SPACE-LoRA: Allocating Activation-Subspace Protection for Continual Learning

Anonymous authors
Paper under double-blind review

Abstract

This study addresses the catastrophic forgetting problem that occurs when sequentially learning successive tasks using Low-Rank Adaptation (LoRA) from a lifelong learning perspective. While existing approaches have primarily constrained parameter updates or learning subspaces to reduce interference with past knowledge, they have not fully considered additive interference. This occurs when a newly added residual adapter on top of a fixed past model generates non-zero responses along input directions important for old tasks, thereby altering previous predictions. To this end, we propose Subspace Protection with Allocated Capacity for Efficient Continual Adaptation (SPACE-LoRA). SPACE-LoRA directly suppresses the responses of the new residual branch along input activation directions that are important for old tasks and adaptively determines the protection coverage for each module based on past-task sensitivity estimated via a common Fisher sensitivity-based coverage target. Under a fixed LoRA rank, this approach adaptively adjusts module-specific protection coverage while suppressing interference along input directions sensitive to old tasks. We assess the effectiveness of activation-subspace protection in mitigating catastrophic forgetting and examine the role of sensitivity-guided protection in continual learning across diverse tasks. Code is available at https://anonymous.4open.science/r/SPACE-LoRA-7864.

1 INTRODUCTION

Continual learning (CL) requires a model to sequentially learn new tasks while retaining knowledge acquired from old tasks, demanding a balance between stability and plasticity (Parisi et al., 2019). Parameter-efficient fine-tuning (PEFT), particularly Low-Rank Adaptation (LoRA), enables CL by learning low-rank updates while keeping the pretrained backbone frozen (Hu et al., 2021). Existing LoRA-based methods address inter-task interference by separating the subspace for new task updates. O-LoRA uses an orthogonality regularization to separate current and previous LoRA subspaces (Wang et al., 2023), whereas InfLoRA determines the LoRA input factor from the interference-free subspace orthogonal to previous tasks before learning each task and keeps the projection fixed during optimization (Liang & Li, 2024). More recent methods further refine this approach. SplitLoRA determines the minor subspace dimension for each module through a first-order bound on aggregated old task gradients, balancing the old gradient energy falling into the subspace against the dimensionality available for new task learning (Qiu et al., 2026). JANUS-LoRA uses gradient rectification to decouple the LoRA factor updates, and reconstructs them through regularized least squares so that their composite weight update approximately follows the intended orthogonal direction (Chen et al., 2026).

However, freezing previously learned parameters does not guarantee that old task behavior is preserved. Even when these parameters remain unchanged, a newly learned LoRA branch can induce non-zero residual outputs on old task activations, which propagate to downstream representations and predictions. Existing subspace-based methods reduce this interference by restricting new task updates relative to subspaces derived from old tasks. However, in these methods, the new branch either still responds to old task activations or stops responding only because one LoRA factor is frozen. Moreover, increasing protection reduces the subspace available for new task adaptation, creating a stability-plasticity trade-off in determining the protected subspace size. Since historical sensitivity from the accumulated tasks is distributed differently across activation directions in different modules, a fixed protection size may protect some modules not enough while unnecessarily constraining

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.

others. Moreover, sizing the protected subspace from aggregate old task gradients does not directly reflect this sensitivity associated with protected directions.

To address these limitations, we propose Subspace Protection with Allocated Capacity for Efficient Continual Adaptation (SPACE-LoRA), which directly constrains the input factor of each new task LoRA branch with respect to selected old task activation directions and independently selects the protection size of each module. For each module, SPACE-LoRA selects the minimum number of directions needed to reach the target coverage of past-task sensitivity. Rather than using directions to determine the adaptation space for the new task, we directly constrain the LoRA input factor to the orthogonal complement of the protected subspace by projecting it after every optimizer step, while keeping both LoRA factors trainable. As a result, the response of the newly added LoRA branch is identically zero for any activation within the protected subspace, regardless of the output factor. Meanwhile, the remaining orthogonal complement remains available for new task adaptation. Because increasing the size of the protected subspace necessarily reduces the space available for new task adaptation, the protection capacity is chosen per module. Old task activations determine the candidate protection directions and their ordering within each module. Fisher information measures how much historical task sensitivity is captured as these directions are included. This coverage-based decision of SPACE-LoRA allows the protection dimensionality to vary according to the sensitivity distribution of each module. We evaluate SPACE-LoRA with a single coverage target across vision and language benchmarks under two training protocols, a wider range of setting than the methods we compare with. Mostly, SPACE-LoRA ranks first or close to first on every benchmark, whereas the strongest baseline changes from benchmark to benchmark.

Our main contributions are summarized as follows:

- We formulate continual low-rank adaptation as a residual-response control problem and introduce a hard activation-subspace constraint that directly projects the input factor of each new LoRA branch. This constraint guarantees zero residual response on the selected historical activation subspace while leaving the complementary input space available for new-task adaptation.
- We introduce a Fisher-coverage scheme that adapts the protection size to each module's sensitivity to previous tasks. By allocating stronger protection to modules that require greater preservation while avoiding unnecessary protection in less sensitive modules, SPACE-LoRA balances knowledge retention and adaptation capacity across the network.
- We evaluate SPACE-LoRA on both vision and language models across a diverse set of continual learning benchmarks and against a broad range of competitive baselines. The results show that SPACE-LoRA consistently alleviates catastrophic forgetting while preserving adaptation to incoming tasks, thereby improving the trade-off between stability and plasticity across different model architectures and task domains.

2 RELATED WORK

Continual learning (CL) studies how models learn from sequential tasks while retaining previously acquired knowledge from earlier tasks, and the main difficulty is catastrophic forgetting (De Lange et al., 2021). CL is categorized as task-incremental learning, where the task identity is given, class-incremental learning, where the model chooses among all classes seen, and domain-incremental learning, where the input distribution changes while the label space stays fixed (Van de Ven et al., 2022). Existing CL methods are commonly categorized into regularization-based, memory-based, and expansion-based approaches (Wang et al., 2024). Parameter-efficient fine-tuning (PEFT), which freezes a pretrained model and trains only a small number of additional parameters, can reduce the computational cost of adapting to new downstream tasks while maintaining competitive performance. Representative PEFT approaches include adapter-based tuning, which augments a frozen backbone with smaller and trainable modules (Houlsby et al., 2019), and prompt-based tuning, which optimizes learnable prompts while leaving the pretrained parameters fixed (Lester et al., 2021; Wang et al., 2022). Low-Rank Adaptation (LoRA) keeps the pretrained model frozen and parametrizes updates to selected weight matrices with two low-rank factors (Hu et al., 2021). It has been increasingly adopted in CL methods that mitigate interference among sequential task updates.

O-LoRA reduces task interference by placing low-rank updates for different tasks in orthogonal parameter subspaces (Wang et al., 2023). InfLoRA constructs the input factor from a subspace designed to avoid interference with old tasks and keeps the factor fixed while learning the new task (Liang & Li, 2024). PLAN pre-assigns orthogonal basis vectors to each task and selects those least sensitive to interference with previously learned parameters (Wang et al., 2025). LB-CL (Qiao & Mahdavi, 2024) uses the sensitivity of low-rank parameters to identify knowledge-specific components and incorporates them into gradient projection. SplitLoRA partitions the old task gradient space into principal and minor subspaces and adjusts the partition size across modules to balance stability and plasticity (Qiu et al., 2026). JANUS-LoRA targets zero change in the layer output on previous activations (Chen et al., 2026). It projects the full-weight update onto the complement of a historical activation subspace and, because independent factor updates break this projection. reconstruct the factor updates by regularized least squares so that their composite update approximately follows the projected direction.

These LoRA-based approaches are related to projection-based CL methods that restrict new updates with respect to subspaces associated with old task knowledge. GPM constructs important subspaces from old task representations, determining the size based on a layer-wise energy threshold, and projects new gradients onto their orthogonal complements (Saha et al., 2021). Energy-based rank selection has also shown its effectiveness in LoRA-based CL (Li et al., 2026). Adam-NSCL estimates a null space from old task input-feature covariance and constrains subsequent optimization to that space by projecting the optimizer update (Wang et al., 2021). Regularization-based methods such as EWC (Kirkpatrick et al., 2017) quantify parameter importance for old tasks using the Fisher information and restrict changes to parameters that are important for retaining old task knowledge. A separate line of work, such as AdaLoRA and OA-Adapter, allocate adaptation capacity selectively across components to improve parameter efficiency (Zhang et al., 2023; Wan et al., 2025).

In subspace-based methods such as GPM and SplitLoRA, the protection size is determined by representation energy or averaged old task gradients, neither of which measures the per-example loss sensitivity along the activation directions that are actually protected. SPACE-LoRA treats selected old-task activation directions as a protected subspace and projects the input factor onto its orthogonal complement after each optimizer step, while keeping both LoRA factors trainable. As a result, the newly added LoRA branch produces zero response to activations within the protected subspace. In each module, historical activation directions are ordered by decreasing activation eigenvalue, and the protection size is selected as the minimum number of directions required to reach the target Fisher coverage.

3 PRELIMINARIES

3.1 LoRA-based Continual Learning

Consider a task stream $\mathcal{T} = \{t_1, \ldots, t_T\}$. The objective is to learn new tasks while retaining performance on previously learned tasks (Zhou et al., 2024). We consider a setting in which task-specific LoRA adapters are accumulated (Wang et al., 2023). After learning task t, the effective weight of module m combines its pre-trained weight $W_m^0 \in \mathbb{R}^{d_{\text{out},m} \times d_{\text{in},m}}$ with the accumulated adapters:

$$W_m^{(t)} = W_m^0 + \eta \sum_{\tau=1}^{t} B_{m,\tau} A_{m,\tau}, \qquad \eta = \frac{\alpha}{r}, \tag{1}$$

where r denotes the LoRA rank, α is the scaling hyperparameter (Hu et al., 2021), and

$$A_{m,t} \in \mathbb{R}^{r \times d_{\text{in},m}}, \qquad B_{m,t} \in \mathbb{R}^{d_{\text{out},m} \times r}. \tag{2}$$

During training on task t, only $A_{m,t}$ and $B_{m,t}$ within module m are updated, while W_m^0 and all adapters from previous tasks remain frozen.

3.2 Adapter Interference and Catastrophic Forgetting

Even with previously learned adapters frozen, a new adapter can change the output of a module. For a fixed input activation $x \in \mathbb{R}^{d_{\text{in},m}}$ from a previous task, the output change induced by the new adapter at module m is

$$\Delta h_m^{(t)}(x) = \eta B_{m,t} A_{m,t} x. \tag{3}$$

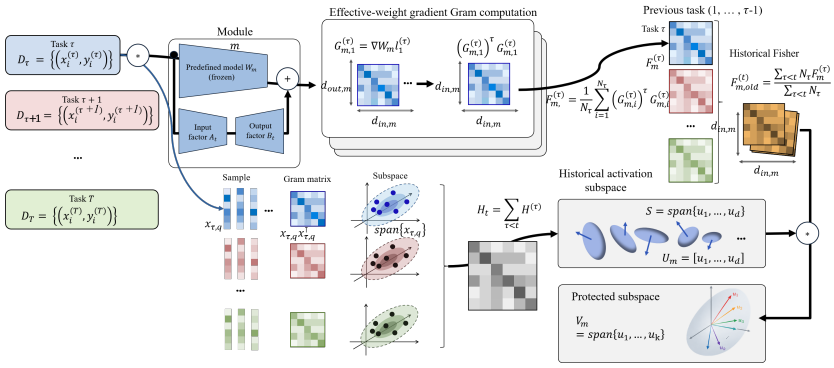


Figure 1: Overview of SPACE-LoRA.

This expression isolates the new adapter's contribution at a fixed module input. Changes in preceding modules may additionally alter the input activation itself. Because each new adapter is optimized for the current task, its contribution may disrupt predictions on previous tasks. Such changes can lead to catastrophic forgetting. Let $a_{j,j}$ denote the accuracy of task j immediately after learning it, and let $a_{t,j}$ denote its accuracy after learning through task t $(t > j)$. Task-wise forgetting is defined as

$$F_j^{(t)} = a_{j,j} - a_{t,j}. \tag{4}$$

Thus, $F_j^{(t)} > 0$ indicates a decline in accuracy on task j following subsequent adaptation.

4 SPACE-LoRA

We propose SPACE-LoRA as shown in Figure 1. SPACE-LoRA directly suppresses the response of a newly added residual adapter along selected past activation directions, while determining the protection size of each module using a common Fisher sensitivity coverage target. Under a fixed LoRA rank, this design suppresses interference along input directions that are sensitive to previous tasks while leaving the complementary input subspace available for learning the new task, thereby aiming to achieve a better trade-off between knowledge retention and adaptation. In the subsequent sections, we describe how to construct the protected activation subspace and how to determine the required protection size for each module.

4.1 Residual Response Nullification

As discussed in Section 3, even when all previously learned adapters are frozen, a newly added residual branch can alter past predictions if it produces a non-zero response to past-task inputs. Therefore, continual adaptation should protect not only previously learned parameters but also the responses of the new adapter to input directions associated with old tasks. Recent LoRA-based continual learning methods address this issue by constraining new-task adaptation with respect to subspaces associated with previously learned tasks. InfLoRA constructs a low-interference adaptation subspace by removing current activation components aligned with historical gradient subspaces, whereas Janus-LoRA constrains the effective weight update to produce no change on historical activations. Motivated by these approaches, we instead directly constrain the response of the newly introduced residual branch by making its input factor insensitive to selected historical activation directions.

At the endpoint of each task τ, we collect the input activations of each linear module m over the task training set D_τ using the model obtained after completing the task. Before learning task t, these activations are summarized by the Historical Gram matrix

$$H_m^{<t} = \sum_{\tau<t} \sum_{x \in D_\tau} \sum_{q} x_{m,q} x_{m,q}^\top, \tag{5}$$

where $x_{m,q}$ denotes the activation vector at the q-th input position of module m. We decompose

$$H_m^{<t} = U_m \Lambda_m U_m^\top, \tag{6}$$

where the eigenvectors in $U_m = [u_{m,1}, \ldots, u_{m,d_m}]$ are ordered by decreasing eigenvalue. These eigenvectors identify the principal input directions occupied by past activations. Protecting the full basis would leave no input direction available to the new adapter. We therefore protect only the first k_m activation directions and define

$$V_m = [u_{m,1}, \ldots, u_{m,k_m}], \qquad P_m = V_m V_m^\top. \tag{7}$$

Let A_m denote the input factor of the new LoRA branch. After each optimizer update, we apply the hard projection

$$A_m \leftarrow A_m(I - P_m), \tag{8}$$

which maintains

$$A_m V_m = 0. \tag{9}$$

Hence, for any $x \in \mathrm{span}(V_m)$ with $x = V_m c$,

$$\Delta W_m x = B_m A_m V_m c = 0. \tag{10}$$

Thus, the newly added branch produces no residual response along the protected historical directions, while the orthogonal complement remains available for learning the new task. This is a hard parameter constraint rather than a soft orthogonality regularizer. A detailed characterization of its exact nullification property and its behavior outside the protected subspace is provided in Appendix A.

4.2 Module-wise Fisher-Coverage Protection

The number of protected activation directions need not be identical across modules, since their importance to previously learned tasks can differ substantially. Using a fixed protection size may therefore either leave sensitive directions insufficiently protected or unnecessarily reduce the input space available for new-task adaptation. We instead determine k_m independently for each module as the minimum number of activation-ordered directions required to capture a prescribed fraction of its historical loss sensitivity.

To quantify this sensitivity, after completing task τ, we compute the gradient of each training example with respect to the effective weight of module m,

$$G_{m,i}^{(\tau)} = \nabla_{W_m} \ell_i^{(\tau)} \in \mathbb{R}^{d_{\mathrm{out},m} \times d_{\mathrm{in},m}}, \tag{11}$$

and define the input-space empirical Fisher

$$F_m^{(\tau)} = \frac{1}{N_\tau} \sum_{i=1}^{N_\tau} \left(G_{m,i}^{(\tau)}\right)^\top G_{m,i}^{(\tau)}. \tag{12}$$

This statistic is computed at the endpoint of task τ using the current task classifier and the observed ground-truth labels. The historical Fisher available before task t is

$$F_{m,\mathrm{past}}^{(t)} = \frac{\sum_{\tau<t} N_\tau F_m^{(\tau)}}{\sum_{\tau<t} N_\tau}. \tag{13}$$

Further details on the construction and accumulated historical Fisher are provided in Appendix B.

Let $U_m = [u_{m,1}, \ldots, u_{m,d_m}]$ be the full activation basis from the previous subsection, ordered by decreasing activation eigenvalue, where $d_m = d_{\mathrm{in},m}$. We measure the historical sensitivity of each activation direction as

$$p_{m,j}^{(t)} = u_{m,j}^\top F_{m,\mathrm{past}}^{(t)} u_{m,j}. \tag{14}$$

The activation statistics therefore determine the directions and their order, while the Fisher evaluates the loss sensitivity associated with those directions. For a prefix of length k, we define its Fisher coverage as

$$C_m(k) = \frac{\sum_{j=1}^{k} p_{m,j}^{(t)}}{\mathrm{tr}\left(F_{m,\mathrm{past}}^{(t)}\right)}. \tag{15}$$

Since U_m is a complete orthonormal basis, $\mathrm{tr}(F_{m,\mathrm{past}}^{(t)}) = \sum_{j=1}^{d_m} p_{m,j}^{(t)}$, so $C_m(k)$ represents the fraction of total historical sensitivity captured by the first k activation directions. Given a common coverage target ρ, we select

$$k_m = \min\left\{k \in \{0, \ldots, d_m - r\} : C_m(k) \geq \rho\right\}. \tag{16}$$

Algorithm 1 Overview of SPACE-LoRA

Require: Task datasets $\{D_t\}_{t=1}^{T}$, LoRA rank r, coverage target ρ

for $t = 1, \ldots, T$ **do**
  Freeze the base model and all previously learned adapters
  Add a new rank-r LoRA branch for task t
  for each protected module m **do**
    if $t = 1$ **then**
      Set $k_m \leftarrow 0$ and $P_m \leftarrow 0$
    else
      Obtain the activation-ordered basis from $H_m^{<t}$
      Determine k_m using the Fisher-coverage target ρ
      Construct $V_m = [u_{m,1}, \ldots, u_{m,k_m}]$ and $P_m = V_m V_m^\top$
    end if
  end for
  for each optimization step on D_t **do**
    Update the new LoRA branch using the current-task loss
    Project $A_{m,t} \leftarrow A_{m,t}(I - P_m)$ for each protected module m
  end for
  Update the historical activation and Fisher statistics using D_t
end for

The upper bound $d_m - r$ leaves at least r input dimensions available to the rank-r adapter. If the total sensitivity is zero, we set $k_m = 0$. If no admissible prefix reaches the coverage target, we set $k_m = d_m - r$, corresponding to the maximum allowable protection size. The resulting first k_m activation directions are protected using the hard response constraint in Section 4.1, and the protected subspace remains fixed throughout the current task.

4.3 Update Procedure

Algorithm 1 summarizes the overall training procedure of SPACE-LoRA. Before learning a new task, each module constructs its protected activation basis from the historical activation statistic and determines the protection size using the Fisher-coverage rule in Section 4.2. The resulting protected subspace is then fixed throughout the current task. A new LoRA branch is trained while its input factor is projected onto the orthogonal complement of the protected subspace after each optimizer update, as described in Section 4.1. After completing the task, the endpoint activation and Fisher statistics are computed and incorporated into the historical statistics used for subsequent tasks. The first task is trained without protection because no historical statistics are available. Algorithm 2 in the appendix provides the complete procedure, including the construction and incremental update of the historical statistics.

5 Empirical Study

5.1 Experimental Setup

Datasets and Baselines. We evaluate SPACE-LoRA on vision and language continual learning benchmarks. Our vision benchmarks comprise ImageNet-R, DomainNet, VTAB, and VDD. For language continual learning, we use a suite of 15 NLP classification datasets following the benchmark used by Progressive Prompts and O-LoRA (Razdaibiedina et al., 2023; Wang et al., 2023). We conduct systematic comparisons with representative LoRA-based continual learning methods: InfLoRA, SplitLoRA, and Janus-LoRA for vision, and these methods together with O-LoRA for language. The comparisons use a common experimental protocol within each benchmark. For methods originally developed for vision, their language experiments use T5 adaptations based on the official implementations. The adaptation details are provided in the appendix.

Experimental Design. We evaluate ImageNet-R, DomainNet, and VTAB under class-incremental learning (CIL), where prediction is performed over all classes observed so far without access to the task identity at inference. For ImageNet-R, we follow the task-count configurations used by Qiu

Table 1: Class-incremental learning results on ImageNet-R.

	ImageNet-R					
	5 Tasks		10 Tasks		20 Tasks	
Method	Acc. ↑	F ↓	Acc. ↑	F ↓	Acc. ↑	F ↓
Joint-LoRA	82.68	-	82.68	-	82.68	-
InfLoRA	78.89 ± 0.55	6.02	75.90 ± 0.22	6.28	$\mathbf{72.23 \pm 0.58}$	6.82
SplitLoRA	74.46 ± 0.22	10.80	70.25 ± 0.32	7.98	59.84 ± 0.25	**4.81**
Janus-LoRA	$\mathbf{79.47 \pm 0.47}$	6.41	$\mathbf{76.07 \pm 0.45}$	7.70	72.15 ± 0.36	8.22
SPACE-LoRA ($\rho = 0.90$)	79.09 ± 0.31	**4.82**	74.64 ± 0.44	6.42	69.65 ± 0.50	6.80
SPACE-LoRA ($\rho = 0.95$)	78.98 ± 0.33	4.90	74.33 ± 0.51	**5.98**	69.33 ± 0.31	5.34

et al. (2026) and partition its 200 classes into 5, 10, and 20 tasks, with 40, 20, and 10 classes per task, respectively. For DomainNet, we use five domains excluding Quickdraw and partition its 345 classes into five tasks of 69 classes each. For VTAB, we adopt the 5-task cross-domain construction introduced by APER, with 10 classes per task. For VDD, we use a task-incremental learning (TIL) setting: ImageNet is excluded, and each of the remaining nine datasets constitutes a separate task, with task identity available at inference. For language continual learning, each dataset constitutes one task. We consider sequences of 5, 10, and 15 tasks, constructed from the corresponding prefixes of a fixed 15-dataset order. Shared tasks use the same data examples across these configurations. Evaluation uses the candidate label set of the dataset being evaluated, rather than a joint label space spanning all datasets. We use pretrained ViT-B/16 and T5-large backbones for the vision and language experiments, respectively. Detailed dataset statistics, data splits, task sequences, and implementation settings are provided in Appendix C.

Evaluation Metrics and Implementation Details. We evaluate performance using average accuracy, $\mathrm{ACC}_t = \frac{1}{t}\sum_{j=1}^{t} a_{t,j}$. Class-incremental evaluation considers all seen classes without task identity, whereas task-incremental evaluation uses task identity to select the corresponding head. For vision experiments, we use ViT-B/16 pretrained on ImageNet-21K, with rank-10 LoRA adapters applied to the key and value projections in all 12 transformer blocks. All vision methods follow the same task splits and training protocol within each benchmark. We train for 50 epochs per task on ImageNet-R and VTAB, and 5 epochs on DomainNet. For VDD, we train for 1,500 optimization steps per task. For comparisons with the baselines, we report SPACE-LoRA with Fisher coverage thresholds of $\rho = 0.90$ and 0.95. For language experiments, we use T5-large as the primary backbone and follow the training settings of O-LoRA, including the number of training epochs, batch size, and learning rate.

5.2 Main Results

Results on ImageNet-R. We first evaluate SPACE-LoRA on ImageNet-R, where successive tasks introduce disjoint subsets of its classes. Table 1 reports the results for three task partitions of ImageNet-R. On ImageNet-R, both SPACE-LoRA configurations achieve slightly higher mean final accuracy and lower forgetting than InfLoRA in the 5-task setting, with $\rho = 0.90$ yielding the lowest forgetting among continual learning methods. In the 10-task setting, $\rho = 0.95$ achieves the lowest forgetting, whereas $\rho = 0.90$ exhibits slightly more forgetting than InfLoRA. However, both configurations achieve lower final accuracy than InfLoRA and Janus-LoRA in the 10- and 20-task settings. These results highlight the importance of balancing old task protection with new task adaptation.

Results on DomainNet, VTAB, and VDD. We consider benchmarks encompassing multiple visual domains or distinct classification tasks. Table 2 shows that SPACE-LoRA achieves the lowest forgetting on DomainNet with $\rho = 0.95$, while achieving slightly lower final accuracy than Janus-LoRA. On VTAB, SPACE-LoRA with $\rho = 0.90$ achieves the highest accuracy of 86.05% and the lowest forgetting of 5.94 percentage points. On VDD, SPACE-LoRA reaches 86.26% accuracy with $\rho = 0.90$, outperforming other baselines. Both coverage settings outperform Janus-LoRA and Split-

Table 2: Continual learning performance on DomainNet, VTAB, and VDD. DomainNet and VTAB use class-incremental evaluation, whereas VDD uses task-incremental evaluation.

	DomainNet: 5 Tasks		VTAB: 5 Tasks		VDD: 9 Tasks	
Method	Acc. ↑	F ↓	Acc. ↑	F ↓	Acc. ↑	F ↓
InfLoRA	72.44	7.81	85.52	6.74	80.88	**1.86**
SplitLoRA	67.51	9.81	73.35	19.67	64.78	3.73
Janus-LoRA	**73.41**	7.82	80.47	10.59	78.53	4.16
SPACE-LoRA ($\rho = 0.90$)	73.26	6.06	**86.05**	**5.94**	**86.26**	3.32
SPACE-LoRA ($\rho = 0.95$)	73.19	**5.54**	85.20	6.51	80.41	2.66

Table 3: Continual learning performance on the 5-, 10-, and 15-task language benchmarks.

	5 Tasks		10 Tasks		15 Tasks	
Method	Acc. ↑	F ↓	Acc. ↑	F ↓	Acc. ↑	F ↓
O-LoRA	48.76	27.70	54.56	22.21	37.52	32.48
InfLoRA	71.68	**-0.90**	70.83	**0.48**	67.54	3.91
SplitLoRA	61.60	0.40	58.72	0.54	54.85	**0.50**
Janus-LoRA	70.96	-0.75	66.97	2.96	63.08	2.26
SPACE-LoRA ($\rho = 0.90$)	**71.88**	-0.10	**73.95**	0.98	**69.79**	2.53

LoRA in accuracy on VTAB and VDD. These results show that SPACE-LoRA achieves competitive performance across benchmarks involving multiple visual domains or heterogeneous task sequences.

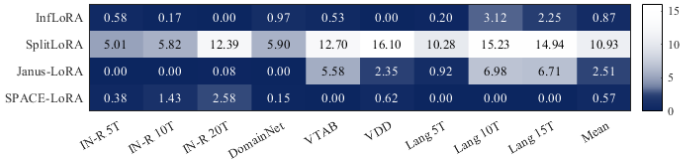


Figure 2: Gap to the best method on each benchmark (pp). Darker indicate a smaller gap.

Results on Language Continual Learning. Table 3 reports results on the 5-, 10-, and 15-task language benchmarks. SPACE-LoRA achieves the highest final accuracy in the 5- and 10-task settings, reaching 71.88% and 73.95%, respectively, with no forgetting on average in the 5-task setting and 0.98 points of forgetting in the 10-task setting. In the 15-task setting, SPACE-LoRA achieves 69.79% accuracy, exceeding InfLoRA by 2.25 points, with lower forgetting (2.53 versus 3.91 points). Although SPACE-LoRA does not achieve the lowest forgetting in any setting, it maintains competitive final accuracy across all three task sequences. These results complement the heterogeneous-task evaluation on VTAB and VDD with continual adaptation across distinct text classification tasks. Figure 2 shows that SPACE-LoRA stays close to the best method on every benchmark, whereas the strongest baseline changes from benchmark to benchmark.

5.3 Effect of Protection Coverage

The Fisher coverage threshold ρ controls how much historical information is covered by the protected activation subspace. We examine $\rho \in \{0.75, 0.80, 0.90, 0.95, 0.99\}$ on the 5-task ImageNet-R benchmark while keeping the remaining training configuration fixed. In addition to final accuracy and forgetting, we report the average protected dimension $\bar{k}$. Here, $\bar{k}$ averages the protected dimensions across modules and incremental tasks, excluding the first task.

Table 4 illustrates the trade-off between retaining previous knowledge and preserving capacity for subsequent adaptation. As ρ increases from 0.75 to 0.99, the average protected dimension grows from 90 to 517, while forgetting decreases overall from 7.41 to 3.50 percentage points. Final accuracy does not improve monotonically: it increases to 79.09% at $\rho = 0.90$ and then decreases to

Table 4: Effect of Fisher coverage on 5-task ImageNet-R.

ρ	Acc. ↑	F ↓	$\bar{k}$
0.75	78.12 ± 0.26	7.41	90
0.80	78.35 ± 0.37	7.34	113
0.90	$\mathbf{79.09 \pm 0.31}$	4.82	211
0.95	78.98 ± 0.33	4.90	322
0.99	78.30 ± 0.34	**3.50**	517

Table 5: Protection rule ablation on 5-task ImageNet-R and 10-task language continual learning with T5-large, with $\rho = 0.90$ for Fisher-based protection.

	ImageNet-R: 5 Tasks		Language: 10 Tasks (T5-large)	
Protection rule	Acc. ↑	F ↓	Acc. ↑	F ↓
Fisher (SPACE-LoRA)	$\mathbf{79.09 \pm 0.31}$	4.82	**73.95**	0.98
Uniform	76.97	**4.40**	72.61	**-0.12**

78.30% at $\rho = 0.99$. Thus, stronger protection can improve retention without producing a corresponding improvement in final predictive performance.

5.4 Ablation on Protection Allocation

We next examine the role of Fisher information in determining module-wise protection. We compare Fisher-guided protection with uniform allocation on 5-task ImageNet-R using ViT-B/16 and on the 10-task language benchmark using T5-large. The Fisher-guided configuration uses $\rho = 0.90$. This ablation complements the coverage sweep by examining how the protection rule affects final accuracy and forgetting.

Table 5 shows that Fisher-guided protection achieves higher final accuracy than uniform allocation in both settings. On ImageNet-R, it achieves 79.09% accuracy compared with 76.97% for uniform allocation, an improvement of 2.12 percentage points, while exhibiting slightly higher forgetting (4.82 versus 4.40 points). On the language benchmark, Fisher-guided protection achieves 73.95% accuracy compared with 72.61% for uniform allocation, an improvement of 1.34 percentage points. These results support the effectiveness of Fisher-guided protection in improving final predictive performance over uniform allocation, while showing that these gains do not arise from lower forgetting.

6 Conclusion

In this work, we introduced SPACE-LoRA, a continual adaptation method that addresses catastrophic forgetting by suppressing the residual response of newly introduced LoRA branches. SPACE-LoRA projects the input factor of each new branch onto the orthogonal complement of selected historical activation directions, which guarantees zero residual response on the protected subspace while leaving the remaining directions for new task adaptation. A module-wise Fisher-coverage criterion further sets the protection size from the historical loss sensitivity along activation-ordered directions. Experiments on vision and language benchmarks show that SPACE-LoRA reduces forgetting while remaining competitive, and the coverage analysis shows the trade-off between protecting previous tasks and learning new ones. Although Fisher-coverage sizing does not consistently outperform uniform sizing, protecting important historical activation directions is effective for continual low-rank adaptation.

AI Use Statement

We used generative AI tools to assist with drafting and polishing the manuscript. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content.

REFERENCES

Cheng Chen, Pengpeng Zeng, Yuyu Guo, Lianli Gao, Hengtao Shen, and Jingkuan Song. Janus-lora: A balanced low-rank adaptation for continual learning. *arXiv preprint arXiv:2605.28495*, 2026.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Longhua Li, Lei Qi, Qi Tian, and Xin Geng. Energy-structured low-rank adaptation for continual learning. *arXiv preprint arXiv:2605.27482*, 2026.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23638–23647. IEEE, 2024.

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

Fuli Qiao and Mehrdad Mahdavi. Learn more, but bother less: parameter efficient continual learning. *Advances in Neural Information Processing Systems*, 37:97476–97498, 2024.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, Weili Guan, Min Zhang, and Liqiang Nie. Splitlora: Balancing stability and plasticity in continual learning through gradient space splitting. In *International Conference on Learning Representations*, volume 2026, pp. 48458–48480, 2026.

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

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

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Zhiyi Wan, Wanrou Du, Liang Li, Miao Pan, and Xiaoqi Qin. Adaptive budget allocation for orthogonal-subspace adapter tuning in llms continual learning. *arXiv preprint arXiv:2505.22358*, 2025.

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.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 184–193. IEEE, 2021.

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

Xiequn Wang, Zhan Zhuang, and Yu Zhang. Plan: Proactive low-rank allocation for continual learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2909–2918. IEEE, 2025.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 139–149. IEEE, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*, 2024.

A Formal Properties of the Hard Protection Constraint

This section provides a formal characterization of the hard input-space constraint used in SPACE-LoRA. We consider a linear module m whose newly introduced LoRA branch at task t is given by

$$\Delta h_m^{(t)}(x) = \eta B_{m,t} A_{m,t} x, \qquad \eta = \frac{\alpha}{r}, \tag{17}$$

where

$$A_{m,t} \in \mathbb{R}^{r \times d_m}, \qquad B_{m,t} \in \mathbb{R}^{o_m \times r}. \tag{18}$$

Let

$$V_m = [v_{m,1}, \ldots, v_{m,k_m}] \in \mathbb{R}^{d_m \times k_m}, \qquad V_m^\top V_m = I, \tag{19}$$

denote the protected basis selected for module m, and let

$$P_m = V_m V_m^\top \tag{20}$$

be the corresponding orthogonal projector. After each optimizer update, the input factor is projected onto the orthogonal complement of the protected subspace as

$$A_{m,t} \leftarrow A_{m,t}(I - P_m). \tag{21}$$

A.1 Exact Nullification on the Protected Subspace

The projection in Eq. equation 21 directly removes every component of $A_{m,t}$ that acts on the protected directions. Since $P_m = V_m V_m^\top$ and $V_m^\top V_m = I$,

$$A_{m,t}(I - P_m)V_m = A_{m,t}\left(V_m - V_m V_m^\top V_m\right) \tag{22}$$

$$= A_{m,t}(V_m - V_m) = 0. \tag{23}$$

Therefore, the projected input factor satisfies

$$A_{m,t} V_m = 0. \tag{24}$$

This property directly determines the response of the new residual branch to inputs contained in the protected subspace. For any

$$x \in \operatorname{span}(V_m), \tag{25}$$

there exists $c \in \mathbb{R}^{k_m}$ such that $x = V_m c$. The corresponding residual response is then

$$\Delta h_m^{(t)}(x) = \eta B_{m,t} A_{m,t} x \tag{26}$$

$$= \eta B_{m,t} A_{m,t} V_m c \tag{27}$$

$$= 0. \tag{28}$$

Thus, the new LoRA branch produces exactly zero response for inputs lying in the selected protected subspace, irrespective of the value of $B_{m,t}$.

This property follows directly from the parameter projection rather than from a soft regularization objective. It does not require a penalty term, a first-order approximation of the loss, or an assumption that the optimization gradient is approximately orthogonal to the protected directions. As long as Eq. equation 24 is maintained, the new branch cannot introduce an additional response to an input that lies entirely in $\mathrm{span}(V_m)$.

The scope of this result is local to the corresponding module and its input. It guarantees that the newly added branch produces no additional response when the module receives an input within the protected subspace. It does not require that every old task activation lie exactly in that subspace, nor does it imply that downstream activations remain unchanged after adaptation at earlier modules. Therefore, the module-level property should not by itself be interpreted as a guarantee of zero forgetting at the network level.

A.2 Interference Outside the Protected Subspace

For an arbitrary input $x \in \mathbb{R}^{d_m}$, the protected and unprotected components can be separated as

$$x = P_m x + (I - P_m)x. \tag{29}$$

Because Eq. equation 24 also gives

$$A_{m,t}P_m = A_{m,t}V_m V_m^\top = 0, \tag{30}$$

the new adapter responds only to the component outside the protected subspace,

$$A_{m,t}x = A_{m,t}(I - P_m)x. \tag{31}$$

Consequently,

$$\left\|\Delta h_m^{(t)}(x)\right\|_2 = |\eta| \left\|B_{m,t}A_{m,t}(I - P_m)x\right\|_2 \tag{32}$$

$$\leq |\eta| \left\|B_{m,t}\right\|_2 \left\|A_{m,t}\right\|_2 \left\|(I - P_m)x\right\|_2, \tag{33}$$

where the inequality follows from submultiplicativity of the spectral norm. For fixed adapter norms, the magnitude of the module-output perturbation is therefore controlled by how much of the input remains outside the protected subspace. Inputs contained entirely in the protected subspace give zero residual response, while inputs close to that subspace can affect the new branch only through their remaining orthogonal component.

The size of this residual component can be related to the activation statistics used to construct V_m. Let the accumulated historical activation Gram matrix be

$$H_m = \sum_i x_i x_i^\top = U_m \Lambda_m U_m^\top, \tag{34}$$

with eigenvalues ordered as

$$\lambda_{m,1} \geq \lambda_{m,2} \geq \cdots \geq \lambda_{m,d_m} \geq 0. \tag{35}$$

When

$$V_m = [u_{m,1}, \ldots, u_{m,k_m}], \tag{36}$$

the total historical activation energy remaining outside the protected subspace is

$$\sum_i \|(I - P_m)x_i\|_2^2 = \mathrm{tr}\left[(I - P_m)H_m\right] \tag{37}$$

$$= \sum_{j=k_m+1}^{d_m} \lambda_{m,j}. \tag{38}$$

Therefore, for a fixed protection dimension k_m, selecting the leading eigenvectors of the historical activation Gram matrix minimizes the total squared residual energy of past activations outside the protected subspace. The activation statistics thus determine the ordering of directions to be protected,

while the Fisher-based allocation described in the main text determines how many of these directions are retained for each module.

Equation 33 characterizes the residual response at the module level rather than task-level forgetting. Its magnitude also depends on the learned norms of $A_{m,t}$ and $B_{m,t}$, which may vary across tasks and protection configurations. In addition, an output perturbation introduced at an earlier module can alter the input received by subsequent modules. For this reason, a small residual activation energy outside the protected subspace does not directly translate into an equivalent bound on the final network output.

B Construction of Historical Statistics

SPACE-LoRA uses two historical statistics to construct and evaluate the protected input directions: an activation Gram matrix and an input-space empirical Fisher matrix. Both statistics are collected at the endpoint of each task and accumulated across previously observed tasks. During later tasks, the protection subspace and its allocation are determined from these cached statistics without revisiting the raw training examples of previous tasks.

B.1 Historical Activation Statistics

For each module m, we summarize the input activations observed at the endpoint of task τ using the uncentered Gram matrix

$$H_m^{(\tau)} = \sum_{i \in D_\tau} \sum_q x_{m,i,q} x_{m,i,q}^\top, \tag{39}$$

where $x_{m,i,q} \in \mathbb{R}^{d_m}$ denotes the input activation at the q-th valid input position of example i. The historical activation statistic available before task t is

$$H_m^{<t} = \sum_{\tau<t} H_m^{(\tau)}. \tag{40}$$

Hence, after completing task t, the cached activation statistic can be updated incrementally as

$$H_m^{<t+1} = H_m^{<t} + H_m^{(t)}, \tag{41}$$

without retaining activations or training examples from earlier tasks.

Then, We compute its eigendecomposition

$$H_m^{<t} = U_m \Lambda_m U_m^\top, \tag{42}$$

where the eigenvalues are ordered in descending order. The corresponding eigenvectors

$$U_m = [u_{m,1}, \ldots, u_{m,d_m}] \tag{43}$$

define the ordered candidate directions for protection.

Importantly, the activation statistic determines both the directions and their prefix order. SPACE-LoRA does not reorder these directions according to the Fisher sensitivity. The Fisher is instead used only to determine how far this activation-ordered prefix should be extended for each module. Given a protection dimension k_m, the protected basis is therefore

$$V_m = [u_{m,1}, \ldots, u_{m,k_m}]. \tag{44}$$

B.2 Historical Fisher Sensitivity

While the activation Gram determines the candidate protection directions, SPACE-LoRA uses an input-space empirical Fisher statistic to evaluate how sensitive each direction is to previously learned tasks.

For an example i from task τ, let

$$G_{m,i}^{(\tau)} = \nabla_{W_m} \ell_i^{(\tau)} \in \mathbb{R}^{o_m \times d_m} \tag{45}$$

denote the gradient of the supervised loss with respect to the effective weight of module m. At the endpoint of task τ, we compute this statistic using the current task classifier and the observed ground-truth labels. We define the task-level input-space empirical Fisher as

$$F_m^{(\tau)} = \frac{1}{N_\tau} \sum_{i=1}^{N_\tau} \left(G_{m,i}^{(\tau)}\right)^\top G_{m,i}^{(\tau)}. \tag{46}$$

This matrix corresponds to the input-dimensional partial trace of the observed-label empirical Fisher associated with the effective weight W_m. The historical Fisher available before task t is accumulated using the number of examples in each previous task:

$$F_{m,\text{past}}^{(t)} = \frac{\sum_{\tau<t} N_\tau F_m^{(\tau)}}{\sum_{\tau<t} N_\tau}. \tag{47}$$

In practice, this quantity can be maintained using the accumulated Fisher numerator together with the total number of examples, without storing task-specific training data.

For the j-th activation direction $u_{m,j}$, we define its historical sensitivity as

$$p_{m,j}^{(t)} = u_{m,j}^\top F_{m,\text{past}}^{(t)} u_{m,j}. \tag{48}$$

Equivalently, this quantity can be written as the average squared sensitivity along $u_{m,j}$ over all examples from previously observed tasks:

$$p_{m,j}^{(t)} = \frac{1}{\sum_{\tau<t} N_\tau} \sum_{\tau<t} \sum_{i=1}^{N_\tau} \left\| G_{m,i}^{(\tau)} u_{m,j} \right\|_2^2. \tag{49}$$

Since $F_{m,\text{past}}^{(t)}$ measures squared gradient responses along input directions, each directional sensitivity $p_{m,j}^{(t)}$ is nonnegative. Moreover, because $U_m = [u_{m,1}, \ldots, u_{m,d_m}]$ forms a complete orthonormal basis, the sensitivities over all activation directions sum to the total historical sensitivity:

$$\sum_{j=1}^{d_m} p_{m,j}^{(t)} = \sum_{j=1}^{d_m} u_{m,j}^\top F_{m,\text{past}}^{(t)} u_{m,j} \tag{50}$$

$$= \text{tr}\left(U_m^\top F_{m,\text{past}}^{(t)} U_m\right) \tag{51}$$

$$= \text{tr}\left(F_{m,\text{past}}^{(t)}\right). \tag{52}$$

Therefore, the cumulative sensitivity of the first k activation directions, $\sum_{j=1}^{k} p_{m,j}^{(t)}$, represents the amount of historical sensitivity captured by that activation-ordered prefix. This identity provides the basis for the Fisher-coverage ratio used in Section 4.2.

C Additional Experimental Details

C.1 Vision Experimental Setup

For ImageNet-R, the 5, 10, and 20 task settings use the same underlying image split and class ordering. This allows us to examine the effect of increasing the number of sequential learning stages while keeping the overall class set fixed.

For DomainNet, we use five domains consisting of clipart, infograph, painting, real, and sketch, while excluding Quickdraw. The 345 classes are divided into five disjoint tasks with 69 classes per task. Images from all five retained domains are included in every task.

For VTAB, we use a class incremental benchmark consisting of five datasets, Resisc45, DTD, Oxford IIIT Pets, EuroSAT, and Oxford Flowers102, in this order. Each dataset contributes 10 classes and constitutes a separate task. This results in five tasks and 50 classes in total. We use the officially released data and hold out 10% of the training images from each class as a validation set. This split yields 1,615 training images, 181 validation images, and 8,619 test images.

Table 6: Datasets used in the language continual learning experiments in their training order.

Step	Dataset	Source Suite	Task Type
1	AG News	Standard CL	Topic Classification
2	Amazon	Standard CL	Sentiment Analysis
3	Yelp	Standard CL	Sentiment Analysis
4	DBpedia	Standard CL	Topic Classification
5	Yahoo	Standard CL	Question Answering
6	MNLI	GLUE	Natural Language Inference
7	QQP	GLUE	Paraphrase Detection
8	RTE	GLUE	Natural Language Inference
9	SST-2	GLUE	Sentiment Analysis
10	IMDB	Independent	Sentiment Analysis
11	WiC	SuperGLUE	Word Sense Disambiguation
12	CB	SuperGLUE	Natural Language Inference
13	COPA	SuperGLUE	Question Answering
14	MultiRC	SuperGLUE	Question Answering
15	BoolQ	SuperGLUE	Boolean Question Answering

For VDD, we use nine of the ten datasets in the Visual Domain Decathlon and exclude ImageNet. The nine datasets are Aircraft, CIFAR 100, Daimler Pedestrian Classification, DTD, GTSRB, Omniglot, SVHN, UCF101, and VGG Flowers. Each dataset constitutes one task in this order, such that every task introduces both a new domain and a new label set. The resulting benchmark contains 2,128 classes in total, with the number of classes per task ranging from 2 to 1,623. Since the labels of the official test split are not publicly available, we use the official validation split as the test set and hold out 10% of the training images from each class for validation. This results in 156,598 training images, 17,178 validation images, and 64,439 test images.

C.2 Language Experimental Setup

We evaluate continual language adaptation using a sequence of 15 text classification and natural language understanding datasets. Table 6 presents the datasets in the order in which they are introduced to the model. The 5 task setting uses the first five datasets, the 10 task setting uses the first ten datasets, and the 15 task setting uses the complete sequence. Therefore, each longer setting extends the preceding sequence without changing the order of previously introduced tasks.

For most tasks, we construct a fixed dataset containing 1,000 training samples, 200 validation samples, and up to 500 test samples. The validation and test sets do not overlap whenever sufficient labeled data are available.

For GLUE and SuperGLUE datasets whose public test labels are unavailable, we construct evaluation splits from the corresponding labeled validation data. The resulting test sets for RTE and WiC contain 77 and 438 samples, respectively. For smaller datasets such as CB and COPA, we use the available labeled evaluation samples for performance measurement.

All experiments with T5 large use AdamW with a constant learning rate, a batch size of 8, and 3 epochs per task. We use LoRA with rank 8 and scaling parameter 16 for the query and value projections in all attention modules. A new adapter is introduced for each task, and the maximum input length is 128 tokens.

We select the learning rate once using the five task setting with seed 0 and use the selected value for all methods and sequence lengths. No method specific learning rate tuning is performed. All compared methods use the same data, optimizer, and training schedule, while their method specific hyperparameters follow the default settings of their official implementations.

C.3 Evaluation Protocol

The model is evaluated after completing each task in the sequence. We report average accuracy after the final task to measure overall predictive performance and backward transfer to measure changes

in performance on previously learned tasks. All methods are evaluated using the same task ordering, dataset splits, training schedule, and evaluation protocol.

Algorithm 2 Continual Adaptation with SPACE-LoRA

Require: Task datasets $\{D_t\}_{t=1}^{T}$, protected modules $\mathcal{M}$, LoRA rank r, coverage target ρ

Initialize $H_m^{<1} \leftarrow 0$ and $F_{m,\text{past}}^{(1)} \leftarrow 0$ for all $m \in \mathcal{M}$
Initialize the number of previous-task examples $N_{<1} \leftarrow 0$
for $t = 1, \ldots, T$ **do**
    Freeze the base model and all LoRA adapters learned from previous tasks
    Introduce a new rank-r LoRA branch $(A_{m,t}, B_{m,t})$ for each $m \in \mathcal{M}$
    for each module $m \in \mathcal{M}$ **do**
        if $t = 1$ **then**
            $k_m \leftarrow 0$, $P_m \leftarrow 0$
        else
            Compute $H_m^{<t} = U_m \Lambda_m U_m^\top$ with eigenvectors ordered by decreasing eigenvalue
            for $j = 1, \ldots, d_m$ **do**
                $p_{m,j}^{(t)} \leftarrow u_{m,j}^\top F_{m,\text{past}}^{(t)} u_{m,j}$
            end for
            if $\operatorname{tr}(F_{m,\text{past}}^{(t)}) = 0$ **then**
                $k_m \leftarrow 0$
            else
                Find the smallest $k \in \{0, \ldots, d_m - r\}$ satisfying

$$\frac{\sum_{j=1}^{k} p_{m,j}^{(t)}}{\operatorname{tr}\left(F_{m,\text{past}}^{(t)}\right)} \geq \rho$$

                if such k exists **then**
                    $k_m \leftarrow k$
                else
                    $k_m \leftarrow d_m - r$
                end if
            end if
            $V_m \leftarrow [u_{m,1}, \ldots, u_{m,k_m}]$
            $P_m \leftarrow V_m V_m^\top$
        end if
    end for
    for each optimization step on D_t **do**
        Compute the current-task loss and update the trainable parameters
        for each module $m \in \mathcal{M}$ **do**
            Project the input factor $A_{m,t} \leftarrow A_{m,t}(I - P_m)$
        end for
    end for
    for each module $m \in \mathcal{M}$ **do**
        Collect endpoint activations on D_t and compute

$$H_m^{(t)} \leftarrow \sum_{i \in D_t} \sum_{q} x_{m,i,q} x_{m,i,q}^\top$$

        Using the current task classifier and ground-truth labels, compute $G_{m,i}^{(t)} \leftarrow \nabla_{W_m} \ell_i^{(t)}$ for each $i \in D_t$
        Compute

$$F_m^{(t)} \leftarrow \frac{1}{N_t} \sum_{i=1}^{N_t} \left(G_{m,i}^{(t)}\right)^\top G_{m,i}^{(t)}$$

        Update the historical activation statistic $H_m^{<t+1} \leftarrow H_m^{<t} + H_m^{(t)}$
        Update the historical Fisher

$$F_{m,\text{past}}^{(t+1)} \leftarrow \frac{N_{<t} F_{m,\text{past}}^{(t)} + N_t F_m^{(t)}}{N_{<t} + N_t}$$

    end for
    $N_{<t+1} \leftarrow N_{<t} + N_t$
end for