Dynamic BB-OPLoRA: Rigid Core, Flexible Border for Subspace-Aware Low-Rank Adaptation

Ali Sharifian [1] **Julian Herold** [1] **Tara Gheshlaghi** [2] **Alexander Schug** [1]

Abstract

Low-Rank Adaptation (LoRA) enables efficient fine-tuning of large language models through compact additive updates, but these updates can still interfere with pretrained directions that support broad model capabilities, causing catastrophic forgetting. The recent Orthogonal-Projection LoRA (OPLoRA) method protects dominant pretrained directions, but its rigid preservation rule can limit adaptation and hinder optimization near the spectral boundary. To address this, we introduce Dynamic Budgeted-Border Orthogonal-Projection LoRA (Dynamic BB-OPLoRA), a subspace-aware LoRA that replaces a strict preservation rule with a rigid-core/flexible-border decomposition of the top singular subspace of each pretrained weight matrix. The rigid core preserves the most dominant pretrained directions, while the flexible border allows controlled task-specific adaptation near the spectral boundary without destabilizing the core. The border update is governed by a stiffness-weighted budget that is dynamically adjusted using a gradient-derived pressure signal. We evaluate the method on commonsense reasoning, mathematical reasoning, and Python code generation. Experiments show that Dynamic BB-OPLoRA improves the stability–plasticity trade-off over LoRA and OPLoRA, achieving stronger adaptation while maintaining resistance to cross-domain forgetting.

1. Introduction

Large language models (LLMs) achieve strong performance across a wide range of tasks (Brown et al., 2020; Touvron et al., 2023), but adapting them to downstream domains remains costly. Full fine-tuning updates all model parameters, requiring substantial memory, compute, and storage for each adapted model. This has motivated parameter-efficient fine-tuning methods, which keep most pretrained weights fixed and learn only a small number of task-specific parameters (Houlsby et al., 2019; Lester et al., 2021; Li & Liang, 2021).

Among these methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is widely used for its simplicity and efficiency. LoRA freezes pretrained weights and learns a low-rank additive update that can be merged into the base model after training, avoiding inference-time overhead while using far fewer trainable parameters than full fine-tuning. However, LoRA updates are not explicitly constrained to preserve pretrained knowledge. Even low-rank changes can interfere with directions important for general capabilities, so task adaptation may improve target performance while causing catastrophic forgetting (Biderman et al., 2024; Dou et al., 2024).

A principled strategy for mitigating such interference is to constrain LoRA updates with respect to pretrained subspaces. MiLoRA (Wang et al., 2025) initializes adaptation in the minor singular subspace, LoRA-Null (Tang et al., 2025) projects updates onto the null space of representative pretrained activations, and OPLoRA (Xiong & Xie, 2026) applies double-sided orthogonal projectors to restrict LoRA updates entirely to the orthogonal complements of the pretrained weight matrix's top-K singular subspaces.

This gives OPLoRA an exact preservation guarantee for the protected top-K singular triples during adaptation. While exact preservation effectively mitigates forgetting, freezing the entire top-K subspace can be overly rigid. Because singular values often decay gradually near the spectral cutoff, the transition between preserved and adaptable subspaces is naturally a continuum rather than a discrete boundary. Consequently, marginal protected directions may be closer in importance to nearby unprotected directions than to the leading modes. The cutoff can also be sensitive to multiplicities or near-ties, and directions outside the initial top-K subspace may gain energy during adaptation. This rigidity may also hinder optimization by preventing small but useful updates along boundary directions. Thus, treating all protected directions equally may remove useful adaptation

[1]Scientific Computing Center, Karlsruhe Institute of Technology, Karlsruhe, Germany [2]Department of Computer Science, Karlsruhe Institute of Technology, Karlsruhe, Germany. Correspondence to: Alexander Schug <alexander.schug@kit.edu>.

Preprint. June 8, 2026.

capacity precisely near the boundary where the distinction between preserved and adaptable subspaces is least stable.

Motivated by these observations, we propose **Dynamic Budgeted-Border Orthogonal-Projection LoRA (Dynamic BB-OPLoRA)**, a subspace-aware low-rank adaptation method that preserves the most dominant pretrained directions while allowing controlled adaptation near the spectral boundary. Given a fixed top-K singular subspace of W_0, Dynamic BB-OPLoRA partitions it into a rigid *core* of size $K-N$ and a flexible *border* of size N. The core remains invariant, while the border admits a budgeted Border$\times$Border update induced directly from the existing LoRA factors, adding no trainable parameters. Border flexibility is regulated by a stiffness-weighted budget and adjusted during training using a gradient-derived pressure signal. Our main contributions are as follows.

(i) We introduce **Dynamic BB-OPLoRA**, a core–border decomposition of the top-K singular subspace that exactly preserves a rigid core while allowing budgeted adaptation in a flexible border.

(ii) We propose a parameter-free border construction governed by a novel stiffness-weighted budget. This budget dynamically allocates flexibility using a gradient-derived pressure signal, using existing LoRA factors.

(iii) We prove exact preservation of the core singular triples of W_0 and show improved adaptation quality and forgetting resistance across commonsense reasoning, mathematical reasoning, and Python code generation on the Qwen2.5-7B (Qwen team et al., 2025).

Figure 1 contrasts standard LoRA, OPLoRA, and the proposed Dynamic BB-OPLoRA.

2. Related Work

Full fine-tuning large language models is expensive in memory, compute, and storage, motivating parameter-efficient fine-tuning methods that adapt only a small number of task-specific parameters.

Parameter-efficient adaptation. Early parameter-efficient methods keep the backbone mostly frozen by inserting adapter modules into transformer layers (Houlsby et al., 2019; Pfeiffer et al., 2021) or by optimizing continuous prompts or key–value prefixes (Lester et al., 2021; Li & Liang, 2021). While effective, these approaches can introduce inference-time overhead: adapters add extra modules, whereas prompt- and prefix-based methods require additional learned tokens or states to be processed by the model.

Low-rank adaptation variants. Low-rank adaptation avoids additional modules by representing task-specific changes as additive updates to existing weight matrices. LoRA (Hu et al., 2022) freezes pretrained weights and learns low-rank updates that can be merged into the base model, avoiding additional inference latency. Subsequent variants improve parameter efficiency, optimization, capacity, or initialization: AdaLoRA (Zhang et al., 2023) adaptively allocates rank budgets across layers, LoRA-XS (Bałazy et al., 2024) trains a small matrix between SVD-derived frozen factors, DoRA (Liu et al., 2024) separates weight magnitude and direction, OLoRA (Büyükakyüz, 2024) uses QR-based orthonormal initialization, and PiSSA (Meng et al., 2024) initializes adapters from principal singular components. More recently, GOAT (Fan et al., 2025) combines SVD-structured priors with mixture-of-experts LoRA and optimization-alignment scaling.

Pretrained subspaces and forgetting. Beyond reducing adaptation cost, parameter-efficient fine-tuning should preserve pretrained capabilities, since adaptation can degrade knowledge acquired during pretraining (Biderman et al., 2024; Dou et al., 2024). Prior approaches mitigate forgetting with Fisher-weighted regularization (Kirkpatrick et al., 2017), distillation (Li & Hoiem, 2017), replay (Riemer et al., 2018), or projection-based updates such as Orthogonal Gradient Descent (Farajtabar et al., 2020). Within LoRA, MiLoRA (Wang et al., 2025) favors minor singular directions at initialization, while LoRA-Null (Tang et al., 2025) projects updates onto the null space of representative pretrained activations. Most closely related, OPLoRA (Xiong & Xie, 2026) uses double-sided weight-space projectors to restrict LoRA updates to the orthogonal complement of the top-K singular subspaces and proves exact preservation of the protected singular triples.

However, OPLoRA imposes a hard top-K constraint that treats all protected directions equally. Dynamic BB-OPLoRA keeps fixed SVD bases but relaxes this rule by splitting the top-K subspace into a rigid core and a flexible border. The core remains invariant, while the border admits budgeted Border$\times$Border adaptation. This allows controlled flexibility near the spectral cutoff, where directions can be less stable and may still contain task-relevant signal.

3. Preliminaries

In this section, we review the notation and background used throughout the paper, including LoRA, fixed top-K SVD bases, and the associated orthogonal projectors that underlie the proposed subspace-aware adaptation method.

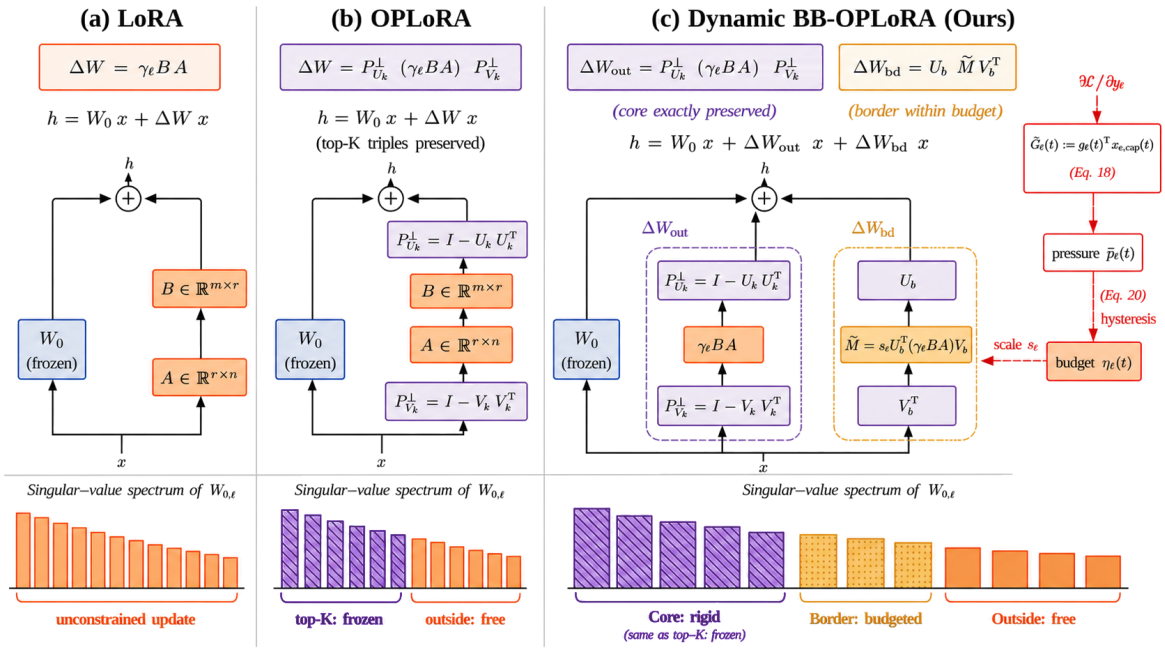

Figure 1. Comparison of LoRA, OPLoRA, and Dynamic BB-OPLoRA. LoRA applies an unconstrained low-rank update, OPLoRA freezes the full top-K subspace via orthogonal projection, and Dynamic BB-OPLoRA splits this subspace into a preserved rigid core and a flexible border whose budget is dynamically adjusted by a gradient-derived pressure signal.

3.1. Low-Rank Adaptation (LoRA)

Consider a pretrained linear layer with frozen weight $\mathbf{W}_0 \in \mathbb{R}^{m_\ell \times n_\ell}$. LoRA (Hu et al., 2022) parameterizes a trainable additive update as a rank-r factorization:

$$\mathbf{W} = \mathbf{W}_0 + \Delta W, \qquad \Delta W = \gamma_\ell \, \mathbf{B}\mathbf{A}, \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{m_\ell \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times n_\ell}$ are trainable, and $r \ll \min(m_\ell, n_\ell)$ denotes the adaptation rank. The scalar $\gamma_\ell > 0$ is a layer-specific scaling factor, typically set to $\gamma_\ell = \alpha/r$ for some hyperparameter $\alpha > 0$.

3.2. Top-K SVD bases and orthogonal projectors

Fix an integer $K \geq 0$. For $K > 0$, let $\mathbf{U}_K \in \mathbb{R}^{m_\ell \times K}$ and $\mathbf{V}_K \in \mathbb{R}^{n_\ell \times K}$ denote the matrices whose columns are the top-K left and right singular vectors of $\mathbf{W}_0$, computed once before training. Following OPLoRA (Xiong & Xie, 2026), we define the corresponding orthogonal projectors

$$\begin{aligned} \mathbf{P}_{U_K} &:= \mathbf{U}_K \mathbf{U}_K^\top, & \mathbf{P}_{V_K} &:= \mathbf{V}_K \mathbf{V}_K^\top, \\ \mathbf{P}_{U_K}^\perp &:= I_{m_\ell} - \mathbf{P}_{U_K}, & \mathbf{P}_{V_K}^\perp &:= I_{n_\ell} - \mathbf{P}_{V_K}. \end{aligned} \tag{2}$$

OPLoRA constrains the LoRA update by applying these projectors on both sides:

$$\begin{aligned} \mathbf{W} &= \mathbf{W}_0 + \Delta W_{\mathrm{OPLoRA}}, \\ \Delta W_{\mathrm{OPLoRA}} &= \mathbf{P}_{U_K}^\perp (\gamma_\ell \mathbf{B}\mathbf{A}) \mathbf{P}_{V_K}^\perp. \end{aligned} \tag{3}$$

4. Dynamic BB-OPLoRA: Rigid Core, Flexible Border

In this section, we introduce Dynamic BB-OPLoRA by first defining the core–border decomposition of the top-K singular subspace, then constructing the corresponding adapter update, and finally describing the stiffness-weighted and pressure-driven budget used to control border adaptation.

4.1. Core–border decomposition of the top-K subspace

Given a border fraction $\rho \in [0, 1]$, define the border size

$$N := \begin{cases} 0, & K = 0 \text{ or } \rho = 0, \\ \max\{1, \lfloor \rho K \rfloor\}, & \text{otherwise}, \end{cases} \tag{4}$$

and the core and border index sets

$$S_c := \{1, \ldots, K - N\}, \qquad S_b := \{K - N + 1, \ldots, K\}. \tag{5}$$

Let $\mathbf{U}_b \in \mathbb{R}^{m_\ell \times N}$ and $\mathbf{V}_b \in \mathbb{R}^{n_\ell \times N}$ denote the submatrices of $\mathbf{U}_K$ and $\mathbf{V}_K$ with columns indexed by S_b. If $K > N$, let $\mathbf{U}_c \in \mathbb{R}^{m_\ell \times (K-N)}$ and $\mathbf{V}_c \in \mathbb{R}^{n_\ell \times (K-N)}$ denote the submatrices of $\mathbf{U}_K$ and $\mathbf{V}_K$ with columns indexed by S_c.

Thus, the top-K subspace is split into a rigid core of size $K - N$ and a flexible border of size N.

4.2. Construction of the adapter update

For layer ℓ, define

$$X_\ell := \gamma_\ell \, \mathbf{B}\mathbf{A} \in \mathbb{R}^{m_\ell \times n_\ell} \tag{6}$$

as the standard LoRA update matrix. Dynamic BB-OPLoRA adapts the pretrained weight as

$$\mathbf{W} = \mathbf{W}_0 + \Delta W_\ell. \tag{7}$$

The adapter update is decomposed as

$$\Delta W_\ell = \Delta W_{\ell,\mathrm{out}} + \Delta W_{\ell,\mathrm{bd}}, \tag{8}$$

where the two terms are defined below.

Outside block. The outside component is obtained by projecting X_ℓ onto the orthogonal complements of the top-K left and right singular subspaces:

$$\Delta W_{\ell,\mathrm{out}} := \mathbf{P}^{\perp}_{U_K} \, X_\ell \, \mathbf{P}^{\perp}_{V_K}. \tag{9}$$

Border block via implicit border parameterization. The border component is induced directly by the LoRA factors $(\mathbf{B}, \mathbf{A})$ and introduces no additional trainable parameters; we refer to this construction as implicit border parameterization (IBP). Define the border-coordinate matrix

$$M_{\mathrm{raw}} := \mathbf{U}_b^\top X_\ell \mathbf{V}_b = \gamma_\ell \, (\mathbf{U}_b^\top \mathbf{B})(\mathbf{A}\mathbf{V}_b) \in \mathbb{R}^{N \times N}. \tag{10}$$

Applying the stiffness-weighted budget (SWB; Section 4.3) yields a rescaled matrix $\widehat{M} \in \mathbb{R}^{N \times N}$, which is then re-embedded into the ambient space as

$$\Delta W_{\ell,\mathrm{bd}} := \mathbf{U}_b \, \widehat{M} \, \mathbf{V}_b^\top. \tag{11}$$

The rigid core has no update action or cross-subspace interaction. Only the Border×Border and Outside×Outside blocks can be nonzero, corresponding to the budgeted border update and the projected LoRA update, respectively. During training, the border budget is updated every T steps using the pressure-driven rule in Section 4.4.

4.3. Stiffness-Weighted Budget (SWB)

Stiffness profile. Let $d_{\ell,1} \geq d_{\ell,2} \geq \cdots \geq d_{\ell,N} \geq 1$, and define the diagonal stiffness matrix

$$D_\ell := \mathrm{diag}(d_{\ell,1}, \ldots, d_{\ell,N}). \tag{12}$$

Border index $i = 1$ corresponds to global singular index $K - N + 1$ (adjacent to the rigid core), whereas $i = N$ corresponds to K (adjacent to the outside span). Since larger $d_{\ell,i}$ imposes a stronger penalty on the i-th border coordinate under the budget defined below, the nonincreasing ordering preserves greater stiffness near the core and allocates greater flexibility near the spectral boundary. For $N > 1$, we use the parameterization

$$d_{\ell,i} = 1 + \lambda \left(\frac{N - i}{N - 1} \right)^{\kappa}, \qquad i = 1, \ldots, N, \tag{13}$$

with scale $\lambda \geq 0$ and shape $\kappa > 0$. For $N = 1$, we set $d_{\ell,1} := 1$.

Budgeted rescaling. Let $\eta_\ell(t) \geq 0$ denote the border budget at step t, and let $\varepsilon_{\mathrm{cap}} > 0$ be a small constant that prevents division by zero when $\|D_\ell \, M_{\mathrm{raw}}(t) \, D_\ell\|_{\mathrm{F}}$ vanishes. Define

$$s_\ell(t) := \min\left(1, \; \frac{\eta_\ell(t)}{\|D_\ell \, M_{\mathrm{raw}}(t) \, D_\ell\|_{\mathrm{F}} + \varepsilon_{\mathrm{cap}}}\right), \tag{14}$$

and set

$$\widehat{M}(t) := s_\ell(t) \, M_{\mathrm{raw}}(t). \tag{15}$$

Then

$$\left\| D_\ell \, \widehat{M}(t) \, D_\ell \right\|_{\mathrm{F}} \leq \eta_\ell(t). \tag{16}$$

4.4. Pressure-driven dynamic budget

We now define a gradient-derived pressure signal that adapts the border budget $\eta_\ell(t)$ during training: the budget increases when the gradient indicates sustained demand for border adaptation and decreases otherwise.

Adapter-gradient proxy. Let $L(t)$ denote the training loss at step t, and let $*$ aggregate the leading batch and sequence dimensions. Let $x_{\ell,\mathrm{cap}}(t) \in \mathbb{R}^{* \times n_\ell}$ denote the adapter-branch input activation, captured before LoRA dropout and before any outside/border branching, and let $y_\ell(t) \in \mathbb{R}^{* \times m_\ell}$ denote the corresponding layer output. Define

$$g_\ell(t) := \frac{\partial L(t)}{\partial y_\ell(t)} \; \in \; \mathbb{R}^{* \times m_\ell} \tag{17}$$

as the upstream gradient flowing into $y_\ell(t)$. For a linear map $y = xW^\top$ with leading dimensions aggregated into $*$, the gradient of L with respect to W equals $g^\top x \in \mathbb{R}^{m_\ell \times n_\ell}$, where the transpose denotes contraction over $*$. Motivated by this identity, we define the adapter-gradient proxy

$$\widetilde{G}_\ell(t) := g_\ell(t)^\top \, x_{\ell,\mathrm{cap}}(t) \; \in \; \mathbb{R}^{m_\ell \times n_\ell}. \tag{18}$$

Border pressure. In the border basis, the proxy is represented by

$$C_\ell(t) := \mathbf{U}_b^\top \, \widetilde{G}_\ell(t) \, \mathbf{V}_b \; \in \; \mathbb{R}^{N \times N}, \tag{19}$$

and we define the normalized pressure ratio

$$p_\ell(t) := \frac{\|C_\ell(t)\|_{\mathrm{F}}}{\left\| \widetilde{G}_\ell(t) \right\|_{\mathrm{F}} + \varepsilon_{\mathrm{prs}}}, \tag{20}$$

where $\varepsilon_{\text{prs}} > 0$ is a small constant that prevents division by zero when $\left\|\widetilde{G}_\ell(t)\right\|_{\text{F}}$ vanishes. By construction, $p_\ell(t) \in [0, 1)$, measuring the relative magnitude of the proxy gradient aligned with the border subspace.

Exponential Moving Average (EMA) smoothing. To reduce step-level noise, fix $\beta \in (0, 1)$ and define the EMA

$$\bar{p}_\ell(t) := \beta\, \bar{p}_\ell(t-1) + (1-\beta)\, p_\ell(t), \tag{21}$$

initialized at $\bar{p}_\ell(0) := 0$.

Hysteresis budget update. Given thresholds $\tau_\uparrow > \tau_\downarrow \geq 0$, step size $\Delta\eta > 0$, and upper bound $\eta_{\max} > 0$, update the budget every $T \geq 1$ steps according to

$$\eta_\ell \leftarrow \begin{cases} \min\{\eta_\ell + \Delta\eta,\ \eta_{\max}\}, & \bar{p}_\ell > \tau_\uparrow, \\ \max\{\eta_\ell - \Delta\eta,\ 0\}, & \bar{p}_\ell < \tau_\downarrow, \\ \eta_\ell, & \text{otherwise.} \end{cases} \tag{22}$$

The gap $\tau_\uparrow - \tau_\downarrow$ forms a hysteresis band that prevents the budget from oscillating when $\bar{p}_\ell(t)$ hovers near a single threshold. In practice, budget updates are applied only within a middle training window: they start after a short warmup to use early pressure history and stop before fine-tuning ends to stabilize the final optimization phase. The complete per-step update procedure is summarized in Algorithm 1.

Algorithm 1 Dynamic BB-OPLoRA per-step update for layer ℓ

Require: Fixed bases $\mathbf{U}_K, \mathbf{V}_K, \mathbf{U}_b, \mathbf{V}_b$; LoRA factors $\mathbf{B}, \mathbf{A}$; scale γ_ℓ; stiffness D_ℓ; state $(\eta_\ell, \bar{p}_\ell)$; step t; ε_{cap}
Require: Hyperparameters $T, \beta, \tau_\uparrow, \tau_\downarrow, \Delta\eta, \eta_{\max}, \varepsilon_{\text{prs}}$
Compute $\Delta W_{\ell,\text{out}}$ via Equation (9)
if $N = 0$ **then**
  Set $\widehat{M} \leftarrow 0$, $\Delta W_{\ell,\text{bd}} \leftarrow 0$
else
  Compute M_{raw}, s_ℓ, $\widehat{M}$, and $\Delta W_{\ell,\text{bd}}$ via Equations (10), (11), (14) and (15)
end if
Set $\Delta W_\ell \leftarrow \Delta W_{\ell,\text{out}} + \Delta W_{\ell,\text{bd}}$
if $N > 0$ **and** $t \in [t_{\text{start}}, t_{\text{stop}}]$ **and** $t \bmod T = 0$ **then**
  Compute $\widetilde{G}_\ell$, C_ℓ, and p_ℓ via Equations (18) to (20)
  Update $\bar{p}_\ell$ and η_ℓ via Equations (21) and (22)
end if
return ΔW_ℓ and updated state $(\eta_\ell, \bar{p}_\ell)$

5. Experiments

We evaluate Dynamic BB-OPLoRA on commonsense reasoning, mathematical reasoning, and Python code generation. To ensure a controlled comparison with prior subspace-aware LoRA methods, we follow the experimental protocol of OPLoRA (Xiong & Xie, 2026), using the same task families, datasets, backbone model, and evaluation metrics as the OPLoRA study. We match the reported LoRA-related hyperparameters where possible, and re-run all compared methods in our own environment to ensure that the main comparisons are internally controlled. We focus our direct comparison on OPLoRA because it is the closest method to ours and the strongest reported baseline under this protocol; the original OPLoRA study compares against LoRA (Hu et al., 2022), PiSSA (Meng et al., 2024), MiLoRA (Wang et al., 2025), and LoRA-Null (Tang et al., 2025) and reports the best overall results for OPLoRA. Thus, improvements over OPLoRA provide a conservative comparison against prior subspace-aware and SVD-based LoRA variants. For completeness, the corresponding baseline tables from OPLoRA are provided in Appendix B.2.

We evaluate OPLoRA with two preserved-subspace sizes, $K = 16$ and $K = 128$, denoted as OPLoRA-16 and OPLoRA-128, respectively. All reported results are averaged over six independent runs with different random seeds. For our method, each configuration is denoted BB-OPLoRA-K-ρ, where K is the top singular subspace size and $\rho \in [0, 1]$ is the border fraction. Following Equation (4), $N = \max\{1, \lfloor \rho K \rfloor\}$ for $K > 0$ and $\rho > 0$; the top $K - N$ directions form the rigid core and the remaining N directions form the flexible border.

We evaluate four variants: BB-OPLoRA-16-0.25, BB-OPLoRA-16-0.5, BB-OPLoRA-20-0.2, and BB-OPLoRA-128-0.5. Their corresponding rigid-core and flexible-border sizes are $(12, 4)$, $(8, 8)$, $(16, 4)$, and $(64, 64)$, respectively, where each pair is reported as (core size, border size). These settings cover low-K regimes with different core–border splits, as well as a larger preserved-subspace regime with a flexible spectral border.

All additional BB-OPLoRA-specific hyperparameters are kept fixed across all experiments and are reported in Appendix B.1. Therefore, differences among BB-OPLoRA variants arise only from the choice of K and ρ, and differences between BB-OPLoRA and the baselines primarily reflect the proposed border mechanism under the shared OPLoRA experimental protocol. Also, we provide additional ablation studies in Appendix B.3.

5.1. Main Task Evaluation

We first evaluate task performance and cross-domain forgetting across three fine-tuning domains.

5.1.1. Commonsense Reasoning

Setting. The Qwen2.5-7B (Qwen team et al., 2025) is finetuned on the commonsense170k dataset (Hu et al., 2023) and evaluated on eight commonsense reasoning benchmarks:

Table 1. Commonsense reasoning results on Qwen2.5-7B across BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA, with forgetting evaluated on MathQA, MBPP, and RACE. For BB-OPLoRA-K-ρ, the two numbers denote the preserved subspace size K and border fraction ρ, respectively. Values are reported as mean $\pm$ standard deviation over six independent runs with different random seeds. The **best** and second-best results in each column are highlighted.

Method	Fine-Tuning Evaluation								Forgetting Evaluation			Avg
	BoolQ	PIQA	SIQA	HSwag	WinoG	ARC-e	ARC-c	OBQA	MathQA	MBPP	RACE	Avg
LoRA	87.23	80.85	52.32	**79.11**	76.39	76.83	54.35	47.90	50.63	56.10	41.33	63.91 $\pm$ 0.33
OPLoRA-16	87.76	80.82	52.45	79.05	76.38	77.71	55.05	47.50	50.89	57.27	41.28	64.20 $\pm$ 0.42
OPLoRA-128	87.77	80.67	52.47	78.95	76.73	77.46	55.25	48.00	50.78	55.60	**41.95**	64.15 $\pm$ 0.61
BB-OPLoRA-16-0.25	87.53	81.00	**54.31**	79.00	77.09	79.57	56.96	48.05	51.06	60.28	41.93	65.16 $\pm$ 0.34
BB-OPLoRA-16-0.5	87.94	**81.21**	54.16	78.88	**77.12**	**80.97**	**58.65**	47.93	**51.79**	**61.30**	41.23	**65.56** $\pm$ 0.36
BB-OPLoRA-20-0.2	87.90	80.83	53.28	**79.11**	76.94	78.78	55.80	**48.26**	50.69	61.03	41.16	64.89 $\pm$ 0.35
BB-OPLoRA-128-0.5	**88.02**	80.82	52.58	79.08	76.69	78.30	55.46	47.73	50.94	58.30	41.40	64.48 $\pm$ 0.32

Table 2. Mathematical reasoning results on MATH and GSM8K, with forgetting evaluated on ARC-e, ARC-c, and SIQA. In-domain, forgetting, and final averages are computed over the corresponding task groups. Values are reported as mean $\pm$ standard deviation over six independent runs with different random seeds. The **best** and second-best results in each column are highlighted.

Method	Fine-Tuning Evaluation		Forgetting Evaluation					
	MATH	GSM8K	ARC-e	ARC-c	SIQA	In Domain AVG	Forgetting Avg	Avg
LoRA	42.06	74.19	70.39	43.42	47.71	58.12	53.84	55.55 $\pm$ 0.37
OPLoRA-16	41.81	76.44	71.81	43.51	48.22	59.12	54.51	56.36 $\pm$ 0.38
OPLoRA-128	41.75	77.73	73.98	45.00	49.01	59.74	55.99	57.49 $\pm$ 0.30
BB-OPLoRA-16-0.25	42.16	79.60	**76.54**	**46.10**	**49.58**	60.88	**57.41**	**58.79** $\pm$ 0.2
BB-OPLoRA-16-0.5	41.92	79.92	73.49	44.50	49.04	60.92	55.68	57.77 $\pm$ 0.25
BB-OPLoRA-20-0.2	**42.17**	**82.22**	73.65	44.71	48.76	**62.20**	55.71	58.30 $\pm$ 0.31
BB-OPLoRA-128-0.5	41.78	81.64	74.11	44.99	48.81	61.71	55.97	58.27 $\pm$ 0.22

BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-e, ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). To measure catastrophic forgetting, we additionally evaluated the fine-tuned models on three held-out tasks from other domains: MathQA (Amini et al., 2019), MBPP (Austin et al., 2021), and RACE (Lai et al., 2017). We report results for the Qwen2.5-7B in Table 1.

Results. BB-OPLoRA improves over both LoRA and OPLoRA in terms of the overall average score. The strongest configuration is BB-OPLoRA-16-0.5, which achieves an average score of 65.56, compared with 63.91 for LoRA, 64.20 for OPLoRA-16, and 64.15 for OPLoRA-128. This configuration also obtains the best performance on PIQA, WinoGrande, ARC-e, ARC-c, MathQA, and MBPP. In particular, the improvements on ARC-e and ARC-c suggest that allowing a flexible border within the top-K subspace can improve adaptation on challenging commonsense reasoning tasks without sacrificing the preservation objective. BB-OPLoRA-16-0.25 also performs strongly, achieving the best SIQA score and the second-best overall average score among all methods.

Overall, the commonsense results show that BB-OPLoRA preserves the main benefit of OPLoRA while adding controlled flexibility near the spectral boundary. The improvement is most pronounced for BB-OPLoRA-16-0.5, which improves the overall average by 1.36 points over OPLoRA-16 and by 1.41 points over OPLoRA-128. These trends support the role of the border mechanism as a controlled relaxation of the fully rigid top-K constraint.

5.1.2. Mathematical Reasoning

Setting. The Qwen2.5-7B model is fine-tuned on the first 100K examples from the MetaMathQA dataset (Yu et al., 2023) and evaluated on MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). We report the average over these two benchmarks as the in-domain average. To measure catastrophic forgetting, we evaluate the fine-tuned models on three held-out commonsense reasoning tasks: ARC-e, ARC-c, and SIQA. Results are shown in Table 2.

Table 3. Python code generation results on MBPP and MBPP++, with forgetting evaluated on HellaSwag, OBQA, and SIQA. In-domain, forgetting, and final averages are computed over the corresponding task groups. Values are reported as mean ± standard deviation over six independent runs with different random seeds. The **best** and second-best results in each column are highlighted.

Method	Fine-Tuning Evaluation		Forgetting Evaluation					
	MBPP	MBPP++	HSwag	OBQA	SIQA	In Domain AVG	Forgetting Avg	Avg
LoRA	80.69	69.63	72.60	44.20	49.60	75.16	55.47	63.34 ± 0.15
OPLoRA-16	80.91	70.02	72.69	43.87	49.70	75.46	55.42	63.44 ± 0.14
OPLoRA-128	80.51	69.80	72.84	44.40	49.64	75.15	55.63	63.44 ± 0.32
BB-OPLoRA-16-0.25	81.57	70.15	**72.93**	**44.80**	**50.48**	75.86	**56.07**	**63.99** ± 0.35
BB-OPLoRA-16-0.5	81.17	69.84	72.74	44.20	49.76	75.51	55.57	63.55 ± 0.2
BB-OPLoRA-20-0.2	**81.66**	70.37	72.70	44.07	49.72	**76.02**	55.50	63.70 ± 0.4
BB-OPLoRA-128-0.5	81.44	**70.50**	72.70	44.13	49.55	75.97	55.46	63.66 ± 0.25

Results. BB-OPLoRA improves both in-domain mathematical reasoning and cross-domain retention relative to LoRA and OPLoRA. Among all methods, BB-OPLoRA-20-0.2 achieves the best in-domain average score, 62.20, improving over LoRA, OPLoRA-16, and OPLoRA-128 by 4.08, 3.08, and 2.46 points, respectively. This improvement is mainly driven by GSM8K, where BB-OPLoRA-20-0.2 obtains the highest score, 82.22, compared with 74.19 for LoRA, 76.44 for OPLoRA-16, and 77.73 for OPLoRA-128. For MATH, the differences are smaller, but BB-OPLoRA-20-0.2 also achieves the best score, 42.17, closely followed by BB-OPLoRA-16-0.25 with 42.16.

For forgetting evaluation, BB-OPLoRA-16-0.25 achieves the best forgetting average, 57.41, outperforming LoRA, OPLoRA-16, and OPLoRA-128 by 3.57, 2.90, and 1.42 points, respectively. This configuration also obtains the best score on all three held-out commonsense tasks: ARC-e, ARC-c, and SIQA. These results suggest that a smaller preserved subspace with a moderate border fraction can provide strong cross-domain retention after mathematical fine-tuning.

Overall, the best average score is achieved by BB-OPLoRA-16-0.25, with an overall average of 58.79. BB-OPLoRA-20-0.2 achieves the strongest in-domain mathematical reasoning performance, while BB-OPLoRA-16-0.25 provides the strongest forgetting resistance. This pattern indicates a trade-off between in-domain adaptation and cross-domain retention, and shows that the core–border split can be tuned to favor either stronger mathematical reasoning performance or stronger preservation of commonsense capabilities.

5.1.3. Python Code Generation

Setting. The Qwen2.5-7B model is fine-tuned on the CodeFeedback dataset (Zheng et al., 2024) and evaluated on MBPP and MBPP++ (Liu et al., 2023). We report the average over these two benchmarks as the in-domain average. To assess catastrophic forgetting, we evaluate the fine-tuned models on three held-out commonsense reasoning tasks: HellaSwag, OBQA, and SIQA. Results are shown in Table 3.

Results. BB-OPLoRA improves code-generation performance over LoRA and OPLoRA across all tested configurations. Among all methods, BB-OPLoRA-20-0.2 achieves the best in-domain average score, 76.02, compared with 75.16 for LoRA, 75.46 for OPLoRA-16, and 75.15 for OPLoRA-128. This configuration obtains the best MBPP score, 81.66, while BB-OPLoRA-128-0.5 achieves the best MBPP++ score, 70.50. These results suggest that the proposed border mechanism can improve task-specific code-generation performance while retaining the same LoRA backbone parameterization and shared OPLoRA training protocol.

On the cross-domain forgetting benchmarks, BB-OPLoRA-16-0.25 achieves the best forgetting average, 56.07, outperforming LoRA, OPLoRA-16, and OPLoRA-128 by 0.60, 0.65, and 0.44 points, respectively. This variant also obtains the best score on each of the three held-out commonsense tasks: HellaSwag, OBQA, and SIQA. Therefore, BB-OPLoRA-16-0.25 provides the strongest cross-domain retention after code fine-tuning.

Overall, BB-OPLoRA-16-0.25 achieves the best overall average score, 63.99, while BB-OPLoRA-20-0.2 achieves the strongest in-domain code-generation performance. This trend is consistent with the mathematical reasoning results: BB-OPLoRA-20-0.2 favors in-domain adaptation, whereas BB-OPLoRA-16-0.25 provides the strongest balance between adaptation and forgetting resistance.

5.2. Diagnostic Analysis

We next analyze the border dynamics and their effect on the pretrained singular structure.

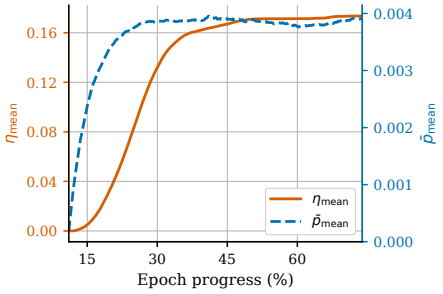

Figure 2. Border-budget dynamics during training. The plot shows the mean pressure $\bar{p}_{\text{mean}}$ and mean border budget η_{mean} over training progress.

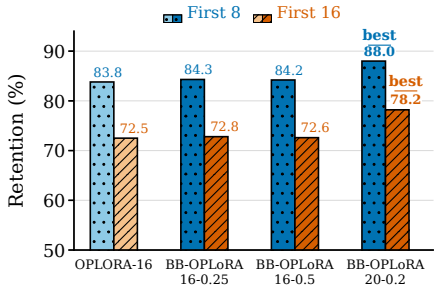

Figure 3. Retention (%) of the original first 8 and first 16 dominant singular modes after fine-tuning. Retention is measured by exact index overlap between the original and adapted top-q singular-mode sets, with $q \in \{8, 16\}$.

5.2.1. Border Budget Dynamics

To examine the training dynamics of the proposed border mechanism, we track the mean border budget η_{mean} and the mean pressure signal $\bar{p}_{\text{mean}}$ over training progress for BB-OPLoRA-16-0.5 on the Qwen2.5-7B fine-tuned for commonsense reasoning. Both η_{mean} and $\bar{p}_{\text{mean}}$ are layer-wise averages computed over all adapted layers. As shown in Figure 2, the pressure signal increases early in training, indicating gradient demand for adaptation in the border subspace. The border budget then increases during the active update window and gradually stabilizes. This behavior is consistent with the intended pressure-driven update: border flexibility is expanded when the training signal supports adaptation near the spectral boundary, and then converges to a stable fixed point.

5.2.2. Subspace Retention

We further analyze how the proposed core–border mechanism affects the singular structure of the adapted weights. Although the rigid core is protected by construction, the relative ordering of singular directions can still change during adaptation, since directions outside the rigid core may gain energy. To quantify this effect, we measure the retention of the original dominant singular modes after fine-tuning. Specifically, we report the percentage of the original first 8 and first 16 singular modes that remain among the corresponding dominant modes after adaptation.

Figure 3 shows this analysis for Qwen2.5-7B fine-tuned for commonsense reasoning. BB-OPLoRA improves retention of the dominant modes compared with OPLoRA-16. The best BB-OPLoRA configuration achieves 88.0% retention for the first 8 modes and 78.2% retention for the first 16 modes, compared with 83.8% and 72.5% for OPLoRA-16, respectively. This reflects the distinction between preserving a fixed subspace and preserving its dominance after adaptation. A fully rigid top-K constraint can shift task-driven updates to outside directions, allowing them to gain spectral energy, whereas the flexible border absorbs part of this pressure and improves retention of the dominant pretrained modes.

5.3. Computational Cost

We report adapter-side FLOPs for one activation vector in a linear layer $W_0 \in \mathbb{R}^{m\times n}$. Let r be the LoRA rank, K the preserved subspace size, and $N \leq K$ the border size. Standard LoRA costs $C_{\text{LoRA}} \approx 2r(m+n)$. The outside branch applies right and left top-K projections around the LoRA update, with cost $C_{\text{out}} \approx (4K+2r+1)(m+n)$.

BB-OPLoRA additionally includes a Border×Border branch, $x_b = xV_b$ and $\delta y_{\text{bd}} = x_b M_c^\top U_b^\top$, which costs $2N(m+n)+2N^2$. Thus, the total BB-OPLoRA adapter-side cost is $C_{\text{BB-OPLoRA}} \approx (4K+2r+2N+1)(m+n)+2N^2$. Since $N \leq K$ and typically $K, N, r \ll \min(m,n)$, the extra border cost is small relative to the frozen base transformation cost $2mn$. The SVD bases are computed once and kept fixed, and the full update matrices are not materialized. Empirically, the overhead is modest. On commonsense evaluation with the Qwen2.5-7B, LoRA took 00:22:48, OPLoRA-16 took 00:23:03, and BB-OPLoRA-16-0.5 took 00:23:18, corresponding to only about 1.1% and 2.2% overhead over LoRA, respectively.

5.4. Implementation details

All experiments were run on a single NVIDIA A100 GPU. All methods used the same training and evaluation protocol, including optimizer, batch size, sequence length, precision, number of training steps, and LoRA hyperparameters. The full hyperparameter configuration is reported in Appendix B.1. The software environment used Python 3.11.7, PyTorch 2.7.0, CUDA 12.6, PEFT 0.14.0, and Transformers 4.45.1.

6. Conclusion

We introduced Dynamic BB-OPLoRA, a subspace-aware low-rank adaptation method that protects a rigid core of dominant pretrained singular directions while allowing bud-

geted adaptation in a flexible border near the spectral boundary. A stiffness-weighted, pressure-driven budget regulates border flexibility during training, helping preserve dominant pretrained structure while improving optimization flexibility. Across commonsense reasoning, mathematical reasoning, and Python code generation, Dynamic BB-OPLoRA improves average performance over LoRA and OPLoRA while maintaining strong resistance to cross-domain forgetting.

Acknowledgments

Computational resources were provided by the NHR Center NHR@KIT on the HoreKa supercomputer under project ID p0026542. The authors gratefully acknowledge support from the DFG-funded Research Training Group RTG 2450 / GRK 2450, *Tailored Scale-Bridging Approaches to Computational Nanoscience*, at Karlsruhe Institute of Technology. A.S. and J.H. recognize support by HIDSS4Health – the Helmholtz Information & Data Science School for Health. A.S. and J.H. recognize support by the Helmholtz Association's Initiative and Networking Fund (INF) under the Helmholtz AI platform grant and the grant ZebraTwin. The funders had no role in study design, data collection, analysis, decision to publish, or preparation of the manuscript.

References

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, 2019.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Bałazy, K., Banaei, M., Aberer, K., and Tabor, J. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*, 2024.

Biderman, D., Portes, J., Ortiz, J. J. G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Büyükakyüz, K. Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*, 2024.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dou, S., Zhou, E., Liu, Y., Gao, S., Shen, W., Xiong, L., Zhou, Y., Wang, X., Xi, Z., Fan, X., et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

Fan, C., Lu, Z., Liu, S., Gu, C., Qu, X., Wei, W., and Cheng, Y. Make lora great again: Boosting lora with adaptive singular values and mixture-of-experts optimization alignment. *arXiv preprint arXiv:2502.16894*, 2025.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pp. 3762–3773. PMLR, 2020.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E.-P., Bing, L., Xu, X., Poria, S., and Lee, R. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5254–5276, 2023.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 785–794, 2017.

Lester, B., Al-Rfou, R., and Constant, N. 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.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Meng, F., Wang, Z., and Zhang, M. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pp. 487–503, 2021.

Qwen team, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Tang, P., Liu, Y., Zhang, D., Wu, X., and Zhang, D. Lora-null: Low-rank adaptation via null space for large language models. *arXiv e-prints*, pp. arXiv–2503, 2025.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, H., Li, Y., Wang, S., Chen, G., and Chen, Y. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4823–4836, 2025.

Xiong, Y. and Xie, X. Oplora: Orthogonal projection lora prevents catastrophic forgetting during parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 34088–34096, 2026.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Zheng, T., Zhang, G., Shen, T., Liu, X., Lin, B. Y., Fu, J., Chen, W., and Yue, X. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, 2024.

A. Additional Theoretical Details

A.1. Boundary Effects and Approximate Bases

Remark A.1 (Boundary multiplicities and near-ties). When the singular values of $\mathbf{W}_0$ exhibit multiplicities or near-ties, individual singular vectors need not be uniquely determined; only the associated singular subspaces are invariant. If a cluster of tied or nearly tied singular values straddles either cutoff, namely K or $K-N$, then the resulting partition into core, border, and outside components may depend on the particular basis returned by the numerical SVD routine.

Remark A.2 (Mitigation by a flexible border). The implementation dependence noted in Remark A.1 is most consequential when ambiguous singular directions are treated as rigid. Dynamic BB-OPLoRA constrains only the top $K-N$ directions to be rigid and assigns the remaining N directions within the top-K span to a flexible, budgeted border. For fixed K, increasing N shifts the rigid/flexible boundary away from the most tie-prone cutoff region, reducing sensitivity to near-ties and subspace-estimation errors concentrated near the spectral boundary.

A.2. Block-Structure Consequences

Assumption A.3 (Exact bases)**.** $(\mathbf{U}_K, \mathbf{V}_K)$ are exact orthonormal bases for the top-K left and right singular subspaces of $\mathbf{W}_0$, respectively.

Proposition A.4 (No top-K–outside interactions)**.** *Under Assumption A.3, the update ΔW defined in Equation (8) satisfies*

$$\begin{aligned} \mathbf{P}_{U_K}\,\Delta W\,\mathbf{P}_{V_K}^{\perp} &= 0, \\ \mathbf{P}_{U_K}^{\perp}\,\Delta W\,\mathbf{P}_{V_K} &= 0. \end{aligned} \tag{A1}$$

Proof. By construction,

$$\begin{aligned} \Delta W &= \Delta W_{\text{out}} + \Delta W_{\text{bd}}, \\ \Delta W_{\text{out}} &= \mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K}^{\perp}, \\ \Delta W_{\text{bd}} &= \mathbf{P}_{U_b}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_b}. \end{aligned}$$

Therefore,

$$\begin{aligned} \mathbf{P}_{U_K}\,\Delta W\,\mathbf{P}_{V_K}^{\perp} &= \mathbf{P}_{U_K}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K}^{\perp} + \mathbf{P}_{U_K}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_K}^{\perp} \\ &= \mathbf{P}_{U_K}\mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K}^{\perp} + \mathbf{P}_{U_K}\mathbf{P}_{U_b}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_b}\mathbf{P}_{V_K}^{\perp} \\ &= 0, \end{aligned}$$

where we used $\mathbf{P}_{U_K}\mathbf{P}_{U_K}^{\perp} = 0$ and $\mathbf{P}_{V_b}\mathbf{P}_{V_K}^{\perp} = 0$. Similarly,

$$\begin{aligned} \mathbf{P}_{U_K}^{\perp}\,\Delta W\,\mathbf{P}_{V_K} &= \mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K} + \mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_K} \\ &= \mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K}^{\perp}\mathbf{P}_{V_K} + \mathbf{P}_{U_K}^{\perp}\mathbf{P}_{U_b}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_b}\mathbf{P}_{V_K} \\ &= 0, \end{aligned}$$

because $\mathbf{P}_{V_K}^{\perp}\mathbf{P}_{V_K} = 0$ and $\mathbf{P}_{U_K}^{\perp}\mathbf{P}_{U_b} = 0$. □

Proposition A.5 (No core interactions)**.** *Under Assumption A.3, the update ΔW satisfies*

$$\begin{aligned} \mathbf{P}_{U_c}\,\Delta W\,\mathbf{P}_{V_c} &= 0, \\ \mathbf{P}_{U_c}\,\Delta W\,\mathbf{P}_{V_b} &= 0, \\ \mathbf{P}_{U_b}\,\Delta W\,\mathbf{P}_{V_c} &= 0. \end{aligned} \tag{A2}$$

Proof. Using the decomposition

$$\Delta W = \Delta W_{\text{out}} + \Delta W_{\text{bd}}, \qquad \Delta W_{\text{out}} = \mathbf{P}_{U_K}^{\perp}\,\Delta W_{\text{out}}\,\mathbf{P}_{V_K}^{\perp}, \qquad \Delta W_{\text{bd}} = \mathbf{P}_{U_b}\,\Delta W_{\text{bd}}\,\mathbf{P}_{V_b},$$

and the orthogonality relations

$$\mathbf{P}_{U_c}\mathbf{P}_{U_K}^{\perp} = 0, \qquad \mathbf{P}_{U_c}\mathbf{P}_{U_b} = 0, \qquad \mathbf{P}_{V_c}\mathbf{P}_{V_K}^{\perp} = 0, \qquad \mathbf{P}_{V_c}\mathbf{P}_{V_b} = 0,$$

we obtain

$$\mathbf{P}_{U_c}\,\Delta W\,\mathbf{P}_{V_c} = 0, \qquad \mathbf{P}_{U_c}\,\Delta W\,\mathbf{P}_{V_b} = 0, \qquad \mathbf{P}_{U_b}\,\Delta W\,\mathbf{P}_{V_c} = 0.$$

□

A.3. Core-Direction Invariance

Let $S_c = \{1, \ldots, K-N\}$. For each $i \in S_c$, let $(u_{\ell,i}, v_{\ell,i})$ denote the corresponding left and right singular-vector pair of $\mathbf{W}_0$.

Proposition A.6 (No update action on core singular directions)**.** *Assume Assumption A.3. Then, for every $i \in S_c$,*

$$\begin{aligned} \Delta W\, v_{\ell,i} &= 0, \\ \Delta W^\top u_{\ell,i} &= 0. \end{aligned} \tag{A3}$$

Consequently,

$$\begin{aligned} (\mathbf{W}_0 + \Delta W)\, v_{\ell,i} &= \mathbf{W}_0 v_{\ell,i}, \\ (\mathbf{W}_0 + \Delta W)^\top u_{\ell,i} &= \mathbf{W}_0^\top u_{\ell,i}. \end{aligned} \tag{A4}$$

Proof. Fix $i \in S_c$. Since $v_{\ell,i} \in \mathrm{range}(\mathbf{V}_c) \subseteq \mathrm{range}(\mathbf{V}_K)$, we have $\mathbf{P}^\perp_{V_K} v_{\ell,i} = 0$. Hence,

$$\begin{aligned} \Delta W_{\mathrm{out}} v_{\ell,i} &= \mathbf{P}^\perp_{U_K} X_\ell \mathbf{P}^\perp_{V_K} v_{\ell,i} \\ &= 0. \end{aligned}$$

Moreover, $v_{\ell,i} \perp \mathrm{range}(\mathbf{V}_b)$, so $\mathbf{V}_b^\top v_{\ell,i} = 0$. Therefore,

$$\begin{aligned} \Delta W_{\mathrm{bd}} v_{\ell,i} &= \mathbf{U}_b \widehat{M} \mathbf{V}_b^\top v_{\ell,i} \\ &= 0. \end{aligned}$$

Thus $\Delta W v_{\ell,i} = 0$.

The transpose identity follows analogously. Since $u_{\ell,i} \in \mathrm{range}(\mathbf{U}_c) \subseteq \mathrm{range}(\mathbf{U}_K)$, we have $\mathbf{P}^\perp_{U_K} u_{\ell,i} = 0$ and $\mathbf{U}_b^\top u_{\ell,i} = 0$. Therefore,

$$\begin{aligned} \Delta W_{\mathrm{out}}^\top u_{\ell,i} &= \mathbf{P}^\perp_{V_K} X_\ell^\top \mathbf{P}^\perp_{U_K} u_{\ell,i} \\ &= 0, \end{aligned} \qquad \begin{aligned} \Delta W_{\mathrm{bd}}^\top u_{\ell,i} &= \mathbf{V}_b \widehat{M}^\top \mathbf{U}_b^\top u_{\ell,i} \\ &= 0. \end{aligned}$$

Hence $\Delta W^\top u_{\ell,i} = 0$. The identities in Equation (A4) follow immediately. □

Remark A.7 (Scope of the invariance). Proposition A.6 applies only to the rigid core. The action of $\mathbf{W}_0 + \Delta W$ on the border subspace is not invariant in general; it is constrained only by the budget bound in Equation (16), and the singular values and vectors associated with S_b may change. Thus, Dynamic BB-OPLoRA is more expressive than variants that freeze the entire top-K subspace, with the additional flexibility concentrated near the spectral boundary. Under multiplicities or near-ties, these statements should be interpreted at the subspace level rather than for individual singular vectors.

B. Additional Experimental Details

B.1. Hyperparameter Configuration

This section reports the shared training hyperparameters and the BB-OPLoRA-specific hyperparameters used across all experiments.

Warmup-calibrated pressure threshold. The pressure threshold is calibrated from an initial warmup window. During this window, the pressure ratio $p_\ell(t)$ is recorded, but the border budget η_ℓ is not updated. Let $\mathcal{W}$ denote the set of warmup steps before the budget-update window begins. We define the warmup mean pressure as

$$\bar{p}^{\mathrm{warm}} = \frac{1}{|\mathcal{W}|} \sum_{t \in \mathcal{W}} p_{(t)}.$$

The layer-specific upper threshold is then set to

$$\tau_\uparrow = 1.3\, \bar{p}^{\mathrm{warm}}.$$

Budget update interval. We specify the budget-update frequency as a relative fraction of an epoch and convert it to an integer step interval before training. Let S_{epoch} denote the number of optimizer steps in one epoch. We use

$$T = \max\{1,\, \mathrm{round}(0.003\, S_{\mathrm{epoch}})\}\,.$$

The pressure-driven budget update in Equation (22) is then applied every T optimizer steps, i.e., when $t \bmod T = 0$.

Table B1. Hyperparameter configuration used across all experiments.

Hyperparameter	Commonsense	Math	Python
LoRA rank (r)	32	64	32
Scaling factor (α)	32	64	32
LoRA dropout	0.05	0.05	0.05
Learning rate	2e-4	3e-4	2e-4
Batch size	32	32	32
Epochs	1	1	1
Weight decay	0.0	0.0	0.0
Learning rate scheduler	Cosine	Cosine	Cosine
Warmup ratio	0.03	0.03	0.03
Precision	bfloat16	bfloat16	bfloat16

Table B2. LoRA target modules used in all experiments.

Module group	Target modules
Attention	`q_proj`, `v_proj`, `o_proj`
MLP	`up_proj`, `down_proj`

Table B3. BB-OPLoRA-specific hyperparameters used across all experiments.

BB-OPLoRA hyperparameter	Value
Stiffness scale λ	1.0
Stiffness shape κ	2.0
EMA coefficient β	0.95
Upper pressure threshold $\tau_{\uparrow}$	$1.3\,\bar{p}_{\ell}^{\mathrm{warm}}$
Lower pressure threshold $\tau_{\downarrow}$	0
Budget step size $\Delta\eta$	0.0125
Maximum border budget $\eta_{\max}$	0.8
Pressure numerical constant $\varepsilon_{\mathrm{prs}}$	10^{-8}
SWB numerical constant $\varepsilon_{\mathrm{cap}}$	10^{-8}
Budget-update window start	11% of training
Budget-update window end	74% of training
Relative update T	$\max\{1, \mathrm{round}(0.003\, S_{\mathrm{epoch}})\}$

B.2. OPLoRA Baseline Tables

For completeness, we provide the corresponding baseline tables from Xiong & Xie (2026). These numbers are not used as the primary comparison in our main results, because our main experiments use re-run baselines under our implementation, environment, and six-seed evaluation protocol.

Table B4. Comparison of LoRA-based fine-tuning methods on commonsense reasoning and forgetting benchmarks. All models are fine-tuned from the Qwen2.5 7B. Bold and underlined values indicate the best and second-best scores. Values are taken from OPLoRA (Xiong & Xie, 2026).

	Fine-Tuning Evaluation								Forgetting Evaluation			
Method	**BoolQ**	**PIQA**	**SIQA**	**HSwag**	**WinoG**	**ARC-e**	**ARC-c**	**OBQA**	**MathQA**	**MBPP**	**RACE**	**Avg**
Baseline Methods												
LoRA	**86.75**	**79.76**	51.43	78.12	73.95	76.26	55.03	45.8	50.35	58.6	39.62	63.24
PiSSA	75.04	75.08	52.04	71.75	75.92	69.23	43.85	42.0	31.86	0.6	**40.57**	52.54
MiLoRA	86.05	77.36	**53.42**	74.73	76.47	71.42	52.13	44.4	46.47	2.6	39.62	56.79
LoRA-Null	83.30	77.61	51.57	75.67	74.51	75.84	54.83	46.8	50.24	19.8	40.11	59.12
OPLoRA-16	86.64	79.38	52.56	**79.01**	**76.64**	**78.54**	55.46	**47.8**	50.92	**59.6**	40.28	**64.26**
OPLoRA-128	86.64	**79.76**	51.43	77.62	74.03	76.14	**56.14**	47.4	**51.29**	59.2	39.23	63.53

Table B5. Comparison of LoRA-based methods fine-tuned on MetaMathQA (first 100K samples) using the Qwen2.5 7B. Best and second-best scores are highlighted in bold and underlined. Values are taken from OPLoRA (Xiong & Xie, 2026).

	Fine-Tuning Eval		Forgetting Eval			
Method	**MATH**	**GSM8K**	**ARC-e**	**ARC-c**	**SIQA**	**Avg**
Baseline Methods						
LoRA	45.96	82.18	81.40	50.42	51.28	62.25
PiSSA	32.74	74.45	72.47	38.65	44.37	52.54
MiLoRA	44.04	82.48	81.27	48.38	51.17	61.47
LoRA-Null	43.72	75.12	73.22	49.89	46.44	57.68
OPLoRA-16	**47.34**	**83.70**	81.86	**50.59**	51.43	62.98
OPLoRA-128	46.72	**83.70**	**82.24**	50.20	**52.92**	**63.16**

Table B6. Performance comparison of LoRA-based methods fine-tuned on CodeFeedback dataset using the Qwen2.5 7B. Best and second-best results are highlighted in bold and underlined. Values are taken from OPLoRA (Xiong & Xie, 2026).

	Fine-Tuning Eval		Forgetting Eval			
Method	**MBPP**	**MBPP++**	**HSwag**	**OBQA**	**SIQA**	**Avg**
Baseline Methods						
LoRA	80.2	68.3	79.1	**47.4**	54.4	65.88
PiSSA	68.3	58.2	78.0	43.2	51.7	59.88
MiLoRA	**80.7**	66.7	78.6	46.2	55.1	65.46
LoRA-Null	79.6	65.7	78.0	46.0	54.8	64.82
OPLoRA-16	80.2	**69.0**	79.1	**47.4**	55.0	66.14
OPLoRA-128	80.4	68.8	**79.2**	**47.4**	**55.5**	**66.26**

B.3. Ablation Studies

We conduct an ablation study on the Qwen2.5-7B in the commonsense reasoning setting to isolate the roles of the stiffness-weighted budget and the pressure-driven dynamic update. The full BB-OPLoRA-20-0.2 model uses both components. In the first ablation, we remove stiffness weighting by setting $d_{\ell,i} = 1$ for all border coordinates in Equation (13), so all border directions are penalized equally. In the second ablation, we remove the pressure-driven update and use a fixed maximum border budget $\eta_\ell = 0.8$ for all layers throughout training. If the border budget is set to zero, the Border$\times$Border component is suppressed and the entire top-K subspace is treated as rigid, recovering the OPLoRA-style constraint. If this rigid top-K constraint is also removed, the update is no longer projected away from the pretrained singular subspace and reduces to standard LoRA.

As shown in Table B7, the full method achieves the best overall average. This suggests that the two components are complementary: uniform border stiffness weakens the benefit of the border mechanism, while a fixed maximum budget is less effective than adapting the budget according to the training signal.

Table B7. Ablation results on the Qwen2.5-7B for commonsense reasoning. The variant without SWB sets $d_{\ell,i} = 1$ for all border coordinates, and the variant without dynamic budget uses a fixed maximum budget $\eta_\ell = 0.8$ for all layers. Values are reported as mean $\pm$ standard deviation over six independent runs with different random seeds. The **best** and second-best results are highlighted.

Method	**Fine-Tuning Evaluation**								**Forgetting Evaluation**			**Avg**
	BoolQ	**PIQA**	**SIQA**	**HSwag**	**WinoG**	**ARC-e**	**ARC-c**	**OBQA**	**MathQA**	**MBPP**	**RACE**	**Avg**
LoRA	87.23	80.85	52.32	**79.11**	76.39	76.83	54.35	47.90	50.63	56.10	41.33	63.91 $\pm$ 0.33
OPLoRA-16	87.76	80.82	52.45	79.05	76.38	77.71	55.05	47.50	**50.89**	57.27	41.28	64.20 $\pm$ 0.42
BB-OPLoRA-20-0.2	**87.90**	80.83	**53.28**	**79.11**	76.94	**78.78**	**55.80**	**48.26**	50.69	**61.03**	41.16	**64.89** $\pm$ 0.35
BB-OPLoRA-20-0.2 w/o SWB	87.37	80.55	53.24	79.05	76.72	76.85	55.72	47.70	50.82	54.90	**41.56**	64.05 $\pm$ 0.40
BB-OPLoRA-20-0.2 w/o dynamic budget	87.65	**80.97**	52.71	78.81	**77.01**	76.87	55.21	47.95	50.75	59.55	41.22	64.43 $\pm$ 0.26

C. Limitations and Future Work

- Relative to a fully rigid top-K preserved subspace (OPLoRA), the flexible border introduces additional degrees of freedom, enabling greater expressivity within the top-K span (subject to the stiffness-weighted budget). While this added flexibility mitigates practical brittleness near the spectral cutoff (e.g., under approximate SVD or near-ties), it does not theoretically eliminate underlying non-uniqueness or leakage phenomena.
- The implicit border parameterization couples the border and outside-subspace updates through the shared LoRA factors. While this design avoids the need for an independent trainable border matrix (keeping the parameter count identical to standard LoRA), it prevents the border and outside adaptations from being independently parameterized.
- Dynamic BB-OPLoRA introduces several hyperparameters governing the dynamic budget, including update windows, pressure thresholds, and update intervals. These choices are explicitly motivated by training dynamics: for example, delaying budget updates ensures the pressure estimate relies on stable training history rather than noisy initial gradients, while pausing updates near the end of fine-tuning stabilizes final optimization. Future work could automate these controls using data-driven criteria, such as pressure signal convergence or validation-based triggers.
- To enable a controlled comparison, our experiments focus on the same benchmark suite and backbone models utilized in the OPLoRA protocol. Future work will extend this evaluation to larger, more diverse LLM backbones across a wider array of tasks. A broader evaluation will help characterize the domains where BB-OPLoRA provides the most benefit, and how tuning the core size (K) and border fraction (ρ) influences the stability-plasticity trade-off.