denote the input and output of the layer, respectively. Only $\boldsymbol{B}$ and $\boldsymbol{A}$ are updated during fine-tuning to reduce the number of trainable parameters from $m \times n$ to $r \times (m + n)$.

LoRA in Continual Learning. LoRA naturally lends itself to continual learning (Biderman et al., 2024): each new task can receive its own LoRA branch $(\boldsymbol{B}_t, \boldsymbol{A}_t)$ while all previous branches are frozen, preventing direct modification of earlier parameters. O-LoRA (Wang et al., 2023) further constrains each new branch to lie in the orthogonal complement of previous branches to reduce subspace overlap across tasks. However, at inference time, since task identities are unavailable, O-LoRA aggregates all branches through addition:

$$\boldsymbol{e} = \left(\boldsymbol{W} + \sum_{t=1}^{T} \boldsymbol{B}_t\boldsymbol{A}_t\right)\boldsymbol{h} \quad (3)$$

This forces every branch to influence every input and allows irrelevant adapters to degrade performance on earlier tasks. GainLoRA (Liang and Li, 2025) addresses this by introducing a learned gating module $g_t(\boldsymbol{x})$ per task that controls the contribution of each branch:

$$\boldsymbol{e} = \left(\boldsymbol{W} + \sum_{t=1}^{T} g_t(\boldsymbol{x}) \cdot \boldsymbol{B}_t \boldsymbol{A}_t \right) \boldsymbol{h} \tag{4}$$

While effective, this approach introduces a per-task MLP gate with its own trainable parameters, and the gates themselves are susceptible to forgetting. GainLoRA mitigates this by applying Gradient Projection Memory (GPM) to the gate parameters, adding further complexity. Moreover, both the adapter and gate parameters scale linearly with the number of tasks.

3 Methodology

3.1 Overview

Figure 1 illustrates the full pipeline of Latent-LoRA. The base language model remains frozen throughout, including its embedding layer. After training each task's adapter, we extract pooled token embeddings $\phi(\boldsymbol{x}) = \mathrm{MeanPool}(\mathrm{Emb}(\boldsymbol{x}))$ from the frozen embedding layer and fit a Gaussian mixture over them. The per-task mixtures collectively form a router that maps any input $\boldsymbol{x}$ to a posterior $p(t \mid \boldsymbol{x})$ over known tasks, without any trainable parameters (§3.3). Each task's adapter R_t is a compact matrix trained in the SVD-derived latent subspace of the pretrained weights and saved as an independent snapshot that no subsequent task can modify (§3.2).

3.2 Compact Latent-Space Adapters

Standard LoRA-based continual learning methods allocate a pair of trainable low-rank matrices $(\boldsymbol{A}_t, \boldsymbol{B}_t)$ per task, with $r(m+n)$ parameters per target module. While modest compared to full fine-tuning, this cost scales linearly with the model dimension and accumulates across tasks.

We adopt a more compact parameterization based on LoRA-XS (Bałazy et al., 2025), which constrains weight updates to the principal subspace of the pretrained weights. Given a target weight $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, we compute its rank-$r$ truncated SVD $\boldsymbol{W} \approx \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V_r}^\top$ once and keep all three factors frozen. Here $\boldsymbol{U}_r \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V}_r \in \mathbb{R}^{n \times r}$ have orthonormal columns, and $\boldsymbol{\Sigma}_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ contains the top r singular values. The per-task adapter is a small trainable matrix $\boldsymbol{R}_t \in \mathbb{R}^{r \times r}$, and the weight update is:

$$\Delta \boldsymbol{W}_t = \boldsymbol{U}_r \, \boldsymbol{\Sigma}_r \, \boldsymbol{R}_t \, \boldsymbol{V}_r^\top. \tag{5}$$

This reduces trainable parameters per module from $r(m+n)$ to r^2, which is a factor of $(m+n)/r$ that grows with model size.

By the Eckart-Young-Mirsky theorem (Eckart and Young, 1936), the subspace $\mathcal{S}_r = \{\boldsymbol{U}_r \boldsymbol{X} \boldsymbol{V}_r^\top : \boldsymbol{X} \in \mathbb{R}^{r \times r}\}$ is the optimal subspace for approximating gradient updates in the Frobenius norm, assuming that fine-tuning gradients lie close to the distribution of pretraining gradients. We refer the reader to (Bałazy et al., 2025) for a formal proof. Beyond parameter efficiency, this parameterization has a structural consequence for continual learning: because the high-dimensional factors $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ are frozen and orthonormal, the interference between any two task adapters reduces to a function of their $r \times r$ matrices alone.

Output perturbation and interference. Forgetting in LoRA-based CL arises when a new task's adapter perturbs the model's output on old-task inputs (Goodfellow et al., 2013). We formalize this at the layer level. For an input $\boldsymbol{h}$, task j's adapter produces the output perturbation:

$$\delta_j(\boldsymbol{h}) = \Delta \boldsymbol{W}_j \, \boldsymbol{h} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h} \tag{6}$$

Define the *latent-space projection* $\psi(\boldsymbol{h}) = \boldsymbol{V}_r^\top \boldsymbol{h} \in \mathbb{R}^r$ and the *weighted adapter* $\tilde{\boldsymbol{R}}_t = \boldsymbol{\Sigma}_r \boldsymbol{R}_t \in \mathbb{R}^{r \times r}$. The inner product of the output perturbations from tasks i and j measures the extent to which task j's perturbation has a component along the direction learned by task i, and it decomposes as:

$$\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h}) \;=\; \psi(\boldsymbol{h})^\top \; \tilde{\boldsymbol{R}}_i^\top \; \tilde{\boldsymbol{R}}_j \; \psi(\boldsymbol{h}), \tag{7}$$

The derivation is given in Appendix A. This interference is a quadratic form in $\psi(\boldsymbol{h})$ whose kernel is $\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j$, a quantity that depends only on the adapters and singular values, with the high-dimensional projection factors $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ canceling out due to orthonormality.

Orthogonal regularization. The decomposition (7) identifies $\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j$ as the quantity governing inter-task interference. We therefore regularize it directly:

$$\mathcal{L}_{\mathrm{ortho}} = \sum_{i<j} \left\| \tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j \right\|_F^2. \tag{8}$$

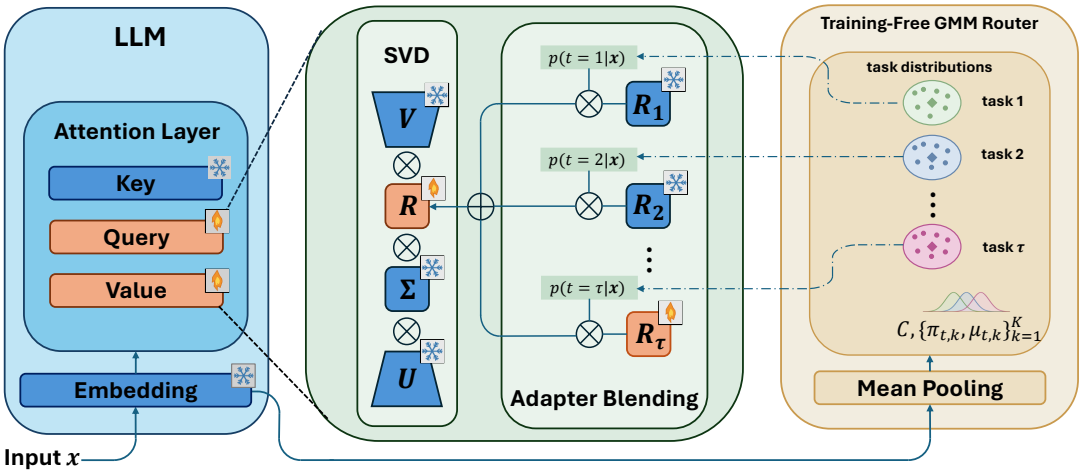


Figure 1: Latent-LoRA trains a per-task square matrix in the frozen SVD subspace and routes inputs via a training-free Gaussian mixture over pooled embeddings

This weighting arises naturally from the interference analysis: directions corresponding to larger singular values produce larger output perturbations and are penalized proportionally. We show that this regularization provides upper-bound control over the interference:

Proposition 1. *Under the latent-space adapter parameterization, the output interference satisfies, for all inputs* $\boldsymbol{h}$*:*

$$\left|\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h})\right| \;\le\; \left\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j\right\|_2 \cdot \|\psi(\boldsymbol{h})\|^2. \quad (9)$$

The proof derives from (7) and is given in Appendix A. As the regularization (8) drives $\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j\|_2$ towards zero, it progressively suppresses interference across all inputs. The factor $\|\psi(\boldsymbol{h})\|^2 = \|\boldsymbol{V}_r^\top \boldsymbol{h}\|^2$ depends only on the input and the pretrained model's singular subspace; it cannot grow during adapter training. Crucially, the regularization target is identical to the interference kernel, with no uncontrolled residual. We contrast this with standard LoRA's incomplete regularization and discuss the implicit regularization benefits of compact adapters in Appendix B.

The full training objective for task t combines the standard language modeling loss with the orthogonal regularization:

$$\mathcal{L}_t = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_t} \log p_\Theta(\boldsymbol{y} \mid \boldsymbol{x}) + \lambda \sum_{i=1}^{t-1} \left\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_t\right\|_F^2, \quad (10)$$

where λ controls the strength of the orthogonal penalty. Only $\boldsymbol{R}_t$ receives gradients; all previous adapters $\{\boldsymbol{R}_i\}_{i<t}$ are frozen snapshots.

Snapshot isolation. After training on task t, the adapter $\boldsymbol{R}_t$ is saved as an independent snapshot and never modified. Once all adapters are frozen, performance on old tasks can only change through two channels: the router assigning a nonzero probability to the wrong adapter, or interference between adapters when they are blended via $\boldsymbol{R}(\boldsymbol{x}) = \sum_t p(t \mid \boldsymbol{x})\boldsymbol{R}_t$. The orthogonal regularization controls the second; the training-free router, introduced next, controls the first.

3.3 Training-Free Task Router

In the task-agnostic setting, the model must determine which adapter(s) to apply without receiving a task identity. Prior methods either aggregate all adapters indiscriminately (Wang et al., 2023) or introduce learned gating modules that require their own protection against forgetting (Liang and Li, 2025). We take a different approach: we observe that the pooled token embeddings produced by the base model's frozen embedding layer already separate task distributions with sufficient margin for accurate routing, and exploit this with a simple probabilistic model that requires no gradient-based training.

Embedding extraction. For an input text $\boldsymbol{x}$, we extract a fixed-size representation using the base model's own embedding layer:

$$\phi(\boldsymbol{x}) = \mathrm{MeanPool}\big(\mathrm{Emb}(\boldsymbol{x})\big), \quad (11)$$

where Emb denotes the model's input embedding table and MeanPool averages the resulting token-level vectors into a single vector $\phi(\boldsymbol{x}) \in \mathbb{R}^d$. Because LoRA adapters are applied to the attention

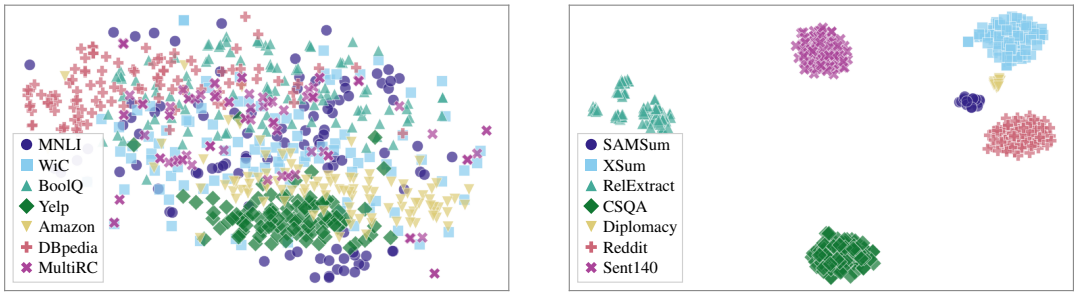


Figure 2: t-SNE projection of pooled token embeddings from the frozen T5-Large embedding layer. Left: Long Sequence. Right: SuperNI.

projections and not to the embedding layer, $\phi(\boldsymbol{x})$ is stationary across the task sequence. Therefore, the distributions the router was fitted on do not drift as new tasks are learned.

Figure 2 visualizes $\phi(\boldsymbol{x})$ via t-SNE (van der Maaten and Hinton, 2008) for tasks from two CL benchmarks, Long Sequence (Razdaibiedina et al., 2023) and SuperNI (Wang et al., 2022) using the frozen T5-Large (Raffel et al., 2020) embedding layer. Task distributions form well-separated clusters without task-specific training, suggesting that pretrained embeddings provide useful structure for routing. Similar patterns across more language model scales are shown in Appendix D.

Gaussian mixture model. After training the adapter for task t, we fit a task-specific Gaussian mixture model (GMM) over the training embeddings $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{D}_t\}$. Each task's distribution is modeled as a mixture of K Gaussians (Reynolds, 2009):

$$p\big(\phi(\boldsymbol{x}) \mid t\big) = \sum_{k=1}^{K} \pi_{t,k}\, \mathcal{N}\big(\phi(\boldsymbol{x}) \,\big|\, \mu_{t,k}, \mathbf{C}\big), \quad (12)$$

where $\pi_{t,k}$ and $\mu_{t,k}$ are the weight and mean of component k for task t, initialized via K-means clustering on the task's embeddings, and $\mathbf{C}$ is a covariance matrix shared across all tasks and components. Using multiple components per task allows the model to capture multi-modal task distributions; for instance, a question-answering task may contain both short factoid questions and long passage-based questions, which occupy different regions of the embedding space.

The shared covariance is computed as the regularized pooled within-task scatter:

$$\mathbf{C} = \frac{\sum_{t=1}^{T} \mathbf{S}_t}{\sum_{t=1}^{T} n_t} + \epsilon \boldsymbol{I}, \quad (13)$$

where $\mathbf{S}_t = \sum_{\boldsymbol{x} \in \mathcal{D}_t} (\phi(\boldsymbol{x}) - \bar{\phi}_t)(\phi(\boldsymbol{x}) - \bar{\phi}_t)^\top$ is the scatter matrix for task t, n_t is the number of training samples, ϵ is a regularization coefficient for numerical stability, and $\bar{\phi}_t$ is the mean embedding for task t.

Sharing a single covariance across tasks follows the Linear Discriminant Analysis (LDA) principle of pooled within-class scatter under a shared-covariance assumption, an approach also used in recent continual learning work (Momeni et al., 2025; Goswami et al., 2023). In our setting, this rescales the embedding space according to common variation across tasks, emphasizing task-discriminative directions. When a new task arrives, $\mathbf{C}$ is updated via (13) without requiring access to previous tasks' data.

Soft routing and adapter blending. At inference, the router computes the log-likelihood of $\phi(\boldsymbol{x})$ under each task's mixture and derives the task posterior under a uniform prior:

$$p(t \mid \boldsymbol{x}) = \frac{p(\phi(\boldsymbol{x}) \mid t)}{\sum_{t'=1}^{T} p(\phi(\boldsymbol{x}) \mid t')}. \quad (14)$$

The posterior is used to blend all stored adapters into a single effective adapter for input $\boldsymbol{x}$:

$$\boldsymbol{R}(\boldsymbol{x}) = \sum_{t=1}^{T} p(t \mid \boldsymbol{x})\, \boldsymbol{R}_t. \quad (15)$$

When the posterior assigns a nonzero weight to a non-target adapter, the interference bound (Proposition 1) constrains interference from non-target adapters.

The router has two properties that distinguish it from learned alternatives. First, it has *no trainable parameters*: the means and weights are set analytically from K-means, and the covariance is a pooled scatter statistic. No gradient ever flows into the router, so it does not introduce

Table 1: Main results on T5-Large across both benchmarks and task orderings. AP: average performance (%)↑. FM: forgetting measure

	SuperNI				Long Sequence			
	Order 1		Order 2		Order 3		Order 4	
Method	AP↑	FM↓	AP↑	FM↓	AP↑	FM↓	AP↑	FM↓
LFPT5	39.03	10.88	29.71	20.72	66.62	14.51	67.40	13.11
EWC	16.91	27.11	20.12	28.37	47.71	20.14	52.19	31.17
LWF	19.34	24.74	30.34	17.63	50.73	19.17	47.94	33.22
TaSL	25.72	19.92	28.09	18.20	71.37	6.20	72.91	6.03
KIFLoRA	28.31	17.21	30.31	16.27	71.34	7.13	73.21	6.58
SeqLoRA	8.64	48.41	7.17	47.22	48.86	29.78	30.46	44.97
IncLoRA	13.08	42.26	16.39	36.51	62.75	12.75	61.76	15.35
O-LoRA	27.91	17.23	33.35	11.11	71.31	3.41	71.60	4.16
InfLoRA	39.02	8.02	39.97	8.22	75.02	3.01	75.35	2.39
GainLoRA	46.77	2.14	47.25	2.12	78.21	0.72	76.26	1.14
Latent-LoRA	**48.60**	**0.01**	**49.75**	**0.01**	**79.95**	**0.57**	**79.87**	**0.73**

an additional source of forgetting and requires no auxiliary protection mechanism. Second, it is *incremental*: adding a new task requires only fitting K means on the new task's embeddings and updating the shared covariance, with no need to revisit or replay data from earlier tasks.

4 Experiments

4.1 Setup

Benchmarks. We evaluate on two continual learning benchmarks for LLMs. **SuperNI** (Wang et al., 2022) covers diverse NLP tasks including dialogue generation, information extraction, question answering, summarization, and sentiment analysis. Following (Zhao et al., 2024), three tasks are selected from each category, yielding 15 tasks arranged into two orderings (Orders 1 and 2). **Long Sequence** (Razdaibiedina et al., 2023) consists of 15 classification tasks spanning natural language inference, sentiment analysis, topic classification, and question answering, also arranged into two orderings (Orders 3 and 4). We adopt the same task selections and orderings as GainLoRA (Liang and Li, 2025) and O-LoRA (Wang et al., 2023) to ensure direct comparability. Full task lists are provided in Appendix C.

Metrics. We report two standard metrics. **Average Performance (AP)** is the mean score across all tasks after training on the final task, and **Forgetting Measure (FM)** measures the mean drop from each task's peak score to its final score:

$$\mathrm{AP} = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}, \tag{16}$$

$$\mathrm{FM} = \frac{1}{T-1}\sum_{j=1}^{T-1}\Big(\max_{i\geq j} a_{i,j} - a_{T,j}\Big), \tag{17}$$

where T is the total number of tasks and $a_{i,j}$ is the model's performance on task j after training on task i.

Baselines. We compare against a range of continual learning methods. These include regularization-based approaches such as EWC (Kirkpatrick et al., 2017) and LWF (Li and Hoiem, 2017), the prompt-based method LFPT5 (Qin and Joty, 2022), and several LoRA-based methods: SeqLoRA, which sequentially fine-tunes a single adapter without forgetting mitigation; IncLoRA (Wang et al., 2023); O-LoRA (Wang et al., 2023); InfLoRA (Liang and Li, 2024); KIFLoRA (Feng et al., 2025); TaSL (Feng et al., 2024); and GainLoRA (Liang and Li, 2025). Following prior work (Wang et al., 2023; Liang and Li, 2025), we focus on the task-agnostic setting (Zeng et al., 2019) where task identities are unavailable at inference, and all methods operate without access to replay data from previous tasks.

Implementation details. Latent-LoRA is model-agnostic and applicable to any transformer-based architecture. Following existing CL works (Wang

et al., 2023; Liang and Li, 2025; Zhao et al., 2024), all methods use instruction tuning (Wei et al., 2022) and are optimized with AdamW (Loshchilov and Hutter, 2019). To ensure fair comparisons, all LoRA-based baselines incorporate adapters into the query and value projections of each transformer block (Vaswani et al., 2017) with rank r=8, following the protocol of prior work (Wang et al., 2023; Liang and Li, 2025). Our method uses rank r=32. We evaluate across both encoder-decoder (Raffel et al., 2020) and decoder-only (Touvron et al., 2023; Grattafiori et al., 2024) architectures at multiple scales. Each experiment is repeated three times with different seeds and the average is reported. Details on learning rates, batch sizes, adapter training, and GMM fitting are provided in Appendix C.3.

4.2 Main Results

Table 1 presents the main results on T5-Large across both benchmarks and task orderings. Our method achieves the highest AP and the lowest FM on all four configurations. On SuperNI, where tasks span diverse NLP categories including generation and classification, we outperform the previous best method (GainLoRA) by 1.8–2.5 points in AP while reducing forgetting to near zero. On Long Sequence, we improve over GainLoRA by 1.7–3.6 points in AP with forgetting below 1%.

Notably, methods that sum adapters at inference, such as O-LoRA and InfLoRA, suffer from substantially higher forgetting, particularly on SuperNI, where task diversity makes adapter interference more severe. GainLoRA mitigates this through learned gating modules, but at the cost of additional trainable parameters and a GPM-based protection mechanism. Our method achieves stronger results without any learned routing component and with fewer parameters per task.

Among non-LoRA baselines, LFPT5 performs competitively on SuperNI Order 1 but degrades on other configurations, while regularization-based methods (EWC, LWF) and sequential approaches (SeqLoRA) exhibit high forgetting across the board, suggesting that parameter-level regularization alone is insufficient for long task sequences.

4.3 Scaling to Larger Architectures

To assess whether our method generalizes beyond T5-Large, we evaluate on four additional model scales spanning both encoder-decoder and decoder-only architectures. Table 2 reports results on T5-XLarge and Table 3 on three Llama variants. Latent-LoRA consistently outperforms baselines across all scales, with the performance gap widening on larger models. On Llama-2-13B, we surpass GainLoRA by over 3 points in AP while maintaining near-zero forgetting.

Table 2: Results on T5-Xlarge. AP (%)↑ / FM (%)↓.

	SuperNI		Long Sequence	
Method	Order 1	Order 2	Order 3	Order 4
O-LoRA	37.91 / 10.12	41.22 / 6.36	74.98 / 1.63	76.63 / 3.10
GainLoRA	51.29 / 3.13	50.86 / 2.12	81.21 / **0.55**	79.91 / **0.81**
Latent-LoRA	**52.64 / 0.01**	**53.71 / 0.01**	**81.61** / 0.74	**80.39** / 1.00

Table 3: Results on Llama models, SuperNI. AP (%)↑ / FM (%)↓.

Model	Method	Order 1	Order 2
Llama-2-7B	O-LoRA	40.13 / 10.09	41.23 / 6.12
	GainLoRA	52.11 / 2.14	50.96 / 2.19
	LatentLoRA	**54.03 / 0.01**	**54.31 / 0.01**
Llama-3-8B	O-LoRA	42.91 / 9.19	42.33 / 5.71
	GainLoRA	54.17 / 1.91	53.47 / 2.10
	LatentLoRA	**56.09 / 0.01**	**56.33 / 0.01**
Llama-2-13B	O-LoRA	44.17 / 10.52	43.12 / 9.96
	GainLoRA	54.93 / 1.98	53.17 / 2.27
	LatentLoRA	**56.21 / 0.01**	**56.72 / 0.01**

4.4 Ablation Study

We ablate our system on T5-Large to evaluate the compact adapter and the GMM router independently.

Adapter. Table 4 compares three configurations under sum-at-inference (no router): (i) O-LoRA, using standard LoRA with orthogonal regularization on the down-projection matrices; (ii) our compact adapter without orthogonal regularization (λ=0) labeled as Latent-LoRA*; and (iii) our compact adapter with Σ-weighted orthogonal regularization (Eq. (8)). Adding the Σ-weighted orthogonal penalty substantially improves the compact adapter, allowing it to surpass O-LoRA on most orderings while using far fewer trainable parameters.

Router. Figure 3 evaluates the GMM router's accuracy. For each task, we compute the posterior probability assigned to the correct task, $p(t_{\text{correct}} \mid \boldsymbol{x})$, across all test inputs. Each box plot summarizes this distribution over a fixed-size subsample. On SuperNI, the router assigns near-perfect probability to the correct task for nearly all inputs, consistent with the cluster separation in Figure 2. On Long Sequence, most tasks are routed with high confidence, though a

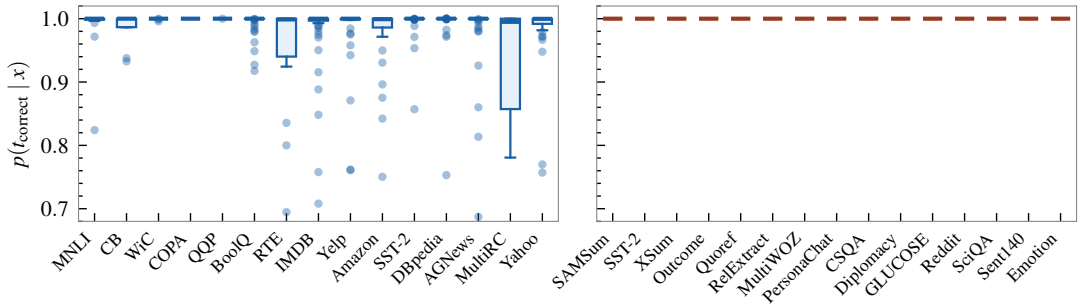


Figure 3: Router posterior probability of the correct task across test inputs for each task, evaluated on the embeddings of T5-Large. Left: Long Sequence: high confidence on most tasks with some variation. Right: SuperNI: near-perfect routing across all tasks.

Table 4: Ablation: adapter architecture under sum-at-inference (no router), T5-Large. AP↑ / FM↓.

	SuperNI		Long Sequence	
Method	1	2	3	4
O-LoRA	27.91 / 17.23	33.35 / 11.11	**71.31** / 3.41	71.60 / 4.16
LatentLoRA*	16.71 / 30.11	23.91 / 23.44	57.31 / 19.34	59.21 / 18.71
LatentLoRA	**33.30 / 12.94**	**35.90 / 8.83**	70.12 / **3.03**	**73.19/ 3.97**

few tasks show a wider spread. Results across all five model scales are reported in Appendix D.

4.5 Parameter Efficiency

Standard LoRA-based methods allocate $r(m+n)$ trainable parameters per target module, where m and n are the weight dimensions. With r=8, this amounts to 2.36M parameters per task on T5-Large and grows to 6.55M on Llama-2-13B. Our compact adapters require only r^2 parameters per module, a quantity independent of model dimension. With r=32, each task adds just 147K parameters on T5-Large and 82K on Llama-2-13B, a reduction of 16× and 80× respectively. Full comparison across all models is in Appendix C.3 This gap widens with model scale, making the approach increasingly attractive for larger architectures. The GMM router adds negligible training cost: fitting K means and updating a shared covariance takes seconds per task, with no gradient computation.

During inference, routing requires only the embedding pass, which is already computed as part of the model's forward pass. This is followed by Mahalanobis distance evaluations against $K \times T$ Gaussian components and a softmax. This is a single vector operation on the pooled embedding, substantially cheaper than the per-task MLP forward passes required by learned gating approaches. The blended adapter $\boldsymbol{R}(\boldsymbol{x})$ is then inserted into each attention projection, adding the same overhead as a single LoRA forward pass.

5 Limitations

While our method achieves strong performance with minimal overhead, several limitations should be acknowledged. First, the shared covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$ must be inverted each time a new task is added, with d being the embedding dimension of the base model. For models with large embedding dimensions (e.g., d=4096 for Llama-2-7B or d=5120 for Llama-2-13B), this inversion has $O(d^3)$ cost and requires careful numerical regularization. In practice, we find the Cholesky-based inversion with the regularization term $\epsilon \boldsymbol{I}$ to be stable across all models tested, but the cost grows cubically with embedding dimension and could become a concern for future models with substantially larger embedding spaces.

Second, the router relies on the assumption that task distributions are separable in the pooled embedding space. While this holds across all five model scales and both benchmarks in our experiments, tasks with highly overlapping input distributions (e.g., two sentiment analysis tasks differing only in domain) may not be reliably distinguished by the GMM. The router's soft blending provides graceful degradation in such cases, but the ceiling on routing accuracy is ultimately set by the separability of the frozen embeddings.

Finally, while we demonstrate strong results across five model scales up to 13B parameters and observed numbers improving with model size, we have not evaluated on models significantly larger than this. Whether the frozen embedding separability and compact adapter capacity hold at scales beyond 13B is an empirical question that warrants further investigation.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 7319–7328.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. 2019. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263.

Reza Rahimi Azghan, Gautham Krishna Gudur, Mohit Malu, Edison Thomaz, Giulia Pedrielli, Pavan Turaga, and Hassan Ghasemzadeh. 2026. Gated adaptation for continual learning in human activity recognition. *Preprint*, arXiv:2603.10046.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2025. Lora-xs: Low-rank adaptation with extremely small number of parameters. *Preprint*, arXiv:2405.17604.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *Preprint*, arXiv:2405.09673.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized language models. In *Advances in Neural Information Processing Systems*.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Yujie Feng, Xu Chu, Yongxin Xu, Zexin Lu, Bo Liu, Philip S. Yu, and Xiao-Ming Wu. 2025. Kif: Knowledge identification and fusion for language model continual learning. *Preprint*, arXiv:2408.05200.

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. 2024. TaSL: Continual dialog state tracking via task skill localization and consolidation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1266–1279, Bangkok, Thailand. Association for Computational Linguistics.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Tinne Tuytelaars, Matthias Bethge, and Joost van de Weijer. 2023. FeCAM: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In *Advances in Neural Information Processing Systems*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

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. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 40, pages 2935–2947.

Yan-Shuo Liang and Wu-Jun Li. 2024. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yan-Shuo Liang and Wu-Jun Li. 2025. Gated integration of low-rank adaptation for continual learning of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. Svft: Parameter-efficient fine-tuning with singular vectors. In *Advances in Neural Information Processing Systems*, volume 37, pages 41425–41446.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yun Luo, Zhen Yang, Xuefeng Bai, Fandong Meng, Jie Zhou, and Yue Zhang. 2023. Investigating forgetting in pre-trained representations through continual learning. *arXiv preprint arXiv:2305.05968*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Saleh Momeni, Sahil Bhatt, Zixuan Cao, and Bing Liu. 2025. Continual learning using a kernel-based method over foundation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Chengwei Qin and Shafiq Joty. 2022. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*.

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

Douglas A Reynolds. 2009. Gaussian mixture models. In *Encyclopedia of Biometrics*, pages 659–663. Springer.

Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey. *arXiv preprint arXiv:2404.16789*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671, Singapore. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. 2019. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1:364–372.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. *Preprint*, arXiv:2401.08295.

A Proofs

A.1 Interference Decomposition

We derive Eq. (7) from the main text. Recall that each task's output perturbation is given by:

$$\delta_j(\boldsymbol{h}) = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h}.$$

The inner product of the perturbations from tasks i and j is:

$$\begin{aligned}\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h}) &= \left(\boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_i \boldsymbol{V}_r^\top \boldsymbol{h}\right)^\top \left(\boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h}\right) \\ &= \boldsymbol{h}^\top \boldsymbol{V}_r \boldsymbol{R}_i^\top \boldsymbol{\Sigma}_r^\top \boldsymbol{U}_r^\top \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h} \\ &= \boldsymbol{h}^\top \boldsymbol{V}_r \boldsymbol{R}_i^\top \boldsymbol{\Sigma}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h} \qquad (18)\end{aligned}$$

where step (18) uses $\boldsymbol{U}_r^\top \boldsymbol{U}_r = \boldsymbol{I}_r$ (since $\boldsymbol{U}_r$ has orthonormal columns) and $\boldsymbol{\Sigma}_r^\top = \boldsymbol{\Sigma}_r$ (since $\boldsymbol{\Sigma}_r$ is a diagonal matrix with real entries).

Substituting the definitions $\psi(\boldsymbol{h}) = \boldsymbol{V}_r^\top \boldsymbol{h}$ and $\tilde{\boldsymbol{R}}_t = \boldsymbol{\Sigma}_r \boldsymbol{R}_t$:

$$\begin{aligned}\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h}) &= \boldsymbol{h}^\top \boldsymbol{V}_r \boldsymbol{R}_i^\top \boldsymbol{\Sigma}_r \boldsymbol{\Sigma}_r \boldsymbol{R}_j \boldsymbol{V}_r^\top \boldsymbol{h} \\ &= \left(\boldsymbol{V}_r^\top \boldsymbol{h}\right)^\top \left(\boldsymbol{\Sigma}_r \boldsymbol{R}_i\right)^\top \left(\boldsymbol{\Sigma}_r \boldsymbol{R}_j\right) \left(\boldsymbol{V}_r^\top \boldsymbol{h}\right) \\ &= \psi(\boldsymbol{h})^\top \, \tilde{\boldsymbol{R}}_i^\top \, \tilde{\boldsymbol{R}}_j \, \psi(\boldsymbol{h}). \qquad (19)\end{aligned}$$

Note that the high-dimensional factors $\boldsymbol{U}_r \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V}_r \in \mathbb{R}^{n \times r}$ cancel entirely: $\boldsymbol{U}_r$ vanishes through the orthonormality condition, and $\boldsymbol{V}_r$ is absorbed into the r-dimensional projection $\psi(\boldsymbol{h})$. The resulting expression depends only on the small $r \times r$ adapter matrices and the singular values.

A.2 Interference Bound

We prove Proposition 1: for all inputs $\boldsymbol{h}$,

$$\left|\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h})\right| \;\le\; \left\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j\right\|_2 \;\cdot\; \|\psi(\boldsymbol{h})\|^2.$$

Proof. From the interference decomposition (Appendix A.1), we have,

$$\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h}) = \psi(\boldsymbol{h})^\top \boldsymbol{M} \, \psi(\boldsymbol{h}),$$

where $\boldsymbol{M} = \tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j \in \mathbb{R}^{r \times r}$.

For any matrix $\boldsymbol{M}$ and vector $\boldsymbol{u}$, the following standard inequality holds:

$$\left|\boldsymbol{u}^\top \boldsymbol{M} \, \boldsymbol{u}\right| \;\le\; \|\boldsymbol{M}\|_2 \, \|\boldsymbol{u}\|^2,$$

where $\|\boldsymbol{M}\|_2 = \sup_{\|\boldsymbol{v}\|=1} \|\boldsymbol{M}\boldsymbol{v}\|$ is the spectral norm (largest singular value) of $\boldsymbol{M}$. This follows from Cauchy–Schwarz:

$$\begin{aligned}\left|\boldsymbol{u}^\top \boldsymbol{M} \, \boldsymbol{u}\right| &\le \|\boldsymbol{u}\| \cdot \|\boldsymbol{M} \, \boldsymbol{u}\| \\ &\le \|\boldsymbol{u}\| \cdot \|\boldsymbol{M}\|_2 \, \|\boldsymbol{u}\| \\ &= \|\boldsymbol{M}\|_2 \, \|\boldsymbol{u}\|^2. \qquad (20)\end{aligned}$$

Applying this with $\boldsymbol{u} = \psi(\boldsymbol{h})$ and $\boldsymbol{M} = \tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j$:

$$\left|\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h})\right| \;\le\; \left\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j\right\|_2 \;\cdot\; \|\psi(\boldsymbol{h})\|^2.$$

Since $\psi(\boldsymbol{h}) = \boldsymbol{V}_r^\top \boldsymbol{h}$ and $\boldsymbol{V}_r$ is fixed from the pretrained SVD, the factor $\|\psi(\boldsymbol{h})\|^2$ depends only on the input and the pretrained model's right singular vectors. It is independent of the adapter parameters and cannot be increased by training. Therefore, as the regularization loss $\mathcal{L}_{\text{ortho}}$ drives $\|\tilde{\boldsymbol{R}}_i^\top \tilde{\boldsymbol{R}}_j\|_2 \to 0$, the interference $|\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h})|$ is driven to zero uniformly over all inputs $\boldsymbol{h}$. □

B Compact Adapters vs. Standard LoRA

This appendix provides additional analysis of the compact latent-space adapter introduced in Section 3.2, contrasting its regularization properties with those of standard LoRA and discussing the role of parameter capacity in continual learning.

B.1 Incomplete Regularization in Standard LoRA

In standard LoRA, each task's weight update takes the form $\Delta \boldsymbol{W}_t = \boldsymbol{B}_t \boldsymbol{A}_t$, where both $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are trainable. The interference between two tasks' perturbations is,

$$\delta_i(\boldsymbol{h})^\top \delta_j(\boldsymbol{h}) = \boldsymbol{h}^\top \boldsymbol{A}_i^\top \boldsymbol{B}_i^\top \boldsymbol{B}_j \boldsymbol{A}_j \boldsymbol{h}. \qquad (21)$$

Existing continual learning methods such as O-LoRA (Wang et al., 2023) penalize $\|\boldsymbol{A}_i^\top \boldsymbol{A}_j\|$ to encourage orthogonality between the down-projection subspaces. However, the up-projections $\boldsymbol{B}_i, \boldsymbol{B}_j$ remain unconstrained. Because these factors enter the interference multiplicatively, even perfect orthogonality in $\boldsymbol{A}$ does not eliminate interference: the $\boldsymbol{B}_i^\top \boldsymbol{B}_j$ term can still take arbitrary values. No uniform bound over all inputs analogous to Proposition 1 can be established for standard LoRA.

In the compact parameterization of Section 3.2, the input-facing projection ($\boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top$, analogous to $\boldsymbol{A}$) and the output-facing projection ($\boldsymbol{U}_r$, analogous to $\boldsymbol{B}$) are both frozen from the pretrained SVD. The trainable $\boldsymbol{R}_t$ is the sole degree of freedom, allowing the bound in Proposition 1 to cover the complete interference expression.

B.2 Compact Capacity as Implicit Regularization

Each compact adapter operates with r^2 free parameters within a fixed subspace, compared to $r(m+n)$

in standard LoRA. Recent work suggests that restricting the trainable parameter space reduces forgetting: Biderman et al. (2024) attribute LoRA's lower forgetting relative to full fine-tuning to its reduced capacity, Aghajanyan et al. (2021) show that effective fine-tuning occurs in a low-dimensional intrinsic subspace, and Azghan et al. (2026) observe that structured parameter-efficient tuning improves cross-task consistency. Our parameterization pushes this further by confining each adapter to an r^2-dimensional space anchored to the pretrained principal subspace. This benefit is most apparent when combined with the router, which limits the number of adapters contributing to each input.

C Experimental Setup

C.1 Benchmark Details

Long Sequence Benchmark. The Long Sequence benchmark (Razdaibiedina et al., 2023) consists of 15 classification tasks drawn from established NLP datasets. Table 5 summarizes each task's category, domain, and evaluation metric. All tasks are evaluated using accuracy.

SuperNI Benchmark. The SuperNI benchmark (Wang et al., 2022) consists of 15 tasks spanning five NLP categories: dialogue generation, information extraction, question answering, summarization, and sentiment analysis. Following Zhao et al. (2024), three tasks are selected from each category. Table 6 lists the tasks and their evaluation metrics. Classification tasks are evaluated using accuracy; all others use Rouge-L.

C.2 Task Orderings

Table 7 lists the task sequences used in all experiments. Orders 1 and 2 correspond to the SuperNI benchmark; Orders 3 and 4 correspond to the Long Sequence benchmark. These orderings follow those used by O-LoRA (Wang et al., 2023) and GainLoRA (Liang and Li, 2025).

C.3 Implementation Details

Model architectures. We evaluate on five model scales: T5-Large (770M parameters), T5-XLarge (3B), Llama-2-7B, Llama-3-8B, and Llama-2-13B. For all models, LoRA adapters are applied to the query and value projections of each attention layer. T5-Large contains 144 target modules (24 encoder self-attention + 24 decoder self-attention + 24 decoder cross-attention, each with query and value projections). T5-XL has the same architecture with larger projection dimensions (4096×1024). Llama models contain 64 target modules for 7B/8B (32 layers $\times$ 2) and 80 for 13B (40 layers $\times$ 2).

Adapter configuration. All LoRA-based baselines use rank r=8 and α=16. Our method uses rank r=32 and α=16. The per-task trainable parameter counts are summarized in Table 8.

Training. All methods are optimized with AdamW (Loshchilov and Hutter, 2019). For T5 models, we use a learning rate of 3×10^{-4} and a batch size of 8. For Llama models, we use a learning rate of 5×10^{-5} and a batch size of 4. All experiments use a constant learning rate schedule. Each task is trained for 30 epochs on T5 and 15 epochs on Llama. The orthogonal regularization coefficient is set to $\lambda = 0.05$ for SuperNI and $\lambda = 0.02$ for Long Sequence, selected based on the relative magnitude of the Σ-weighted ortho loss to the task loss on a held-out validation split of T5-Large.

GMM router. The router uses K=5 Gaussian components per task, initialized via K-means with 30 iterations. The shared covariance regularization coefficient is $\epsilon = 0.01$. At inference, we use soft routing (Eq. 15). The GMM fitting step takes approximately 2–5 seconds per task on CPU and does not require GPU resources.

Evaluation. For classification tasks, we report exact-match accuracy. For generation tasks (SuperNI), we report Rouge-L (Lin, 2004) computed using the `rouge_score` library with stemming enabled. During evaluation, outputs are generated using greedy decoding with a maximum generation length of 50 tokens.

Hardware. All experiments are conducted on NVIDIA A100 GPUs. T5-Large and T5-XL experiments run on a single GPU. Llama experiments use a single GPU with gradient checkpointing enabled. Each experiment is repeated three times with different random seeds, and the average is reported.

D Additional Visualizations

Figures 4 and 5 extend the t-SNE and router posterior analyses from the main text to three additional model scales. The patterns observed on T5-Large (Figures 2 and 3) hold consistently: SuperNI tasks form clearly separable clusters across all models, and Long Sequence tasks show adequate margins with occasional overlap among semantically similar tasks. Router posteriors remain near-perfect on

Table 5: Tasks in the Long Sequence benchmark.

Task	Category	Domain	Metric
Yelp	Sentiment analysis	Yelp reviews	Accuracy
Amazon	Sentiment analysis	Amazon reviews	Accuracy
IMDB	Sentiment analysis	Movie reviews	Accuracy
SST-2	Sentiment analysis	Movie reviews	Accuracy
DBpedia	Topic classification	Wikipedia	Accuracy
AG News	Topic classification	News	Accuracy
Yahoo	Topic classification	Yahoo Q&A	Accuracy
MNLI	Natural language inference	Various	Accuracy
QQP	Paraphrase detection	Quora	Accuracy
RTE	Natural language inference	News, Wikipedia	Accuracy
CB	Natural language inference	Various	Accuracy
WiC	Word sense disambiguation	Lexical databases	Accuracy
COPA	Question answering	Blogs, encyclopedia	Accuracy
BoolQ	Boolean question answering	Wikipedia	Accuracy
MultiRC	Question answering	Various	Accuracy

Table 6: Tasks in the SuperNI benchmark.

Task	Category	Metric
Task639 (MultiWOZ)	Dialogue generation	Rouge-L
Task1590 (Diplomacy)	Dialogue generation	Rouge-L
Task1729 (PersonaChat)	Dialogue generation	Rouge-L
Task181 (Outcome extraction)	Information extraction	Rouge-L
Task748 (GLUCOSE)	Information extraction	Rouge-L
Task1510 (Relation extraction)	Information extraction	Rouge-L
Task002 (Quoref)	Question answering	Rouge-L
Task073 (CommonsenseQA)	Question answering	Rouge-L
Task591 (SciQA)	Question answering	Rouge-L
Task511 (Reddit TIFU)	Summarization	Rouge-L
Task1290 (XSum)	Summarization	Rouge-L
Task1572 (SAMSum)	Summarization	Rouge-L
Task363 (SST-2)	Sentiment analysis	Accuracy
Task875 (Emotion)	Sentiment analysis	Accuracy
Task1687 (Sentiment140)	Sentiment analysis	Accuracy

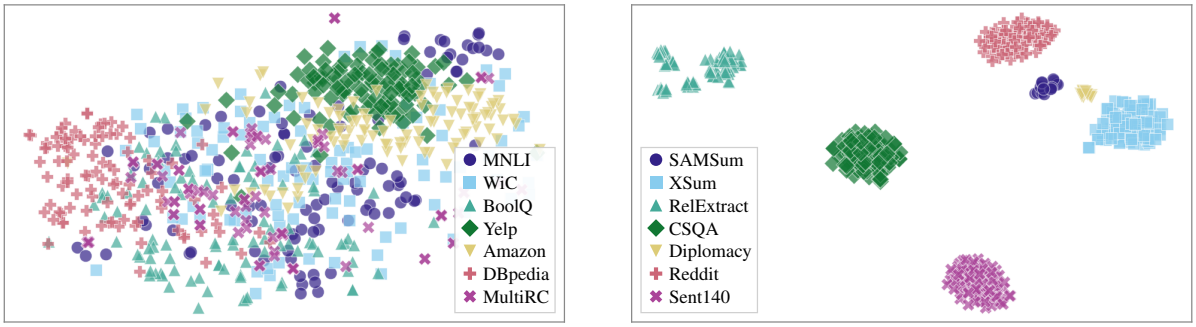
Table 7: Task orderings used in all experiments.

Order	Task sequence
1 (SuperNI)	Task1572 → Task363 → Task1290 → Task181 → Task002 → Task1510 → Task639 → Task1729 → Task073 → Task1590 → Task748 → Task511 → Task591 → Task1687 → Task875
2 (SuperNI)	Task748 → Task073 → Task1590 → Task639 → Task1572 → Task1687 → Task591 → Task363 → Task1510 → Task1729 → Task181 → Task511 → Task002 → Task1290 → Task875
3 (Long Seq.)	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
4 (Long Seq.)	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC

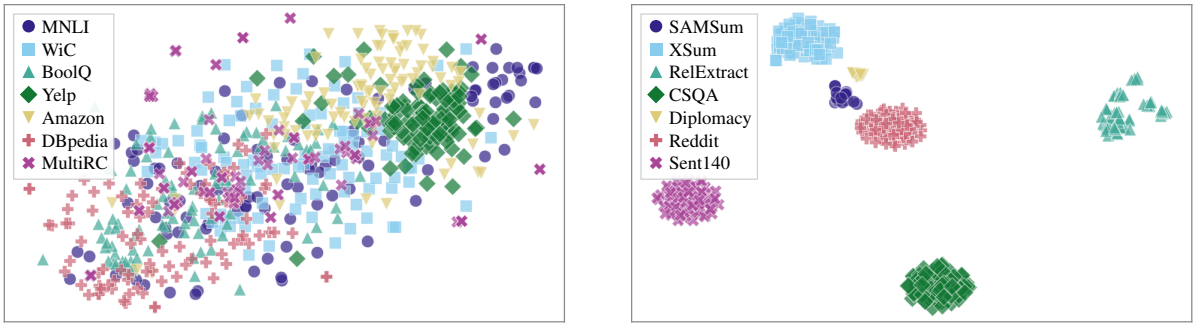
Table 8: Per-task trainable parameters across models.

Model	LoRA (r=8)	Ours (r=32)	Reduction
T5-Large	2,359K	147K	16×
T5-XL	5,898K	147K	40×
Llama-2-7B	4,194K	66K	64×
Llama-3-8B	4,194K	66K	64×
Llama-2-13B	6,554K	82K	80×

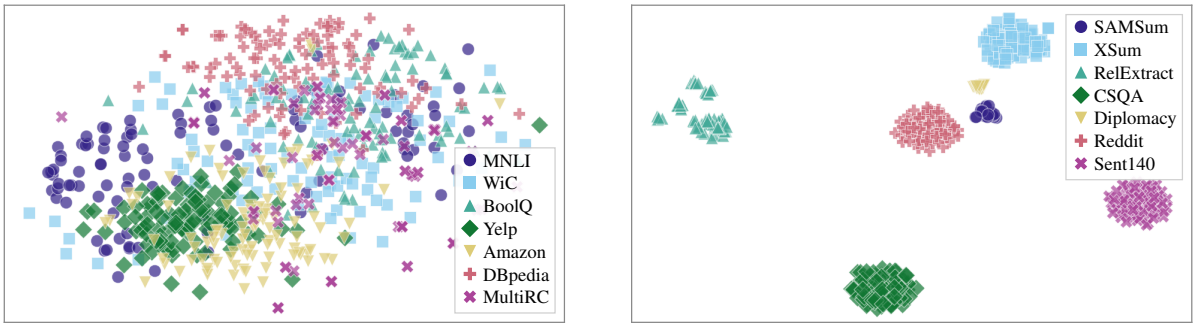
SuperNI and high-confidence on Long Sequence regardless of model family or scale.



(a) T5-XL

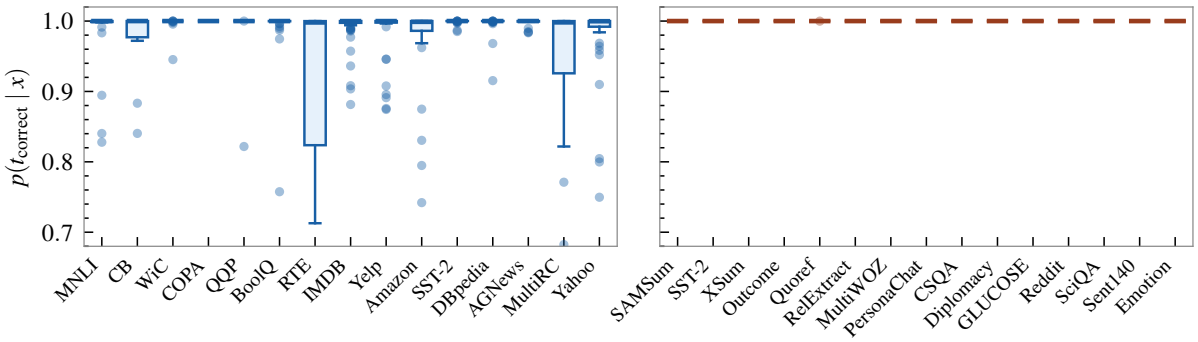


(b) Llama-2-7B

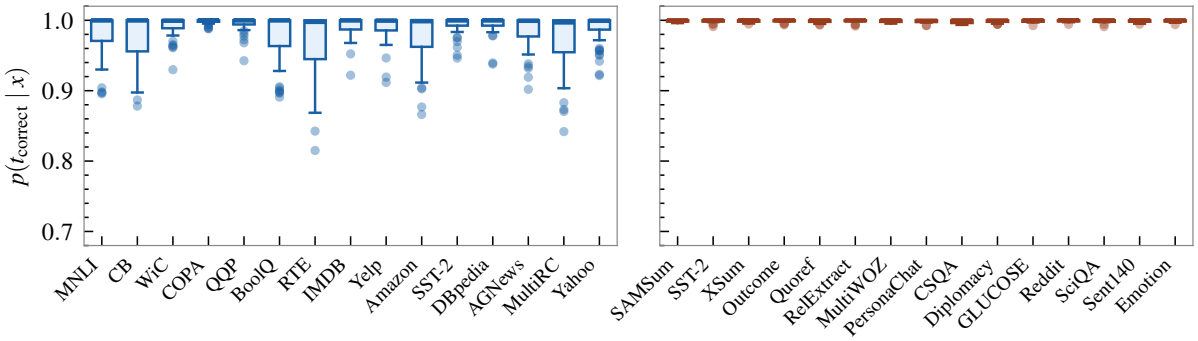


(c) Llama-3-8B

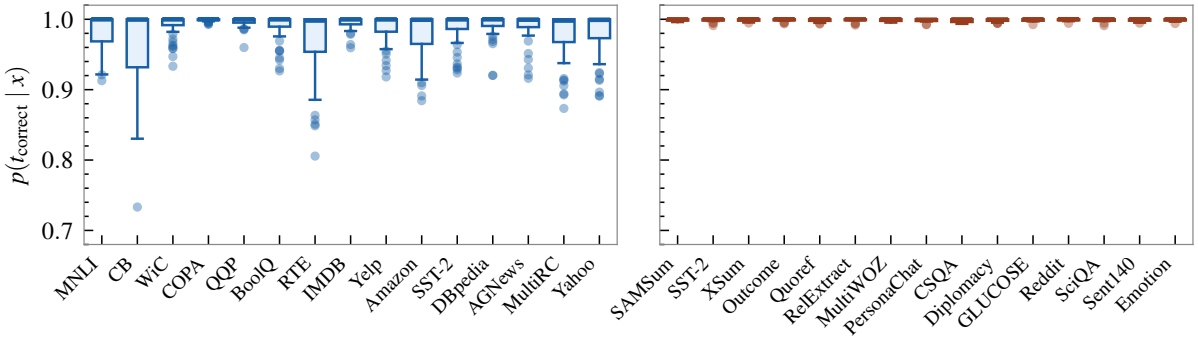
Figure 4: t-SNE projections of pooled token embeddings from frozen embedding layers. Left: Long Sequence. Right: SuperNI.



(a) T5-XL



(b) Llama-2-7B



(c) Llama-3-8B

Figure 5: Router posterior probability of the correct task across test inputs. Left: Long Sequence. Right: SuperNI.