When One Adapter Speaks for Many: Discovering Low-Rank Redundancy in Continual Fine-Tuning

Tanguy Dieudonné [1] **Giulia Lanzillotta** [1] **Enis Simsar** [1] **Louis Barinka** [1] **Thomas Hofmann** [1]

Abstract

Low-Rank Adaptation (LoRA) has become the standard tool for parameter-efficient fine-tuning of large pretrained models. When applied sequentially across tasks in Continual Learning (CL), the standard assumption is that each new task requires a dedicated low-rank adapter. In this work, we challenge this assumption empirically and structurally. We show that task-specific LoRA adapters in CL exhibit significant *low-rank redundancy*: the subspaces spanned by adapters trained on different tasks substantially overlap, and in many cases earlier adapters can faithfully represent later tasks. Building on this observation, we propose **LITELORA**, a *plug-and-play* gating mechanism that learns at train time whether to recruit a new adapter or reuse existing low-rank representations. Our method reduces the number of active adapters by 20–70% while matching or exceeding state-of-the-art performance on standard CL benchmarks, revealing that structural redundancy is pervasive and that selective learning is sufficient to achieve stability without sacrificing plasticity.

1. Introduction

In the era of foundation models, the need for effective and *efficient* adaptation is ever growing. As pretraining costs scale dramatically with model size, repeatedly retraining models from scratch to incorporate new data is increasingly impractical. This motivates a shift toward *lifecycle-aware adaptation*, where models are incrementally updated over time. Parameter Efficient Fine Tuning (PEFT) have emerged as a key mechanism for enabling such updates at low cost. Instead of updating all backbone weights, PEFT methods introduce lightweight modules, such as low-rank adapters (LoRA) (Hu et al., 2021), allowing foundation models to acquire new capabilities while limiting computational and memory overhead. Despite their reduction in costs, PEFT methods still face a fundamental challenge in sequential fine-tuning settings: simply acquiring new information may overwrite existing knowledge and degrade overall model performance, a phenomenon called *Catastrophic Forgetting* (CF).

Continual Learning (CL) algorithms (Hadsell et al., 2020) address CF by ensuring the *stability* of past knowledge during adaptation. In the context of Low-Rank Adaptation, the prevalent strategy is to impose active constraints on adapters to minimize cross-task interference. For instance, O-LoRA (Wang et al., 2023) orthogonalizes adapters across tasks, while InfLoRA (Liang & Li, 2024) aligns the adapter's row space with the task input span. While effective at preserving stability, these constraints inevitably reduce the model's *plasticity*—its ability to learn new tasks—as the available optimization space becomes increasingly restricted over time. Conversely, recently Wu et al. (2025) showed that decoupling the magnitude and direction of updates reduces forgetting without restricting the parameter space, proposing a constraint-free method named SD-LoRA. While this preserves plasticity, the lack of constraints can lead to unbounded forgetting as the task sequence grows. Consequently, existing methods force a binary choice between rigidity (high stability, low plasticity) and drift (high plasticity, low stability).

Our work is motivated by a simple empirical question: do different tasks in a CL stream require genuinely distinct low-rank subspaces, or do they share latent structure? We show empirically that the answer is strongly the latter. We propose **LITELORA**, a method that learns a differentiable binary gate to determine, for each new task, whether existing adapters already span the relevant low-rank subspace or whether a new one is needed. Built on top of SD-LoRA (Wu et al., 2025), LITELORA uses 20–70% fewer adapters while matching or exceeding state-of-the-art performance.

2. Problem Setup

PEFT and LoRA. Parameter-efficient fine-tuning via low-rank adaptation (LoRA) is a widely adopted approach

[1]Department of Computer Science, ETH Zurich, Zurich, Switzerland. Correspondence to: Tanguy Dieudonné <tdieudonne@ethz.ch>.

Presented at the ICML 2026 Workshop "Continual Adaptation at Scale: Towards Sustainable AI". Copyright 2026 by the author(s).

for fine-tuning pretrained models to downstream tasks. Specifically, given a pretrained model with weight matrix $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, LoRA (Hu et al., 2021) introduces the parameter update $\Delta\mathbf{W} = \mathbf{AB} \in \mathbb{R}^{m\times n}$ where $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times n}$ are two low-rank learnable matrices with rank $r \ll \min\{m, n\}$. The LoRA-updated output becomes $\boldsymbol{h} = (\mathbf{W}_0 + \mathbf{AB})\boldsymbol{x}$.

Continual Learning. In CL, a sequence of tasks $\{\mathcal{T}_1, \cdots, \mathcal{T}_T\}$ must be learned sequentially. The t-th task $\mathcal{T}_t$ consists of a training dataset $\mathcal{D}_t = \{x_t^{(i)}, y_t^{(i)}\}_{i=1}^{|\mathcal{D}_t|}$ where $x_t^{(i)}$ denotes an input image and $y_t^{(i)}$ its corresponding label. When training on $\mathcal{D}_t$, the model does not have access to previous data $\{\mathcal{D}_k\}_{k=1}^{t-1}$ but is required to mitigate catastrophic forgetting of the $t-1$ previously learned tasks. We follow the class-incremental learning setting where the model must classify all classes seen across all tasks, without being provided the task identity at inference. When applying LoRA to CL, during learning on task $\mathcal{T}_t$, we compute the output at a given layer as

$$\boldsymbol{h} = (\mathbf{W}_0 + \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 + \cdots + \mathbf{A}_t\mathbf{B}_t)\boldsymbol{x}.$$

3. LITELORA

We propose a modular gating framework that can wrap any adapter-based CL method.

3.1. Learnable Gating Mechanism

Given a sequence of tasks, each task i is associated with a candidate adapter $\Delta\mathbf{W}_i \in \mathbb{R}^{m\times n}$. The cumulative weight update after learning task t is:

$$\mathbf{U}_t = \sum_{i=1}^{t} \beta_i \cdot \Delta\mathbf{W}_i, \tag{1}$$

where $\Delta\mathbf{W}_i$ is the adapter for task i and $\beta_i \in \{0, 1\}$ is a binary gate determining whether it contributes to the forward pass. For previously learned tasks ($i < t$), β_i is fixed after training. For the current task, we initialize $\beta_t \equiv 1$ to learn candidate features, and subsequently optimize it to determine whether the newly introduced adapter is necessary.

We employ a *global gating strategy*, where a single scalar logit l_i controls the activation of the adapter for task i across all layers. This formulation encourages the model to make a binary decision at the task level: either the new task requires a dedicated adapter, or it can be solved by reusing existing knowledge.

Differentiable Selection via Gumbel-Sigmoid. To make the binary decision differentiable, we parameterize β using a continuous logit l. We employ the Gumbel-Sigmoid relaxation (Jang et al., 2016) combined with the Straight-Through Estimator (STE) (Bengio et al., 2013). During the forward pass, we compute a "soft" gate β_{soft} and discretize it:

$$\beta_{\text{soft}} = \sigma\left(\frac{l + G}{\tau}\right), \quad \beta = \mathbb{I}(\beta_{\text{soft}} > 0.5), \tag{2}$$

where σ is the sigmoid function, τ is a temperature parameter (fixed to $\tau = 1$), and $G \sim \text{Gumbel}(0, 1)$ is noise sampled during training. To enable gradient flow, we use the STE formulation:

$$\beta_{\text{train}} = \beta - \text{SG}(\beta_{\text{soft}}) + \beta_{\text{soft}}, \tag{3}$$

where $\text{SG}(\cdot)$ denotes the stop-gradient operator. This allows the network to use discrete binary weights in the forward pass while gradients propagate through the continuous β_{soft} in the backward pass. At *training time* we add *Gumbel noise* G to the new logit, as we found that this would lead to increased levels of sparsity. Intuitively, the added noise may randomly turn on and off the adapter during training, acting like a dropout regularizer. Importantly, *the Gumbel noise is sampled only during training*. At inference time, we disable the noise and compute a deterministic selection using the learned logit $\beta = \mathbb{I}(\sigma(l_i) > 0.5)$. This ensures reproducible predictions while the training process benefits from stochastic exploration. We find that Gumbel noise acts as a dropout-like regularizer; see Appendix A.3 for an ablation, but remark that it is not essential to the method. By default, noise is injected only in Phase 2; Phase 1 is deterministic (the "No phase 1 noise (Ours)" row in Table 4).

Two-phase Training Simultaneously optimizing the adapter parameters $(\mathbf{A}, \mathbf{B})$ and the structural gate (l) can lead to instability. Therefore, we decouple the learning process for each task t into two distinct phases:

- **Phase 1: Feature Acquisition.** We freeze all previous adapters and the current gate logit l_t (initializing $\beta_t = 1$). We train the new adapter parameters $\{\mathbf{A}_t, \mathbf{B}_t, \alpha_t\}$ and the classification head. This phase focuses entirely on learning the features necessary to solve the new task, assuming the capacity is available.
- **Phase 2: Structural Pruning.** This phase is extremely lightweight: one epoch over a single scalar versus thousands of adapter parameters in Phase 1. We freeze the newly learned adapter parameters and the classification head, and train *only* the scalar logit l_t under a global sparsity penalty (the full training loss is given in Eq. 4, Appendix A.4). Since all previously learned gating logits $\{l_i\}_{i<t}$ are frozen, the sparsity penalty on their corresponding gates is constant and does not contribute to the gradients. As a result, only the current logit l_t is updated. This phase forces the model to verify if the new adapter is strictly necessary. If the task can be

solved using the accumulated knowledge of previous adapters, the sparsity penalty drives β_t to 0.

At the end of Phase 2, we evaluate the final gate β_t. If $\beta_t = 0$, the parameters $\mathbf{A}_t, \mathbf{B}_t$ are permanently discarded, resulting in zero parameter growth for that task.

4. Empirical analysis

Benchmarks. We evaluate LITELORA on three benchmark datasets for class-incremental learning. *CIFAR-100* (Krizhevsky, 2009) contains 100 classes of 32×32 images, which we split into 10 disjoint tasks of 10 classes each. *ImageNet-A* (Hendrycks et al., 2021b) consists of 200 classes of natural adversarial examples; we split it into 20 disjoint tasks of 10 classes each. *ImageNet-R* (Hendrycks et al., 2021a) consists of 200 ImageNet classes (Deng et al., 2009) rendered in artistic styles; we split it into 20 tasks of 10 classes each.

Implementation. We instantiate LITELORA on top of SD-LoRA (Wu et al., 2025), using its magnitude–direction decomposition $\Delta\mathbf{W}_i = \alpha_i \cdot \overline{\mathbf{A}_i\mathbf{B}_i}$ as the per-task adapter, chosen for its strong plasticity properties, where $\overline{\mathbf{AB}} = \frac{\mathbf{AB}}{\|\mathbf{AB}\|_F}$. The gating mechanism is otherwise backbone-agnostic and could equally wrap O-LoRA or InfLoRA adapters. We adopt the setup of SD-LoRA (Wu et al., 2025), which has been shown to outperform prior LoRA-based CL methods such as InfLoRA (Liang & Li, 2024). For all three datasets we use a pre-trained ViT-B/16 backbone (Dosovitskiy et al., 2021) with a hidden dimension $d = 768$. LoRA adapters of rank $r = 10$ are applied to the Query and Value projections in all 12 Transformer blocks.

Metrics. We evaluate performance using two standard metrics: **Average Anytime Accuracy (A)**, defined as the mean accuracy across all seen tasks evaluated after each incremental step; and **Forgetting (F)**, which measures the average decline in performance for each task relative to its peak value.

4.1. LITELORA matches accuracy with far fewer adapters

As detailed in Table 1, our method consistently matches or exceeds the accuracy of SD-LoRA while substantially reducing parameter growth. In particular, our method has two important characteristics:

- **Efficiency:** On CIFAR-100, we achieve comparable accuracy using only 5–8 adapters, compared to the fixed 10 used by SD-LoRA. This efficiency gain is even more pronounced on the two 20-task ImageNet benchmarks: on ImageNet-A (Order 1), our method requires only 6 adapters versus 20 for the baseline, and on ImageNet-R it uses just 6–7 adapters across all three orderings, corresponding to a 65–70% reduction in added parameters.
- **Robustness:** Despite the variation in active adapter count across orderings and datasets, accuracy remains stable. This demonstrates that the gating mechanism successfully identifies the minimal capacity required for each curriculum, maintaining a strong accuracy–efficiency trade-off even under both adversarial-style shift (ImageNet-A) and rendition/style shift (ImageNet-R).

4.2. The ordering matters for efficiency

Whether a new adapter is recruited depends on the semantic gap between the new task and those already learned: if existing representations suffice, phase 2 discards the new adapter. The order in which tasks are visited therefore directly controls total adapter count. Across three orderings on all three benchmarks (Table 1), the number of active adapters varies from 5 to 8 on CIFAR-100, from 6 to 14 on ImageNet-A, and from 6 to 7 on ImageNet-R, while accuracy remains stable throughout, confirming that the additional adapters are redundant rather than necessary. On ImageNet-A Order 1, retaining just 6 adapters improves average accuracy by 0.9 points over SD-LoRA, suggesting this ordering presents tasks in a sequence where prior representations transfer well. ImageNet-R exhibits an even more stable pruning profile across orderings, which is consistent with the idea that rendition shifts preserve reusable structure that can be captured by a small shared set of low-rank adapters. In all cases, the gating mechanism adapts to the curriculum.

4.3. Low-Rank Redundancy is Structured

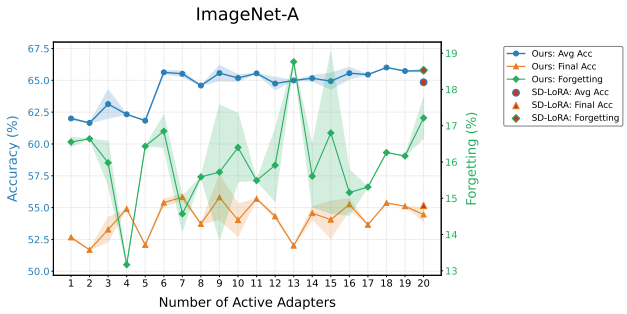

Figure 1. Sparsity–accuracy frontier on ImageNet-A (order 1). We report average accuracy and forgetting as a function of the number of active adapters.

We now ask not just how many adapters are needed, but whether the redundancy itself has stable structure. To answer this, we sweep the sparsity regularization strength $\lambda_{\text{sparsity}}$ across a broad range, yielding configurations with

Table 1. **Main Results.** Comparison of SD-LoRA and our method (LITELORA) on CIFAR-100 ($N = 10$), ImageNet-A ($N = 20$), and ImageNet-R ($N = 20$) across three different task orderings. We report Average Accuracy (A↑), Forgetting (F↓), and the number of active adapters (#Ad↓).

	Order 1			**Order 2**			**Order 3**		
Method	A↑	F↓	#Ad↓	A↑	F↓	#Ad↓	A↑	F↓	#Ad↓
CIFAR-100 ($N = 10$)									
SD-LoRA	91.71	5.92	10	92.54	**4.93**	10	91.27	**5.96**	10
Ours	**91.81**	**4.66**	**5**	**92.64**	5.14	**8**	**91.41**	6.09	**7**
ImageNet-A ($N = 20$)									
SD-LoRA	64.85	18.53	20	66.30	**14.70**	20	62.08	**15.14**	20
Ours	**65.75**	**16.55**	**6**	**66.53**	17.27	**14**	**63.64**	16.49	**13**
ImageNet-R ($N = 20$)									
SD-LoRA	82.41	10.25	20	81.55	8.89	20	81.27	13.67	20
Ours	**82.83**	**7.84**	**6**	**81.64**	**8.49**	**7**	**81.71**	**12.26**	**6**

active adapter counts varying from 1 to the full sequence length N.

Figure 1 illustrates the resulting sparsity–accuracy frontier on ImageNet-A; the CIFAR-100 frontier is given in Appendix A.2. We observe that average accuracy saturates quickly, often peaking at intermediate values rather than at the full budget. For instance, on CIFAR-100, configurations with 5–7 active adapters outperform the SD-LoRA baseline by 0.1% and our own dense (10-adapter) model by 0.3%. Similarly, on ImageNet-A, the 6-adapter configuration surpasses SD-LoRA by 0.9% and remains within 0.13% of our 20-adapter model, despite a 70% reduction in parameters. These results establish that the low-rank subspace needed to represent a stream of N tasks is spanned by far fewer than N adapters.

This trend confirms our driving hypothesis: excessive adapter capacity increases the risk of forgetting without improving plasticity. As sparsity increases, forgetting generally decreases while accuracy remains stable. However, extreme sparsity (e.g., 1–2 adapters) eventually degrades performance, highlighting the necessity of our data-driven selection rule over a fixed, minimal budget.

Further evidence for this comes from sweeping $\lambda_{\text{sparsity}}$: on CIFAR-100 (Table 2), the pruned set grows monotonically with λ, revealing a stable importance hierarchy. On ImageNet-A (Appendix A.2) the ordering is noisier, but Tasks 5 and 8 are pruned in 88% of configurations, confirming that the gating mechanism captures meaningful redundancy structure even across 20 fine-grained tasks. Together, these patterns **validate** the hypothesis that task-level gating learns a meaningful redundancy structure, promoting the selective reuse of previously learned low-rank components.

Table 2. **CIFAR-100 Pruning Order.** Evolution of the pruned task set as the sparsity penalty λ increases. The strictly nested sets (each row is a superset of the previous) indicate a stable hierarchy of task importance. Minor deviations occur at λ=0.04 and λ=0.05 due to the stochastic nature of the Gumbel gate. Task 6 is consistently the first to be removed, suggesting its knowledge is already well-captured by earlier adapters.

$\lambda_{\text{sparsity}}$	Active	Pruned Tasks (Cumulative)
≤ 0.01	10	$\emptyset$
0.015	9	$\{6\}$
0.02	8	$\{4, 6\}$
0.027	7	$\{2, 4, 6\}$
0.028	6	$\{2, 4, 6, 9\}$
0.04	5	$\{2, 4, 6, 7, 8\}$
0.05	4	$\{2, 3, 4, 6, 8, 9\}$
0.054	2	$\{2, 3, 4, 5, 6, 7, 8, 9\}$
0.1	1	$\{1, \ldots, 9\}$

5. Conclusion

In this work, we presented **LITELORA**, a parameter-efficient continual learning method that dynamically adjusts model capacity to the complexity of the task stream. By revisiting the rigid "one-adapter-per-task" paradigm, we demonstrated that high performance does not require linear parameter growth. Instead, we introduced a differentiable gating mechanism that effectively acts as a structural switch, allowing the model to learn new parameters only when the semantic novelty of a task demands it.

Our empirical analysis on CIFAR-100, ImageNet-A, and ImageNet-R yields two key insights. First, the stability–plasticity dilemma can be mitigated by decoupling feature learning from structural allocation; our method preserves the plasticity of unconstrained LoRA while achieving stability through selective reuse. Second, standard CL benchmarks

contain significant redundancy; we showed that up to 70% of task-specific adapters can be pruned with no loss in accuracy and comparable forgetting.

We hope to motivate further theoretical study of when and why low-rank subspaces are shared across tasks.

Impact Statement

Our work improves the efficiency of continual adaptation for foundation models by reducing unnecessary parameter growth. This can lower the computational and energy costs of maintaining large models over time, contributing to more sustainable machine learning practices.

As with other advances in model efficiency, reduced costs may enable wider deployment, which underscores the importance of responsible use and appropriate safeguards in downstream applications.

References

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2020.09.004. URL https://www.sciencedirect.com/science/article/pii/S1364661320302199.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. *CVPR*, 2021b.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images, 2009.

Liang, Y.-S. and Li, W.-J. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23638–23647, June 2024.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Wang, X., Chen, T., Ge, Q., Xia, H., Bao, R., Zheng, R., Zhang, Q., Gui, T., and Huang, X. Orthogonal subspace learning for language model continual learning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=L7ZBpZZ8Va.

Wu, Y., Piao, H., Huang, L.-K., Wang, R., Li, W., Pfister, H., Meng, D., Ma, K., and Wei, Y. SD-loRA: Scalable decoupled low-rank adaptation for class incremental learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5U1rlpX68A.

A. Appendix

A.1. Limitations

First, the final keep/discard decision depends on hyperparameters such as the sparsity weight, gating temperature, and the duration of Phase 2 optimization. These choices may require tuning across backbones or datasets.

Second, while our method enables task-level pruning, it implicitly assumes that redundant adapters are not uniquely required for future tasks. In other words, when a task-specific adapter is removed, we assume its contribution is already subsumed by earlier learned adapters. Although our empirical results suggest a strong redundancy structure and a stable pruning hierarchy (Appendix A.2), this assumption may not hold in settings with highly compositional tasks, where later tasks could depend on features that are not fully captured by earlier adapters.

Finally, our selection phase introduces additional optimization steps per task (albeit lightweight), and we have not yet studied the strongest scaling regimes (e.g., longer streams or higher-resolution backbones).

A.2. Pruning Consistency Analysis

We report the full pruning frequency analysis for ImageNet-A, where individual decisions are noisier due to the larger number of fine-grained tasks.

Nevertheless, a *soft* importance hierarchy emerges. Table 3 reports the fraction of runs (across 43 λ values in the active pruning range) in which each task is pruned. Tasks 5 and 8 are pruned in 88% of configurations, and a clear gradient of importance exists in between. This confirms that the gating mechanism captures meaningful task-level redundancy even when individual pruning decisions are noisy.

Table 3. **ImageNet-A Pruning Frequency.** The percentage of runs (out of 43 configurations with $\lambda \in [0.01, 0.04]$) in which specific tasks were pruned.

Tasks	Prune Frequency
0	0%
2, 3	16–21%
9, 11, 18	33–40%
1, 4, 12, 13, 15	42–49%
10, 14, 16, 17, 19	54–66%
6, 7	65–72%
5, 8	88%

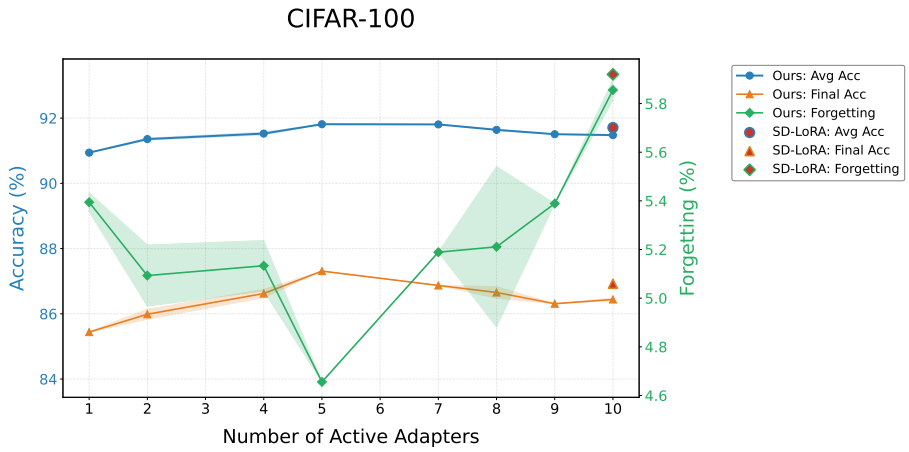

Figure 2. Sparsity–accuracy frontier on CIFAR-100 (order 1). We report average accuracy and forgetting as a function of the number of active adapters. Note that varying the regularization strength parameter $\lambda_{\text{sparsity}}$ implicitly changes the number of adapters, therefore it was not possible to test all points on the x axis.

A.3. Ablation Studies

We study the impact of injecting stochastic Gumbel noise during Phase 1; by default we keep Phase 1 deterministic and add noise only in Phase 2. As shown in Table 4, adding noise in both phases does not improve and generally lowers average accuracy across all sparsity regimes, confirming that Phase 1 noise is unnecessary.

Table 4. Gumbel-noise ablation on ImageNet-A (20 tasks) across different sparsity regimes. Results reported for Order 1.

Regime	Model variant	Avg. Acc. (%)↑	Last Acc. (%)↑	Forgetting (%)↓	#Ad↓
Dense	Noise in both phases	64.93	53.46	17.56	20
	No phase 1 noise (Ours)	**65.80**	54.11	17.22	20
	No noise	65.74	**54.84**	**16.25**	20
Balanced	Noise in both phases	62.01	50.89	18.37	11
	No phase 1 noise (Ours)	62.19	51.22	**15.89**	10
	No noise	**63.01**	**52.73**	16.71	**8**
Sparse	Noise in both phases	61.69	**54.64**	**13.54**	3
	No phase 1 noise (Ours)	**62.20**	53.13	15.53	3
	No noise	62.03	53.72	15.09	3

A.4. Experiment Details

We provide the detailed hyperparameters used for our experiments on CIFAR-100, ImageNet-A, and ImageNet-R in Table 5. The sparsity penalty $\lambda_{\text{sparsity}}$ was selected via grid search on the first seed/ordering.

Table 5. Hyperparameter settings for LITELORA.

Parameter	**CIFAR-100**	**ImageNet-A**	**ImageNet-R**
Backbone	ViT-B/16 (IN-21k)	ViT-B/16 (IN-21k)	ViT-B/16 (IN-21k)
LoRA Rank (r)	10	10	10
# Tasks	10	20	20
Phase 1: Representation Learning			
Optimizer	SGD	Adam	Adam
LR Scheduler	Cosine	Constant	Constant
Learning Rate	8×10^{-3}	1×10^{-2}	1×10^{-2}
Weight Decay	2×10^{-4}	2×10^{-4}	2×10^{-4}
Epochs	20	20	20
Batch Size	128	128	128
Init α	1.0	1.0	1.0
Init Logit (l_0)	0.5	0.5	0.5
Phase 2: Selection Learning			
Optimizer	AdamW (Loshchilov & Hutter, 2019)	AdamW	AdamW
Learning Rate	0.05	0.05	0.05
Epochs	1	1	1
Batch Size	128	16	16
$\lambda_{\text{sparsity}}$ (Range)	[0.015, 0.1]	[0.01, 0.04]	[0.02, 0.05]
Gumbel τ	1.0	1.0	1.0

Combined Training Loss. For task $\mathcal{D}_t$, we optimize the network by combining cross-entropy loss with an ℓ_1 sparsity regularization applied to the gates:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{sparsity}} \sum_{i=1}^{t} \beta_i, \tag{4}$$

where the sum runs over the per-task scalar gates and, since all previous gates are frozen, only β_t receives a gradient.