More Than Memory Savings: Zeroth-Order Optimization Mitigates Forgetting in Continual Learning

Supplementary Material

A. Training, Evaluation Configurations

Training configurations: For our proposed ZO-FC method, we optimize the classifier using FO SGD with a learning rate of 0.01 that follows a cosine decay schedule; the adapter parameters are updated using the SPSA and ZO SGD with a constant learning rate of 0.01. For SPSA, we set the perturbation magnitude to $\varepsilon = 0.001$, and the query budget to $Q = 4$, and apply L2 norm clipping at a threshold of 1.0 to the estimated gradients for stability. The training budgets are set per dataset: (1) On CIFAR100, the FO classifier is trained for 10 epochs, and the ZO adapter is trained for 20 epochs; (2) On ImageNet-R, the FO classifier is trained for 20 epochs, and the ZO adapter is trained for 40 epochs. For all baseline methods, we adopt the official implementations and follow the hyperparameter settings recommended in their original papers and the LAMDA-PILOT benchmark [5], running on 1×A6000, to ensure a fair comparison. Specifically, for the baseline combining an adapter with a cosine classifier, we train for a unified 10 epochs on CIFAR100 and 20 epochs on ImageNet-R. All runs use a batch size of 48 to ensure fair comparison of memory usage during optimization.

Evaluation Metrics: Let K be the number of tasks. Let $A_{i,j}$ denote the test accuracy on task j after completing training on task i (so the final accuracies are $\{A_{K,j}\}_{j=1}^{K}$).

Average accuracy (avg).

$$\text{avg} \;=\; \frac{1}{K}\sum_{j=1}^{K} A_{K,j}.$$

Last-task accuracy (last).

$$\text{last} \;=\; A_{K,K}.$$

Catastrophic forgetting (fgt): For each old task $j < K$, let $A_j^\star$ be the best accuracy that task ever achieved during training:

$$A_j^\star \;=\; \max_{i\in\{j,\ldots,K\}} A_{i,j}.$$

Forgetting is the average drop from this best value to the final value:

$$\text{fgt} \;=\; \frac{1}{K-1}\sum_{j=1}^{K-1}\left(A_j^\star \;-\; A_{K,j}\right).$$

B. Algorithm

As shown in Algorithm 1, ZO-FC updates the classifier ψ with standard FO SGD while updating the PEFT adapter ϕ via a two-point ZO (SPSA) step. For each mini-batch, we first compute one FO loss and back-propagate *only* through the head (Lines 1–3). We then draw Q Rademacher directions $\{\Delta_q\}_{q=1}^{Q}$ and evaluate the loss at $\phi \pm \varepsilon\Delta_q$ (Lines 5–10), yielding the finite-difference estimates $\hat{g}_q = \frac{L^+ - L^-}{2\varepsilon}\,\Delta_q$ (Line 11). The ZO estimates are averaged and clipped before a ZO-SGD update on ϕ (Lines 12–14).

Algorithm 1 The proposed ZO-FC

Require: mini batch $\mathcal{B}$, classifier ψ, adapter ϕ, learning rates η_ψ, η_ϕ; perturbation scale ε, query budget Q

1: **// FO step on classifier**
2: $L \leftarrow L(\phi, \psi; \mathcal{B})$ {back-prop only through ψ}
3: $\psi \leftarrow \psi - \eta_\psi \nabla_\psi L$
4: **// ZO step on adapter (SPSA)**
5: **for** $q = 1$ to Q **do**
6: sample $\Delta_q \sim \{\pm 1\}$ {store Δ_q only}
7: $\phi' \leftarrow \phi + \varepsilon\Delta_q$
8: $L^+ \leftarrow L(\phi', \psi; \mathcal{B})$
9: $L^- \leftarrow L(\phi' - 2\varepsilon\Delta_q, \psi; \mathcal{B})$
10: restore $\phi' \leftarrow \phi$ {no need to store ϕ'}
11: $\hat{g}_q \leftarrow \frac{L^+ - L^-}{2\varepsilon}\Delta_q$
12: **end for**
13: $\bar{g}_\phi \leftarrow \frac{1}{Q}\sum_q \hat{g}_q$
14: $\phi \leftarrow \phi - \eta_\phi\,\text{clip}(\bar{g}_\phi)$

C. Proof Assumptions for ZO SPSA searches flat minima

Under the standard assumptions that L is convex, three-times continuously differentiable, and its third derivatives are bounded (the usual "L_3-smoothness" condition), Zhang et al. [7] proved that in $T = \tilde{\mathcal{O}}(d^4/\varepsilon^2)$ iterations: SPSA

drives the ε-smoothed loss $\tilde{L}_\varepsilon(\theta)$ to within ε of its global minimum:

$$\tilde{L}_\varepsilon(\theta_T) - \min_\theta \tilde{L}_\varepsilon(\theta) \;\le\; \varepsilon, \tag{1}$$

where θ_T is the final parameters after T SPSA steps. By the definition of $\tilde{L}_\varepsilon$, Zhang et al. [7] also showed

$$\mathrm{Tr}\left[\nabla^2 \mathcal{L}(\theta_T)\right] - \min_{\theta^*} \mathrm{Tr}\left[\nabla^2 \mathcal{L}(\theta^*)\right] \;\le\; \varepsilon, \tag{2}$$

that is, among all minimizers of the original loss $L(\theta)$, SPSA selects those with the smallest Hessian trace-flatter minima.

Standard GD optimizes the raw loss f, which does not guarantee bias toward flatter minima. Empirically, SGD sometimes finds wide valleys due to noise, but this is incidental and depends on schedule and noise scale. Explicit methods such as **Entropy-SGD** [1], **Sharpness-Aware Minimization (SAM)** [2] bias GD trajectories into wide valleys by injecting perturbations into the update rule. In contrast, **two-point ZO** *always* optimizes the smoothed loss $\tilde{f}$, and theory shows it minimizes to within ϵ of the smallest-Hessian-trace solution. This explains why our ZO-FC retains low forgetting across various streams: it consistently biases the shared adapter toward flat representations, without extra regularization.

D. Proof for flat minima in Training dynamics of ZO-FC

When combining zeroth order adapter with first order classifier, the flat minima effect is maintained for continual learning. For the adapter ϕ and classifier ψ, a two-point SPSA step on ϕ is an unbiased estimator of the gradient of the *Gaussian-smoothed* loss $\tilde{L}_\varepsilon(\phi,\psi) = \mathbb{E}_u[L(\phi+\varepsilon u,\psi)]$: $\mathbb{E}[g_\phi] = \nabla_\phi \tilde{L}_\varepsilon(\phi,\psi)$. If the classifier ψ is updated on a faster time-scale (FO) while ϕ evolves slowly (ZO), the limiting dynamics follow $\bar{\psi} = -\nabla_\psi L(\phi,\psi)$ (fast) with $\psi^\star(\phi) = \arg\min_\psi L(\phi,\psi)$, and $\bar{\phi} = -\nabla_\phi \tilde{L}_\varepsilon(\phi,\psi^\star(\phi))$ (slow). Hence the hybrid iterates descend the reduced objective $J(\phi) = \min_\psi \tilde{L}_\varepsilon(\phi,\psi)$. This connects ZO-FC to flat minima: smoothing acts directly on the representation ϕ, biasing toward lower curvature for stability, while FO keeping the classifier plastic.

E. Component Analysis: Classifier Variants and FO, ZO swap

We further investigate the design of our approach by comparing the impact of different learnable classifiers (cosine vs. linear) and trainable component optimization strategies (FO vs. ZO). Table 1 and Table 2 present detailed results on the Inc-5 setting for both CIFAR100 and ImageNet-R.

Classifier Choice: As shown in Table 1, the FO optimized adapter performs closely no matter combined with either a cosine or linear classifier though cosine classifier works better than linear classifier when without adapter. FO-optimized adapters achieve strong performance regardless of whether they are paired with a cosine or linear classifier. However, the cosine classifier consistently yields higher accuracy and lower forgetting than its linear counterpart when used alone or within hybrid settings. When combined with linear classifiers, ZO adapter struggle with catastrophic forgetting, suggesting poor alignment between the decision boundary and the adapted feature space under noisy updates. This performance gap indicates that the cosine classifier better captures and reflects the fine-grained adaptations introduced by the ZO-optimized adapter, leading to superior retention and generalization.We also evaluate an EASE-style variant: the temporary FO classifier used during training is discarded and replaced by per-class prototypes after each task as final classifier. Replacing the temporary FO learnable classifier during training with prototypes as final classifier yields performance near SimpleCIL but clearly below ZO-FC and with higher forgetting.

Method	CIFAR100 Inc-5			ImageNet-R Inc-5		
	avg	last	fgt	avg	last	fgt
Cosine Cls.	85.61	78.33	5.17	60.6	52.12	3.98
Linear Cls.	84.78	76.24	9.32	61.19	44.42	17.63
Adapter + Cosine Cls.	88.41	83.64	9.58	74.71	69.62	8.49
Adapter + Linear Cls.	90.19	83.97	7.69	73.08	66.12	9.01
ZO-FC (Cosine)	88.39	83.34	5.26	72.01	66.63	4.40
ZO-FC (Linear)	85.94	79.61	5.92	62.07	51.78	9.89
ZO Adapter + Prototype	84.94	80.87	10.77	67.06	62.98	16.39

Table 1. Ablation study of classifier types under FO optimization of the classifier and ZO updates on the adapter

Effect of ZO Placement: Table 2 isolates the impact of applying ZO optimization to different trainable components of the model. When the adapter is optimized with ZO and the classifier remains FO, that is, ZO-FC, the model achieves strong performance. In contrast, reversing this setup and applying ZO to the classifier while keeping the adapter FO optimized results in remarkable performance degradation. This contrast confirms that the classifier requires precise gradient-based optimization to maintain stable decision boundaries across tasks, while the adapter can tolerate noisy ZO updates without compromising overall performance.

Method	CIFAR100 Inc-5			ImageNet-R Inc-5		
	avg	last	fgt	avg	last	fgt
ZO-FC	88.39	83.34	5.26	72.01	66.63	4.40
FO adapter + ZO Cls.	62.45	53.22	25.65	23.35	11.72	18.07

Table 2. Effect of ZO,FO components placement

These results separate the roles of representation

smoothing (adapter) and decision-boundary plasticity (classifier), and motivate our choice of a single shared adapter.

F. Data Augmentation and Effects

All image augmentations are applied in the dataloader before the model forward. No augmentation is performed inside the forward pass. This pipeline is identical for FO, ZO, and ZO-FC. Stochastic layers in the backbone (e.g., dropout) are disabled during ZO updates. The detailed augmentations are shown below, following configurations in [5] and [4]:

CIFAR100, ImageNet-R. *Train:* RandomResizedCrop(224, scale $\in$(0.05,1.0), ratio $\in$(3/4,4/3)), RandomHorizontalFlip(p=0.5). *Test:* Resize(256), CenterCrop(224).

DomainNet. *Train:* RandomResizedCrop(224), RandomHorizontalFlip, Normalize(μ=(0,0,0),σ=(1,1,1)). *Test:* Resize(256), CenterCrop(224), Normalize(μ=(0,0,0), σ=(1,1,1)).

Method	CIFAR100 Inc-5			ImageNet-R Inc-5		
	avg	last	fgt	avg	last	fgt
ZO-FC w aug	88.39	83.34	5.26	72.01	66.63	4.40
ZO-FC w/o aug	88.63	83.14	8.02	73.04	67.35	5.19

Table 3. Effect of data augmentation on ZO-FC.

Augmentations act as input-space regularization that encourages invariances. In continual learning, that regularization reduces representation drift toward the current task and thus mitigates forgetting. On CIFAR100, removing augmentation (keeping only resize/center-crop) yields similar accuracy, and higher forgetting; on ImageNet-R, the accuracy increases slightly with much higher forgetting. This supports that standard augmentation helps ZO-FC stability rather than hurting ZO quality.

G. Further studies on Prompt-based Methods

Method	CIFAR100 Inc-5			ImageNet-R Inc-5		
	avg	last	fgt	avg	last	fgt
L2P	26.43	4.79	19.05	11.83	2.52	10.21
CODA-Prompt	10.12	2.40	11.53	6.47	1.21	5.83

Table 4. Increased epochs for prompt based method, 100 for Cifar100, 200 for INR.

We conduct further experiments on L2P and CODA-Prompt with more updates (100 epochs on CIFAR100, 200 on ImageNet-R). Despite the longer schedule, the two prompt based methods fail to converge under ZO optimization and their accuracy results remain far below adapter-based ZO training and ZO-FC. This matches recent studies [6] that prompt-based methods are harder to converge under ZO SPSA because optimizing prompt selection with penalties and contrastive alignment loss requires precise gradients.

H. Memory and Computation Cost Analysis

We evaluate the memory efficiency and computational overhead of our ZO-FC method relative to both FO and ZO optimized CIL methods. This is critical for continual learning in resource-constrained environments, such as edge devices.

Memory–Accuracy Trade-off: As shown in Table 5, FO optimization methods that incorporate PEFT modules incur significant memory costs—up to 7.21 GB for CODA-Prompt and over 4 GB for most adapter-based baselines. In contrast, ZO optimization yields dramatic memory savings, with most ZO variants requiring < 0.7 GB, regardless of the adapter type. Notably, our proposed ZO-FC maintains only 0.70 GB of peak GPU memory usage—on par with minimal FO models using fixed classifiers. This demonstrates that our hybrid strategy preserves the memory efficiency of ZO methods, while achieving high performance, as shown in Figure 1, 2.

Scenario	Peak GPU Memory (GB)
FO Optimization	
Learnable Cls.	0.70
Adapter + Learnable Cls.	4.35
LAE	4.48
L2P	5.58
CODA-Prompt	7.21
APER	4.25
EASE	4.38
InfLoRA	4.98
ZO Optimization	
Learnable Cls.	0.69
Adapter + Learnable Cls.	0.69
LAE	0.69
APER	0.66
EASE	0.87
InfLoRA	0.88
ZO-FC	0.70

Table 5. Per epoch Optimization Memory cost with batch size 48

Runtime–Memory Trade-off: Table 6 shows the per-epoch training time and corresponding optimization memory cost. ZO methods achieve extreme memory efficiency, but incur increasing compute overhead with higher query budgets, reflecting the cost of additional function queries. Due to optimizer decomposition for different trainable modules, our approach costs slightly longer runtime than ZO. Despite this trade-off, Our approach's runtime remains

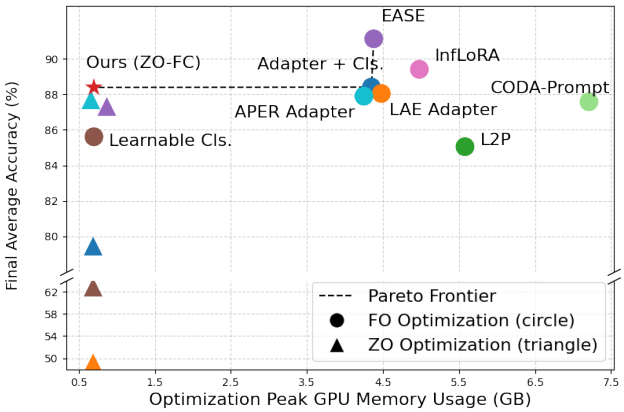

Figure 1. Memory–accuracy trade-off across CIL methods on CIFAR100. Note that, same color denotes the same method.

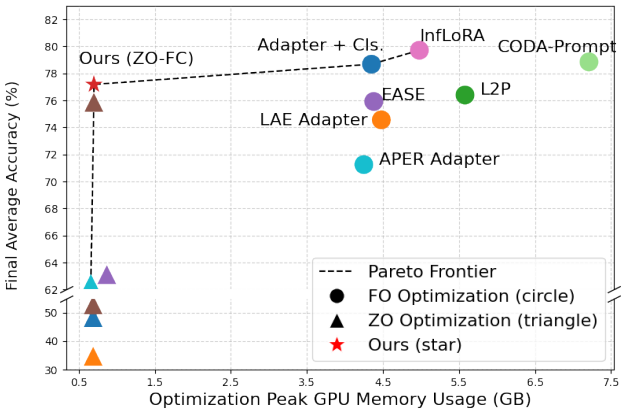

Figure 2. Memory–accuracy trade-off across CIL methods on DomainNet. Note that, same color denotes the same method.

Scenario	Peak GPU Memory Cost	Running time
FO-SGD	4.35 GB	24.54s
ZO-SGD (q=1)	0.69 GB	19.44s
ZO-SGD (q=4)	0.69 GB	52.31s
ZO-FC (q=1)	0.70 GB	22.15s
ZO-FC (q=4)	0.70 GB	59.36s

Table 6. Per epoch Memory cost & Time Cost with batch size 48

manageable to low-resource hardware.

Computation Cost Analysis: We use FLOPs to measure computation cost. Let **B** be batch size; **H**×**W** the input resolution (224×224); **N** the ViT token count ($N = 1 + (H/16)(W/16) = 197$ at 224^2); **D** the hidden width (ViT-B: D=768); **C** the current #classes; $\mathbf{L_a}$ the #blocks with adapters; and **r** the adapter bottleneck (we use r=5).

Denote the ViT backbone forward cost per image as $F_{\text{backbone}}(H, W)$. A bottleneck adapter applies two linear maps per token ($D \to r$ and $r \to D$), so the adapter overhead per image is

$$F_{\text{PET}} \approx L_a\, N\,(2Dr + 2rD) = L_a\, N\,(4Dr). \tag{3}$$

For the *cosine* classifier, each forward computes: feature ℓ_2–norm ($\sim 3D$ ops), *weight* ℓ_2–norm across C classes ($\sim 3CD$ ops), class dot-products ($\sim 2CD$), and a scalar scale σ ($\sim C$):

$$F_{\text{cls}} \approx (5CD) + 3D + C. \tag{4}$$

Note. In our implementation, feature and weight normalizations are recomputed *every forward*, so the $3CD$ term in (4) is paid per forward. If one caches unit-norm weights (or renormalizes once per optimizer step), then the per-forward cost becomes $(2CD) + 3D + C$ with an additional $3CD$ once per step.

The total *per-image forward* FLOPs is

$$F_{\text{fwd,img}} \approx F_{\text{backbone}}(H, W) + L_a N(4Dr) + \big(5CD{+}3D{+}C\big). \tag{5}$$

For per-batch multiplicities (FO / ZO / hybrid), We report FLOPs using the standard convention that a backward pass over the trainable subgraph costs ≈2× the forward pass, so one training step is ≈3× the forward FLOPs. So let α_{FC} be the FC-classifier backward cost in forward-equivalents of the *head* subgraph (we use $\alpha_{\text{FC}} \approx 2$), and let q be the ZO (SPSA) query budget:

- **FO:** forwards/batch = 1; backward over *trainable parts* (adapters+head). Cost: $B\,F_{\text{fwd,img}} + \alpha_{\text{FC}}\,B\,F_{\text{cls}}$ plus the adapter backward (captured by the 3× rule below).
- **ZO on adapters:** forwards/batch = $2q{+}1$; no backward. Cost: $(2q{+}1)\,B\,F_{\text{fwd,img}}$.
- **ZO-FC (early epochs):** $2q$ ZO forwards on adapters + one FO forward for the classifier + one eval forward; and a classifier-only backward. Cost: $(2q{+}2)\,B\,F_{\text{fwd,img}} + \alpha_{\text{FC}}\,B\,F_{\text{cls}}$.
- **ZO-FC (late; classifier frozen):** same forwards as ZO: $2q{+}1$. Cost: $(2q{+}1)\,B\,F_{\text{fwd,img}}$.

Using (5), the relative *per-batch* overhead vs. FO is

$$\Delta F_{\text{ZO vs FO}} \approx 2q \cdot B \cdot F_{\text{fwd,img}}, \tag{6}$$

$$\Delta F_{\text{ZO-FC(early) vs FO}} \approx (2q{+}1) \cdot B \cdot F_{\text{fwd,img}}, \tag{7}$$

$$\Delta F_{\text{ZO-FC(late) vs FO}} \approx 2q \cdot B \cdot F_{\text{fwd,img}}. \tag{8}$$

Remarks. (i) In FO we backprop through the *trainable* parts (adapters+classifier); the ViT backbone is frozen (no grads) but still incurs one forward per batch. (ii) For ViT-B/16 at 224^2, N=197 and D=768; the adapter overhead scales linearly with r and is small for L_a=5, r=5.

Computation Cost Measurement: We use ViT-B/16 at 224^2 with batch size B=48. The *measured* forward FLOPs per image (fvcore) is $\mathbf{1.768 \times 10^{10}}$, so per-batch forward FLOPs is $\mathbf{8.484 \times 10^{11}}$. FO total is *estimated* via the common rule $\text{FLOPs}_{\text{FO,total}} \approx 3 \times \text{FLOPs}_{\text{forward}}$, yielding $\mathbf{2.545 \times 10^{12}}$ per batch. ZO performs $(2q{+}1)$ forward

Table 7. Per-batch FLOP *breakdown* for FO, ZO, and ZO-FC. *Forward FLOPs / batch* are **measured** once with `fvcore` and scaled by the number of forwards in each regime (baseline single-forward cost: 8.484×10^{11}). *Backward FLOPs / batch* are **estimated**: FO via the $3\times$ rule; ZO-FC via the analytical head-only term $2B\,F_{\text{cls}}$ with $B{=}48,\ D{=}768,\ C{=}5$ at the measured epoch.

Regime	q	Forwards / batch	Forward FLOPs (meas.)	Backward FLOPs (est.)	Total FLOPs (est.)	×FO
FO	–	1	0.848×10^{12}	1.697×10^{12}	$\mathbf{2.545 \times 10^{12}}$	**1.00×**
ZO	1	$2q{+}1{=}3$	$\mathbf{2.545 \times 10^{12}}$	0	$\mathbf{2.545 \times 10^{12}}$	**1.00×**
ZO	4	$2q{+}1{=}9$	$\mathbf{7.636 \times 10^{12}}$	0	$\mathbf{7.636 \times 10^{12}}$	**3.00×**
ZO-FC (early)	1	$\approx 2q{+}1$	$\mathbf{2.545 \times 10^{12}}$	2.06×10^{6}	$\mathbf{2.545 \times 10^{12}}$	**1.00×**
ZO-FC (late)	1	$2q{+}1{=}3$	$\mathbf{2.545 \times 10^{12}}$	0	$\mathbf{2.545 \times 10^{12}}$	**1.00×**
ZO-FC (early)	4	$\approx 2q{+}1$	$\mathbf{7.636 \times 10^{12}}$	2.06×10^{6}	$\mathbf{7.636 \times 10^{12}}$	**3.00×**
ZO-FC (late)	4	$2q{+}1{=}9$	$\mathbf{7.636 \times 10^{12}}$	0	$\mathbf{7.636 \times 10^{12}}$	**3.00×**

Notes: (i) FO backward is estimated via the rule-of-thumb $\text{FLOPs}_{\text{total}} \approx 3 \times \text{FLOPs}_{\text{forward}}$ when training adapters+classifier with a frozen backbone. (ii) ZO-FC early includes a classifier-only FO step; its backward is $2B\,F_{\text{cls}}$ and scales linearly with C. (iii) Because this term is tiny relative to ViT forwards, totals for ZO and ZO-FC practically coincide at the same q.

evaluations with no backward; at $q{=}1$ this matches FO's total, and at $q{=}4$ it is $9\times$ the forward-only batch cost, i.e., $\mathbf{7.636 \times 10^{12}}$. ZO-FC uses the same ZO forwards and adds a classifier-only backward during early epochs. Given our small C and a frozen backbone, this extra term ($\alpha_{\text{FC}}\,B\,F_{\text{cls}}$) is orders of magnitude smaller than a ViT forward, so totals numerically coincide with ZO at the same q (both early and late). This explicitly reports and quantifies the *additional forward passes* required by ZO and how training cost scales with q. In practice, parallelizing the $2q$ ZO forwards [3] can reduce wall-clock time without changing FLOPs and is compatible with ZO-FC for on-device continual learning.

I. CIL Accuracy Trends: ZO-FC vs. FO Methods

To have a good view of the effectiveness of our proposed ZO-FC method, we demonstrate the last accuracy trends across the full task stream on both CIFAR100 and ImageNet-R, under two common incremental learning settings: Inc-10 and Inc-5. Figure 3, 4, 5, 6 illustrate the evolution of performance as more tasks are introduced. These trends highlight the long-term stability of ZO-FC under class-incremental learning.

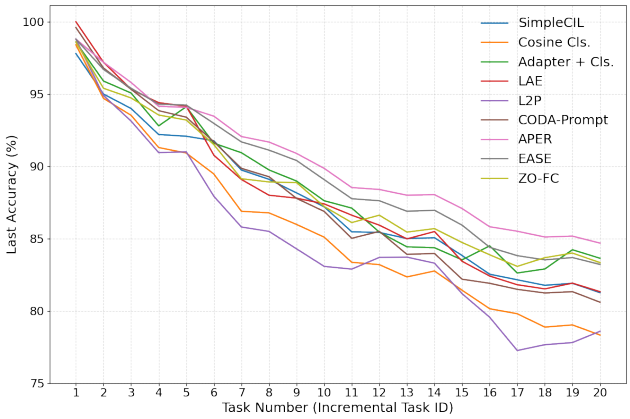

Figure 3. CIFAR100, Inc-5: last accuracy over tasks.

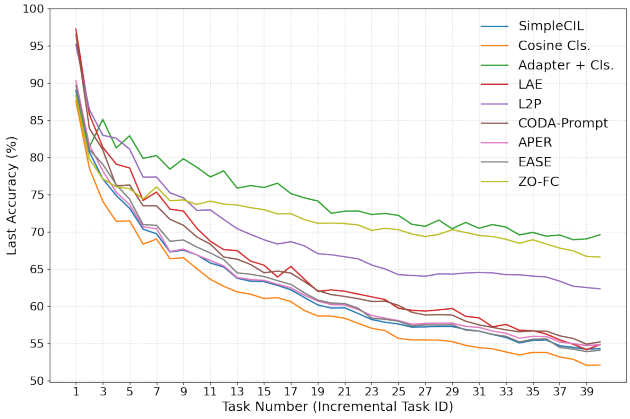

Figure 4. ImageNet-R, Inc-5: last accuracy over tasks.

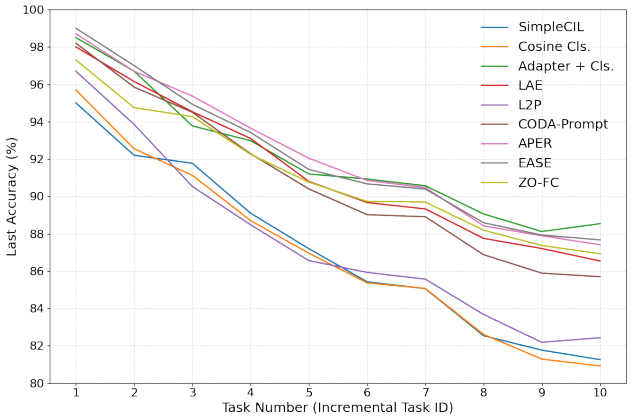

Figure 5. CIFAR100, Inc-10: last accuracy over tasks.

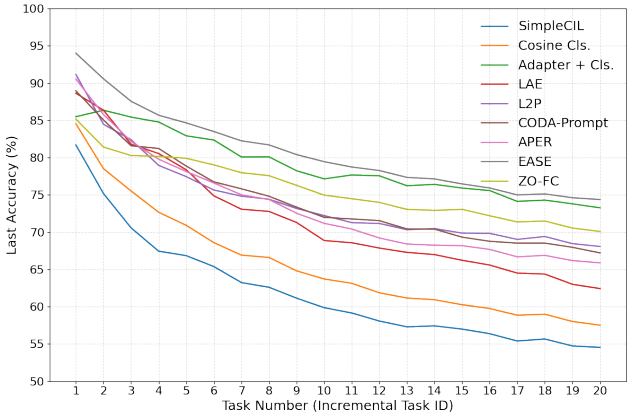

Figure 6. ImageNet-R, Inc-10: last accuracy over tasks.

References

[1] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019. 2

[2] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. 2

[3] Lei Gao, Amir Ziashahabi, Yue Niu, Salman Avestimehr, and Murali Annavaram. Enabling resource-efficient on-device fine-tuning of llms using only inference engines. *arXiv preprint arXiv:2409.15520*, 2024. 5

[4] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024. 3

[5] Hai-Long Sun, Da-Wei Zhou, De-Chuan Zhan, and Han-Jia Ye. Pilot: A pre-trained model-based continual learning toolbox. *SCIENCE CHINA Information Sciences*, 68(4):147101, 2025. 1, 3

[6] Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. AdaZeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 977–995, Miami, Florida, USA, 2024. Association for Computational Linguistics. 3

[7] Liang Zhang, Bingcong Li, Kiran Koshy Thekumparampil, Sewoong Oh, Michael Muehlebach, and Niao He. Zeroth-order optimization finds flat minima, 2025. 1, 2