Rethinking LLM Fine-Tuning via Weight Space Reparameterization: Preserving Safety during Downstream Adaptation

Min-Seong Kim* **Jongbok Won*** **Jeesup Park** **Yonghee Choi** **Dong-Jun Han**
Yonsei University
{minseongkim, jongbok.won, jeesup, 2025321337, djh}@yonsei.ac.kr

Abstract

Fine-tuning large language models (LLMs) on downstream tasks often weakens their previously aligned safety behavior, creating an inherent trade-off between task adaptation and safety preservation. Existing approaches attempt to mitigate this issue by identifying safety-relevant neurons, layers, or update directions in the original parameter space. However, safety-related information is often distributed across many directions in this space, causing downstream updates to overwrite parameters that encode safety behavior. We propose **WSR-Tune**, a Weight Space Reparameterization-based fine-tuning framework that preserves safety during downstream adaptation. Our approach constructs a safety-conditioned basis and reparameterizes the weight matrices such that safety-relevant information becomes concentrated in a small subset of directions. We then identify safety-critical directions and freeze them in the reparameterized space, while updating the remaining complementary directions for downstream adaptation. By explicitly structuring the parameter space to disentangle safety and task information, our method reduces interference between objectives and enables simultaneous preservation of safety and improvement in downstream performance. Experimental results show that our method achieves a more favorable trade-off between downstream performance and safety retention, demonstrating its effectiveness for reliable LLM fine-tuning.

1 Introduction

Large Language Models (LLMs) are increasingly being deployed in real-world settings where safety is critical [1]. As their capabilities continue to advance, LLMs are expected to play important roles across diverse domains such as healthcare, education, and software engineering [2, 3, 4, 5]. Without adequate safety tuning, LLMs can provide harmful assistance, violate policies, and exhibit unsafe behavior at scale [6]. Ensuring that these models remain helpful while consistently refusing harmful or disallowed requests has therefore become a central challenge in modern LLM development.

In practice, LLM deployment typically involves multiple post-training stages: a base model is first improved through instruction tuning and/or safety tuning, and is then further adapted to user-specific downstream tasks [7]. However, such downstream adaptation can degrade previously learned safety behaviors [8]. This issue becomes more critical as attempts to misuse LLMs have become increasingly sophisticated [9, 10, 11]. Recent work has introduced a wide range of jailbreak strategies, including manually crafted prompts [12], automated prompt optimization [13, 14], adversarial fine-tuning [15, 16], and other adaptive attacks [17, 18, 19] designed to elicit harmful or policy-violating

*Equal contribution.
Code available at https://github.com/min-seong-kim/WSR-Tune

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

outputs even from safety-tuned models [20]. These threats suggest that safety must be preserved throughout model adaptation, deployment, and continual improvement.

A central challenge in LLM fine-tuning is thus preserving safety during downstream adaptation, motivating a growing body of safety-preserving methods. *Data- and optimization-based methods* incorporate safety supervision or select safety-compatible training data to guide downstream optimization [21, 8, 22]. *Parameter- and update-constrained methods* protect safety-relevant components or restrict update directions to limit safety degradation [23, 24, 25]. *Low-rank and post-hoc model-space methods* constrain low-rank updates or modify fine-tuned weights to preserve or restore safety [26, 27, 28, 29]. These methods show that safety can be partially preserved or recovered during downstream adaptation.

Challenges. However, despite prior efforts, existing methods share a key limitation: they operate entirely in the *original parameterization* of the model. For example, consider a safety-tuned 2×2 weight matrix W from an arbitrary layer, which can be expressed as

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = w_{11}b_{11} + w_{12}b_{12} + w_{21}b_{21} + w_{22}b_{22}, \tag{1}$$

where $\{w_{ij}\}$ are the model parameters, $\{b_{ij}\}$ are the canonical basis matrices, and each b_{ij} has a single nonzero entry of 1 at position (i, j) and zeros elsewhere (e.g., $b_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $b_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$).

In this representation, safety-related information is not localized to a specific coordinate w_{ij}, but is instead distributed across multiple directions in the parameter space (i.e., across w_{11}, w_{12}, w_{21}, w_{22}). Consequently, even if prior methods identify and preserve safety-relevant parameters, downstream adaptation can still overwrite other parameters that encode safety behavior. In other words, preserving a subset of parameters in the original space is often insufficient, as the parameterization does not provide a clear separation between safety-preservation and task adaptation.

Table 1 empirically demonstrates this limitation. Full-parameter adaptation on a downstream task severely degrades safety, while freezing 10%, 30%, and 50% of the original-space parameters only partially alleviates this degradation. In contrast, applying the same freeze-and-adapt procedure after reparameterization preserves safety much more effectively while maintaining sufficient capacity for downstream adaptation. These results suggest that robust safety preservation cannot be achieved solely by selecting parameters to freeze in the original space; rather, it requires reparameterizing the model so that safety-preserving and task-adaptive directions are more easily disentangled. We thus pose the following key question:

Table 1: Impact of freezing safety-relevant parameters on safety preservation after downstream fine-tuning. The reported values are averaged over Direct, AutoDAN, PAIR, and PAP attacks using safety-tuned Llama-2-7B.

Fine-tuning setting	**Safety** $\downarrow$
FT (0% frozen)	53.84
FT (10% frozen)	33.95
FT (30% frozen)	21.44
FT (50% frozen)	18.43
WSR-Tune (10% frozen)	**14.67**

Can we construct a weight-space representation that concentrates safety-relevant information into a compact subspace, while leaving the remaining directions available for downstream adaptation?

Contributions. Motivated by this question, we propose **WSR-Tune**, a safety-aware LLM fine-tuning framework based on *weight-space reparameterization*. Specifically, we replace the above canonical parameterization in Equation (1) with a reparameterized form as

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \tilde{w}_{11}\phi_{11} + \tilde{w}_{12}\phi_{12} + \tilde{w}_{21}\phi_{21} + \tilde{w}_{22}\phi_{22}, \tag{2}$$

where ϕ_{ij} indicates a new basis and $\tilde{w}_{ij}$ are the corresponding weights in the reparameterized space. Unlike the canonical basis where each b_{ij} contains a single nonzero entry, the new basis $\{\phi_{ij}\}$ can consist of directions with multiple nonzero entries.

Building on this framework, we construct a safety-conditioned basis $\{\phi_{ij}\}$ and reparameterize the weight space from $\{w_{ij}\}$ to $\{\tilde{w}_{ij}\}$ such that safety-relevant behavior is concentrated in a small subset of directions (i.e., only a few basis directions in $\{\phi_{ij}\}$ capture safety-relevant information), rather than being diffusely distributed. This safety-conditioned reparameterization is our key contribution: it restructures the weight space to better separate safety-preserving and task-adaptive directions. Under this reparameterized space, we selectively freeze the coefficients associated with safety-critical

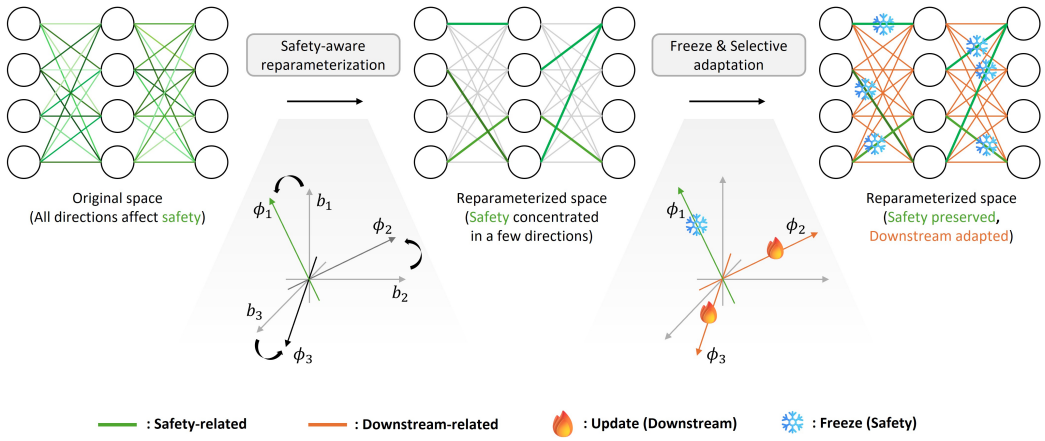

Figure 1: Overview of **WSR-Tune**. The weight space is reparameterized into a safety-conditioned basis, in which safety-relevant information is concentrated in a subset of coefficients. Safety-critical coefficients are then identified and frozen, while the remaining coefficients are updated for downstream adaptation. This reduces interference between safety and downstream task information during fine-tuning, improving task performance with minimal safety degradation.

directions while updating the complementary directions for downstream adaptation, as illustrated in Figure 1. As a result, our approach reduces interference between previously learned safety behavior and downstream adaptation, enabling more effective safety-preserving fine-tuning.

Our WSR-Tune can be seamlessly integrated with existing downstream adaptation methods originally designed for the original parameter space, by applying them in the reparameterized space without modifying their update rules. Experimental results demonstrate that our method better preserves safety during downstream fine-tuning and achieves a more favorable trade-off between downstream performance and safety retention, compared to methods operating in the original parameter space.

2 Related Works

Preserving safety during downstream adaptation. Recent efforts to mitigate safety degradation during downstream fine-tuning [15, 30] broadly fall into three categories. *Data- and optimization-based defenses* enforce safety via supervision or sample filtering: SafeInstr [21] mixes safety examples into downstream training, Lisa [8] applies proximal regularization, SEAL [22] selects downstream data through bilevel optimization, and SPARD [31] selects safety examples and applies corrective updates. *Gradient and subspace projection methods* restrict updates to safe sub-manifolds: SafeGrad [25] and SPF [32] filter conflicting task gradients via alignment directions or gradient-derived subspaces, while SaLoRA [27] and GuardSpace [33] project low-rank updates into activation-derived null spaces; SafeAnchor [34] combines Fisher-based gradient constraints with corrective replay for continual adaptation. *Parameter-level protection and post-hoc methods* preserve or restore safety at the weight level: SN-Tune [23] freezes safety-critical neurons, while AsFT [24] regularizes updates relative to an alignment direction. After fine-tuning, SafeLoRA [26] applies alignment-aware projection to LoRA updates, SafeDelta [28] selects task-delta components and compensates for residual safety loss, and RESTA [29] adds a safety vector through task arithmetic. However, most of these methods still attempt to protect safety within the original parameterization of the model. In contrast, our approach addresses this limitation by reparameterizing the weight space itself, enabling safety preservation through a more structured separation between safety-critical and task-adaptive components. We discuss a broader range of methods for preserving safety during downstream adaptation in Appendix J.

Analyzing safety in representation and parameter space. Other recent work analyzes LLM safety through structured dimensions and spaces [35, 36]. A recent study [37] shows that safety-aligned behavior is not governed by a single refusal direction, but by a multi-dimensional residual space in which a dominant direction captures refusal while additional orthogonal directions encode other safety-related features. A complementary line of work [38] studies safety directly in the original weight space and shows that safety-critical regions can be identified not only at the neuron level but

also at the rank level. Safety appears to be highly entangled with general learning rather than isolated in a distinct linear subspace, as both weight-space and activation-space analyses show substantial overlap between safety-related and utility-related directions [39]. Overall, these studies suggest that safety is distributed across directions, ranks, and subspaces rather than being cleanly localized to a small set of canonical parameters [36, 40]. This in turn implies that post-hoc selection of important components in the original space may be insufficient for robust safety preservation.

Subspace-constrained adaptation. Although not focused on safety, a line of prior work studies how to reduce interference by constraining parameter updates within structured subspaces. AlphaEdit [41] projects updates into the null space of preserved knowledge, while orthogonal fine-tuning methods, such as PSOFT [42] orthogonally transforms the principal subspace of pretrained weights to preserve its geometry. These methods mitigate interference by carefully selecting update directions, but they still operate within the original parameterization of the model. In a related but distinct direction, [43] explores weight space rotation primarily in the context of class-incremental learning. While this highlights the potential of restructuring the parameter space, it is not designed to address safety preservation during downstream adaptation. These works motivate our view that the safety problem is not only about identifying certain parameters, but also about choosing a suitable space in which preservation and adaptation can be disentangled.

3 Algorithm

3.1 Problem setup

We consider adapting an LLM to downstream tasks while preserving its previously learned safety behaviors. Let f_{W_0} denote a safety-tuned model with parameters W_0, where "safety-tuned" indicates that the model has learned to refuse harmful requests through prior safety post-training. In many practical settings, such a model is subsequently adapted to a downstream task using a task-specific dataset D_{task}. However, standard fine-tuning on D_{task} often degrades the model's previously learned safety behaviors, even when the downstream data itself is benign. This leads to an inherent trade-off between improving downstream performance and preserving safety during adaptation.

Formally, let $\mathcal{L}(W; D)$ denote the loss of model W on dataset D, and let D_{safe} denote a safety-oriented dataset used to preserve safe behavior. In conventional fine-tuning, adaptation is performed by applying a parameter update ΔW to W_0 in the original weight space. Our objective is to improve downstream performance while preserving safety:

$$\min_{\Delta W} \ \mathcal{L}(W_0 + \Delta W; D_{\text{task}}) \ \text{ subject to } \ \mathcal{L}(W_0 + \Delta W; D_{\text{safe}}) \leq \mathcal{L}(W_0; D_{\text{safe}}) + \epsilon, \tag{3}$$

where $\epsilon \geq 0$ denotes a tolerable increase in safety loss.

3.2 Motivation and Insight

Despite extensive efforts, existing approaches [21, 23, 26, 28, 29] operate in the original parameter space. To understand the limitation of this representation, consider an arbitrary $m \times n$ weight matrix

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix} = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} b_{ij} \tag{4}$$

of a *safety-tuned* model, where $\{w_{ij}\}$ are the model parameters and $\{b_{ij}\}$ are the canonical basis matrices. Each b_{ij} has a single nonzero entry of 1 at position (i, j) and zeros elsewhere.

In this coordinate system, safety-related information is not localized to a specific coordinate w_{ij}, but is instead distributed across multiple directions in the parameter space. Consequently, downstream fine-tuning (even when applied to only a subset of parameters) can still overwrite safety-related information encoded in other parameters, leading to degraded safety. This hypothesis is supported by our preliminary results in Table 1, where freezing a certain fraction of parameters in the original space is insufficient to adequately preserve safety. This raises the following question:

Is the original coordinate system the most suitable for fine-tuning, or can we construct a better basis that enables effective downstream adaptation without compromising safety?

3.3 WSR-Tune: Reparameterized Fine-Tuning for Safety Preservation

We propose WSR-Tune, which moves beyond identifying *what* to protect and instead reconsiders *how* the parameter space itself is represented. To this end, we transform the weight space into a safety-conditioned coordinate system in which safety-relevant behavior is more compactly concentrated, making it easier to preserve during downstream fine-tuning. We replace the canonical parameterization

$$W = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} b_{ij} \xrightarrow{\text{Reparameterization}} W = \sum_{i=1}^{m}\sum_{j=1}^{n} \tilde{w}_{ij}\phi_{ij}, \tag{5}$$

where $\{\phi_{ij}\}$ denotes a new basis and $\{\tilde{w}_{ij}\}$ are the corresponding weights in the reparameterized space. Unlike the canonical basis, where each b_{ij} contains a single nonzero entry, the new basis matrices can represent structured directions spanning multiple parameters. The reparameterized weight matrix $\tilde{W}$ can be expressed in terms of the new basis $\{\phi_{ij}\}$ as

$$\tilde{W} = \begin{bmatrix} \tilde{w}_{11} & \cdots & \tilde{w}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{w}_{m1} & \cdots & \tilde{w}_{mn} \end{bmatrix} = \begin{bmatrix} \langle W, \phi_{11}\rangle & \cdots & \langle W, \phi_{1n}\rangle \\ \vdots & \ddots & \vdots \\ \langle W, \phi_{m1}\rangle & \cdots & \langle W, \phi_{mn}\rangle \end{bmatrix}, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product between two matrices.

This transformation aims to make safety-critical directions more identifiable and concentrated in the reparameterized space, while leaving the complementary directions with sufficient capacity for adaptation. Our key idea is to perform adaptation by updating a subset of $\tilde{w}_{ij}$, rather than directly modifying the original parameters w_{ij}, thereby better preserving safety during downstream adaptation. The high-level idea of WSR-Tune is illustrated in Figure 1. In the following, we describe how our reparameterization is constructed and applied in practice.

Safety-aware weight reparameterization. Our first step is to construct a new basis $\{\phi_{ij}\}$ in which safety-relevant information is concentrated in a small subset of directions. Let $W_l \in \mathbb{R}^{m_l \times n_l}$ denote the weight matrix of the l-th selected layer. For each sample x_i and token position $t \in [T_i]$, let $h_l(x_i, t) \in \mathbb{R}^{n_l}$ denote the input activation of the t-th token to layer W_l, i.e., the hidden representation of that token produced by the preceding layers before being transformed by W_l. We then collect these token-wise activations to form

$$H_l = [\, h_l(x_i, t) \,]_{(i,t)\in\mathcal{T}_{\text{safe}}} \in \mathbb{R}^{n_l \times M}, \tag{7}$$

where $\mathcal{T}_{\text{safe}}$ denotes the set of valid sample-token indices from the safety examples in D_{safe}, T_i denotes the number of valid tokens in sample x_i, and $M = \sum_{i=1}^{N} T_i$ is the total number of collected token-wise activations. Here, D_{safe} consists of harmful prompts paired with safe refusal responses. We then compute the activation covariance and perform singular value decomposition:

$$H_l H_l^\top = U_l \Sigma_l U_l^\top, \tag{8}$$

where $U_l \in \mathbb{R}^{n_l \times n_l}$ is an orthonormal matrix whose columns are ordered according to the singular values in Σ_l. Figure 2 shows that the singular values exhibit a clear low-rank structure, motivating the use of U_l as an activation-induced basis.

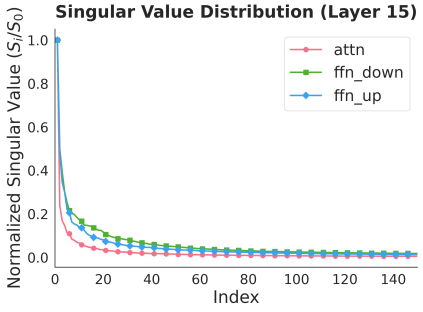

Figure 2: For each layer, singular values are sorted in descending order and normalized by the largest value. Llama-2-7B-Chat model is used.

We thus reparameterize the weight matrix as

$$\widetilde{W}_l = V_l^\top W_l U_l, \tag{9}$$

where U_l is defined above, $V_l \in \mathbb{R}^{m_l \times m_l}$ is an arbitrary unitary matrix, and $\widetilde{W}_l \in \mathbb{R}^{m_l \times n_l}$ denotes the weights in the reparameterized space. Let $\{v_i\}_{i=1}^{m_l}$ and $\{u_j\}_{j=1}^{n_l}$ denote the column vectors of V_l and U_l, respectively. Introducing V_l allows us to define a complete orthonormal basis over the matrix space via outer products of row and column directions, i.e.,

$$\phi_{ij} = v_i u_j^\top, \tag{10}$$

ensuring that each (i, j) corresponds to a distinct basis element. In this work, we set $V_l = I$ for simplicity, which avoids multiple equivalent representations induced by different choices of V_l; Appendix C shows that WSR-Tune remains robust to alternative choices of V_l. Under this formulation, each entry $\widetilde{W}_l(i, j) = \tilde{w}_{ij}$ represents the reparameterized weight along the basis element ϕ_{ij}. This reparameterization defines a coordinate system aligned with activation statistics, in which safety-related information can be represented more compactly. The new coordinate framework is constructed only once before downstream fine-tuning, which is then kept fixed throughout adaptation.

Preserving safety in the reparameterized space. Under this reparameterization, downstream adaptation is applied to the coefficients of $\widetilde{W}_l$, rather than to W_l. To preserve safety, within each layer, we estimate the importance of each coefficient using its gradient magnitude on a safety corpus:

$$G_l = \sum_{x \in D_{\text{safe}}} \left| \frac{\partial \mathcal{L}(x)}{\partial \widetilde{W}_l} \right|. \tag{11}$$

Here, $G_l(i, j)$ quantifies the sensitivity of the safety loss to $\widetilde{W}_l(i, j)$, and larger values indicate greater importance for safety retention. Based on G_l, we construct a binary mask $M_l \in \{0, 1\}^{m_l \times n_l}$, where $M_l(i, j) = 1$ denotes a safety-critical coefficient to be preserved in the reparameterized space. In practice, we select the top-ρ fraction of entries according to G_l within each layer, where $\rho \in [0, 1]$ controls the trade-off between safety retention and downstream adaptability.

Downstream adaptation in the complementary subspace. During downstream fine-tuning, we update only the complementary coefficients while blocking gradients through the masked ones:

$$\widetilde{W}_l \leftarrow \text{stopgrad}(\widetilde{W}_l \odot M_l) + \widetilde{W}_l \odot (1 - M_l), \tag{12}$$

where $\odot$ denotes element-wise multiplication. This constrains adaptation to the complementary subspace, while preserving the coefficients most critical for safety in the reparameterized space.

Our WSR-Tune differs from standard fine-tuning by reparameterizing the weight space for safety preservation. First, it decouples basis construction from task adaptation: a safety-conditioned basis is constructed once using safety data and remains fixed throughout downstream adaptation. Second, protection is applied in the reparameterized space, where coefficients are aligned with safety-relevant activation directions. This enables targeted safety preservation while leaving the remaining directions available for downstream updates, thereby reducing interference between safety preservation and downstream adaptation. Appendix D further examines how this coordinate change enables more safety-selective protection under the same masking rule.

3.4 Discussion

Extension to parameter-efficient fine-tuning. To support practical downstream adaptation under limited memory and compute budgets, we extend WSR-Tune beyond full-parameter fine-tuning to parameter-efficient fine-tuning. We apply safety-conditioned reparameterization and protection to LoRA [44] factors, enabling safety-preserving adaptation with a small set of trainable parameters. Detailed implementation and optimization procedures are provided in Appendix E.

Applicability to different layers. WSR-Tune can be applied not only to MLP layers but also to attention layers, including the key, query, and value projection matrices. In our experiments in Section 4, we apply WSR-Tune to both attention and MLP projections and evaluate its effectiveness.

Compatibility with prior works. We validate this in Section 4, which highlights the plug-and-play compatibility of WSR-Tune, allowing it to seamlessly enhance prior safety-preserving methods without altering their native mechanisms.

Freeze ratio. The freeze ratio ρ controls the fraction of coefficients preserved for safety. In Section 4, we study the impact of varying ρ, demonstrating its role in balancing safety and downstream performance. Notably, WSR-Tune achieves a strong trade-off even with relatively small values of ρ.

Computational overhead. WSR-Tune requires a one-time SVD to construct the safety-conditioned basis, which is then fixed and reused throughout downstream fine-tuning. Thus, the basis construction introduces only a one-time preprocessing cost, without requiring repeated decomposition or additional optimization during adaptation. We provide a detailed runtime analysis in Appendix B.

Table 2: Comparison across various chat and instruction-tuned models. Safety is measured by Jailbreak (JB) AVG (averaged over Direct, AutoDAN, PAIR, and PAP; lower is better), and downstream performance is measured by task accuracy (higher is better). Specifically, we use GSM8K for the Llama-2 Chat models and MATH for the Llama-3 Instruct models. We also report changes relative to Full Params FT: Δ_{S} (safety, lower is better) and Δ_{D} (downstream, higher is better). We further define $\Delta_{\mathrm{Overall}} = -\Delta_{\mathrm{S}} + \Delta_{\mathrm{D}}$ to summarize the safety–utility trade-off.

	Llama-2-7B-Chat					Llama-2-13B-Chat				
	Safety ↓		Downstream ↑		Overall ↑	Safety ↓		Downstream ↑		Overall ↑
Method	**JB AVG**	Δ_{S}	**Accuracy**	Δ_{D}	$\Delta_{\mathrm{Overall}}$	**JB AVG**	Δ_{S}	**Accuracy**	Δ_{D}	$\Delta_{\mathrm{Overall}}$
Full Params FT	20.78	0.00	41.17	0.00	0.00	10.04	0.00	45.94	0.00	0.00
SafeInstr	13.55	-7.23	38.44	-2.73	+4.50	5.82	-4.22	46.55	+0.61	+4.83
Resta	10.56	-10.22	36.92	-4.25	+5.97	10.34	+0.30	44.20	-1.74	-2.04
SEAL	12.75	-8.03	38.89	-2.28	+5.75	10.28	+0.24	45.19	-0.75	-0.99
AsFT	5.27	-15.51	19.71	-21.46	-5.95	0.86	-9.18	28.43	-17.51	-8.33
Lisa	9.95	-10.83	14.40	-26.77	-15.94	1.39	-8.65	36.24	-9.70	-1.05
SafeDelta	5.57	-15.21	19.79	-21.38	-6.17	0.97	-9.07	28.43	-17.51	-8.44
SN-Tune	27.18	+6.40	38.99	-2.18	-8.58	28.97	+18.93	48.22	+2.28	-16.65
RSN-Tune	20.73	-0.05	40.26	-0.91	-0.86	28.79	+18.75	49.96	+4.02	-14.73
WSR-Tune (Ours)	6.90	-13.88	38.99	-2.18	**+11.70**	1.35	-8.69	49.58	+3.64	**+12.33**

	Llama-3.2-3B-Instruct					Llama-3.1-8B-Instruct				
	Safety ↓		Downstream ↑		Overall ↑	Safety ↓		Downstream ↑		Overall ↑
Method	**JB AVG**	Δ_{S}	**Accuracy**	Δ_{D}	$\Delta_{\mathrm{Overall}}$	**JB AVG**	Δ_{S}	**Accuracy**	Δ_{D}	$\Delta_{\mathrm{Overall}}$
Full Params FT	8.65	0.00	21.52	0.00	0.00	9.28	0.00	12.12	0.00	0.00
SafeInstr	8.23	-0.42	22.26	+0.74	+1.16	6.21	-3.07	13.98	+1.86	+4.93
Resta	8.12	-0.53	20.56	-0.96	-0.43	6.74	-2.54	10.56	-1.56	+0.98
SEAL	8.37	-0.28	21.62	+0.10	+0.38	5.45	-3.83	13.18	+1.06	+4.89
AsFT	7.11	-1.54	7.76	-13.76	-12.22	3.36	-5.92	5.66	-6.46	-0.54
Lisa	6.83	-1.82	8.82	-12.70	-10.88	3.52	-5.76	11.32	-0.80	+4.96
SafeDelta	6.48	-2.17	6.48	-15.04	-12.87	4.55	-4.73	1.18	-10.94	-6.21
SN-Tune	21.84	+13.19	19.36	-2.16	-15.35	54.35	+45.07	13.62	+1.50	-43.57
RSN-Tune	24.49	+15.84	19.62	-1.90	-17.74	55.34	+46.06	14.10	+1.98	-44.08
WSR-Tune (Ours)	6.40	-2.25	22.38	+0.86	**+3.11**	5.62	-3.66	13.70	+1.58	**+5.24**

4 Experiments

4.1 Experimental Setup & Details

Datasets and models. Unless otherwise specified, we use the Circuit Breakers dataset [45] as the safety dataset, consisting of harmful prompts with safe refusal responses. To evaluate sensitivity to the safety-data source, we conduct additional experiments using safety data selected from BeaverTails [46] in Appendix F.3. For downstream adaptation, we use GSM8K [47], Hendrycks MATH [48], MedQA [49], and ARC-Challenge [50]. We additionally evaluate AGNEWS [51] and report the results in Appendix F.5, extending the downstream evaluation beyond reasoning tasks. We conduct experiments on a diverse set of large language models spanning the Llama [52, 53], Qwen [54], and Gemma [55] families, where the exact configurations are provided in Appendix A.

Baselines. We compare WSR-Tune with representative safety-preserving fine-tuning baselines; SafeInstr [21] and SEAL [22], which use safety supervision or select safety-compatible data; Resta [29] and SafeDelta [28], which modify fine-tuned weights post hoc; and SN-Tune and RSN-Tune [23], which constrain updates using safety-neuron importance. We also consider AsFT [24], which regularizes updates using alignment directions, and Lisa [8], which alternates safety and task optimization with a proximal penalty.

Evaluation metrics. We evaluate downstream performance using 5-shot accuracy on each downstream dataset. For safety evaluation, we use harmful prompts from AdvBench [13] and report attack success rate (ASR). We consider one direct prompting setting using the original AdvBench prompts, and three jailbreak attack methods, AutoDAN [14], PAIR [17], and PAP [18], which generate attack prompts. We generate jailbreak prompts using the HarmBench [56] and measure ASR with its refusal keywords, counting an attack as successful if the response does not begin with these keywords. We additionally evaluate the same responses using WildGuard [57], an LLM-based judge, in Appendix H.

Implementation details. Unless otherwise specified, all methods follow the same downstream fine-tuning pipeline and are trained for 3 epochs, with method-specific learning rates chosen to

Table 3: Compatibility of WSR-Tune with existing safety-preserving fine-tuning methods on Llama2-Chat models. Lower is better for safety metrics, and higher is better for downstream performance. Values are reported as percentages without the percent sign.

Llama-2-7B-Chat	**Safety ↓**					**Downstream ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**AVG**	**Accuracy**
SN-Tune	0.00	17.69	45.58	45.46	27.18	38.99
SN-Tune + WSR-Tune	**0.00**	**11.92**	**41.92**	**38.46**	**23.07**	**41.02**
SafeInstr	0.00	0.38	18.85	34.96	13.55	38.44
SafeInstr + WSR-Tune	**0.00**	**0.00**	**8.46**	**29.69**	**9.53**	**39.27**
SEAL	0.00	4.23	16.92	29.85	12.75	38.89
SEAL + WSR-Tune	**0.00**	**0.00**	**15.38**	**23.46**	**9.71**	**39.13**

Llama-2-13B-Chat	**Safety ↓**					**Downstream ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**AVG**	**Accuracy**
SN-Tune	**0**	24.42	**33.08**	**58.38**	28.97	48.22
SN-Tune + WSR-Tune	**0**	**7.12**	33.27	**58.38**	**24.69**	**48.75**
SafeInstr	**0**	**0**	9.62	13.65	5.82	46.55
SafeInstr + WSR-Tune	**0**	**0**	**6.54**	**5.19**	**2.93**	**48.60**
SEAL	**0**	0.96	15.19	24.96	10.28	45.19
SEAL + WSR-Tune	**0**	**0**	**2.69**	**1.31**	**1.00**	**48.67**

balance downstream performance and safety retention. Shared training settings and method-specific hyperparameters are provided in Appendix A.1.

4.2 Main Results

Comparison with baselines. We first compare WSR-Tune with representative safety-preserving fine-tuning baselines under downstream adaptation. Table 2 reports the main results across chat models. Overall, WSR-Tune achieves the best trade-off between safety and downstream performance across all model settings. WSR-Tune substantially reduces jailbreak attack success while maintaining competitive or improved downstream performance, leading to the largest overall gains over full-parameter fine-tuning. Figure 3 further visualizes these trade-off. Compared with existing baselines, WSR-Tune is positioned closer to the upper-right region, indicating higher downstream accuracy and stronger defense success rate, achieving a favorable balance between safety retention and downstream adaptation. These results indicate that reparameterizing the weight space enables more effective safety preservation than directly constraining updates in the original parameter space. WSR-Tune also improves the safety-utility trade-off over their respective unconstrained fine-tuning baselines on both models.

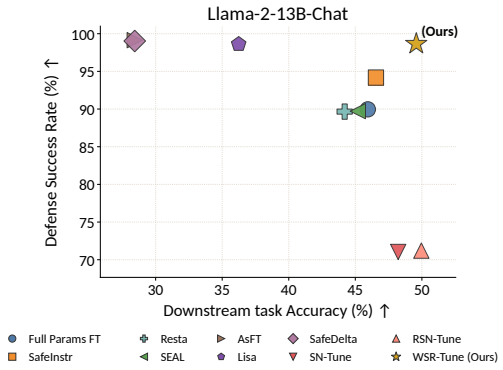

Figure 3: Safety-downstream trade-off among fine-tuning on Llama-2-13B-Chat. The x-axis shows downstream accuracy, and the y-axis shows defense success rate ($1-$ ASR), where higher indicates better safety.

Compatibility with existing fine-tuning methods. A useful property of WSR-Tune is that it can be combined with existing safety-preserving fine-tuning strategies rather than replacing them entirely. Table 3 shows the performance changes after applying WSR-Tune on top of existing baselines. Across SN-Tune, SafeInstr, and SEAL, adding WSR-Tune reduces the average attack success rate while also improving downstream performance. These results suggest that the benefits of reparameterization are complementary to existing parameter-selection and safety-data-based strategies, allowing WSR-Tune to serve as a plug-in mechanism for improving the safety-downstream trade-off of prior methods.

Table 4: Additional results on Qwen-2.5-7B-Instruct and Gemma-2-9B-IT. Both models are fine-tuned on GSM8K as the downstream task. The table follows the same delta-based format as Table 2, all deltas are relative to Full Params FT.

	Qwen-2.5-7B-Instruct					Gemma-2-9B-IT				
	Safety ↓		Downstream ↑		Overall	Safety ↓		Downstream ↑		Overall
Method	**JB AVG**	Δ_S	**Accuracy**	Δ_D	$\Delta_{Overall}$	**JB AVG**	Δ_S	**Accuracy**	Δ_D	$\Delta_{Overall}$
Full Params FT	3.62	0.00	67.32	0.00	0.00	5.37	0.00	69.75	0.00	0.00
SafeInstr	6.66	+3.04	67.70	+0.38	-2.66	8.47	+3.10	70.89	+1.14	-1.96
Resta	1.82	-1.80	66.87	-0.45	+1.35	4.58	-0.79	57.85	-11.90	-11.11
SEAL	4.34	+0.72	70.05	+2.73	+2.01	5.19	-0.18	68.31	-1.44	-1.26
AsFT	3.72	+0.10	71.57	+4.25	+4.15	4.77	-0.60	36.62	-33.13	-32.53
Lisa	4.16	+0.54	70.89	+3.57	+3.03	3.03	-2.34	65.28	-4.47	-2.13
SafeDelta	3.82	+0.20	66.87	-0.45	-0.65	4.52	-0.85	64.37	-5.38	-4.53
SN-Tune	37.55	+33.93	69.52	+2.20	-31.73	36.87	+31.50	74.37	+4.62	-26.88
RSN-Tune	38.31	+34.69	71.34	+4.02	-30.67	41.15	+35.78	71.57	+1.82	-33.96
WSR-Tune (Ours)	2.68	-0.94	70.81	+3.49	**+4.43**	4.99	-0.38	70.81	+1.06	**+1.44**

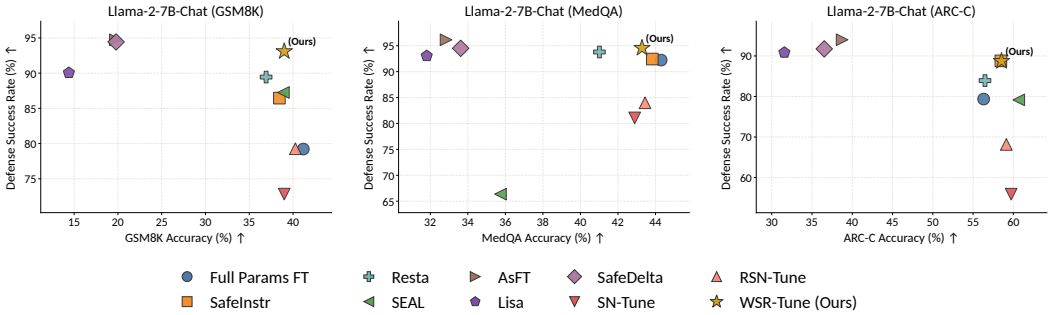

Figure 4: Comparison of safety-preserving downstream among fine-tuning methods on Llama-2-7B-Chat with GSM8K, ARC-C and MedQA.

4.3 Additional Analyses

Applications to other models. We further evaluate WSR-Tune on Qwen-2.5-7B-Instruct and Gemma-2-9B-IT, as shown in Table 4. Both methods preserve safety while maintaining strong downstream performance, suggesting that the benefits of weight-space reparameterization generalize across diverse instruction-tuned model families beyond Llama architectures.

Varying downstream datasets. To assess whether the benefits of WSR-Tune extend beyond mathematical reasoning, we additionally evaluate it on MedQA and ARC-C (Figure 4). These tasks cover diverse settings, including medical question answering and scientific knowledge reasoning. Across all settings, WSR-Tune preserves stronger safety than standard fine-tuning while remaining competitive in downstream performance, indicating that its advantages generalize beyond a single task family.

Results on base models. We also evaluate whether the same trend holds when downstream adaptation starts from base models rather than chat or instruction-tuned models. The detailed results are provided in the Appendix F.2, where WSR-Tune consistently achieves the best performance. This suggests that the advantage of weight-space reparameterization is not limited to a specific alignment regime, but also extends to base-to-safe-to-downstream adaptation pipelines.

Ablation study. Table 5 provides an ablation study on Llama-2-7B-Chat, designed to isolate the effect of each component in our method. Full-parameter fine-tuning severely degrades safety, while simply freezing important parameters in the original space improves safety only partially. Applying an importance mask in the original parameter space improves safety by freezing parameters with high safety importance, but the average ASR is still close to 10%. The full WSR-Tune method, which applies safety-aware masking in the reparameterized space, yields the best overall balance, confirming that both components are important for the final performance gain.

Impact of freeze ratio. Table 6 analyzes the effect of the freeze ratio using Llama-2-7B-Chat. Increasing the freeze ratio to 20% freezes more safety-related coefficients, improving safety preservation while reducing the capacity for downstream adaptation. Overall, WSR-Tune remains robust

Table 5: Ablation study on Llama-2-7B-Chat. Lower JB AVG indicates stronger safety, and higher Accuracy indicates better downstream performance.

Llama-2-7B-Chat		Safety ↓		Downstream ↑		Overall ↑
Category	**Method**	**JB AVG**	Δ_S	**Accuracy**	Δ_D	**Overall**
Ablation	Full parameter FT	20.78	0.00	41.17	0.00	0.00
	Original space + important mask FT	9.17	-11.61	40.41	-0.76	+10.85
	WSR-Tune	6.90	-13.88	38.99	-2.18	**+11.70**

across a moderate range of freeze ratios, maintaining low average ASR and competitive downstream performance. Results for freeze ratios up to 50% are provided in Appendix F.4.

Table 6: Effect of the freeze ratio in WSR-Tune.

	Safety ↓					Downstream ↑
Freeze ratio	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**AVG**	**Accuracy**
1%	0.00	0.00	10.38	16.88	6.82	38.82
5%	0.00	0.00	9.04	16.42	6.37	39.04
10%	0.00	0.00	9.62	18.00	6.90	38.99
15%	0.00	0.00	9.04	18.00	6.76	39.88
20%	0.00	0.00	8.08	14.27	5.59	38.36

Other results. Appendix B reports runtime and memory measurements, showing that SVD-based basis construction incurs a one-time cost relative to downstream training. Appendix C examines sensitivity to the output-side basis V_l. Appendix F further reports results on Llama-3 models (Appendix F.1), adaptation from the Llama-3.1-8B base model (Appendix F.2), robustness when BeaverTails is used as the safety dataset and when different data fractions are used for basis construction and importance masking (Appendix F.3), freeze-ratio results up to 50% (Appendix F.4), evaluation on AGNEWS (Appendix F.5), and complete attack-wise results for the main experiments (Appendix F.6). Appendix H shows that WSR-Tune's favorable safety–utility trade-off persists under an LLM-based judge (WildGuard).

5 Conclusion

In this work, we revisited safety-preserving fine-tuning in LLMs and showed that existing approaches are fundamentally constrained by operating in the original parameter space, where safety information is distributed across directions. We proposed WSR-Tune, a weight-space reparameterization framework that constructs a safety-conditioned basis, enabling downstream adaptation in a complementary subspace. By concentrating safety-relevant information into a small set of coefficients, WSR-Tune allows effective preservation of safety through simple freezing while maintaining strong downstream performance. While prior work highlights the entanglement of safety- and task-related information in the original parameter space, our findings suggest that changing parameter coordinates can enable more targeted protection of safety-relevant components. Together, these results suggest that safety preservation depends not only on which parameters are updated, but also on the coordinates in which they are protected, highlighting weight-space reparameterization as a promising direction for balancing safety preservation and downstream task adaptation.

Acknowledgments

This work was supported in part by the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT), the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) in 2026 (Grant No. RS-2026-25612350, Project Title: Development of an End-to-End Design and Validation Framework for Mathematically Principled Next-Generation AI Architectures and Learning Algorithms), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00457882, AI Research Hub Project), and the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. RS-2026-25470983). Dong-Jun Han is the corresponding author.

References

[1] Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, et al. Position: Trustllm: Trustworthiness in large language models. In *International Conference on Machine Learning*, pages 20166–20270. PMLR, 2024.

[2] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. BloombergGPT: A large language model for finance, 2023.

[3] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. ChatGPT for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 2023.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Shangbin Feng, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair NLP models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11737–11762, 2023.

[7] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[8] Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Lisa: Lazy safety alignment for large language models against harmful fine-tuning attack. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[9] Weichen Yu, Kai Hu, Tianyu Pang, Chao Du, Min Lin, and Matt Fredrikson. Infecting LLM agents via generalizable adversarial attack. In *Red Teaming GenAI: What Can We Learn from Adversaries?*, 2025.

[10] Ang Li, Yin Zhou, Vethavikashini Chithrra Raghuram, Tom Goldstein, and Micah Goldblum. Commercial llm agents are already vulnerable to simple yet dangerous attacks, 2025.

[11] Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to LLM-based agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[12] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. Jailbreaking ChatGPT via prompt engineering: An empirical study, 2024.

[13] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

[14] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[15] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Xianjun Yang, Xiao Wang, Qi Zhang, Linda Ruth Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models, 2024.

[17] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2024.

[18] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How Johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14322–14350, 2024.

[19] James Beetham, Souradip Chakraborty, Mengdi Wang, Furong Huang, Amrit Singh Bedi, and Mubarak Shah. Jailbreaks as inference-time alignment: A framework for understanding safety failures in LLMs. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7689–7713, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[20] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[21] Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Rottger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned LLaMAs: Lessons from improving the safety of large language models that follow instructions. In *The Twelfth International Conference on Learning Representations*, 2024.

[22] Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. SEAL: Safety-enhanced aligned LLM fine-tuning via bilevel data selection. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Yiran Zhao, Wenxuan Zhang, Yuxi Xie, Anirudh Goyal, Kenji Kawaguchi, and Michael Shieh. Understanding and enhancing safety mechanisms of LLMs via safety-specific neuron. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Shuo Yang et al. AsFT: Anchoring safety during LLM fine-tuning within narrow safety basin. *arXiv preprint arXiv:2506.08473*, 2025.

[25] Biao Yi, Jiahao Li, Baolei Zhang, Lihai Nie, Tong Li, Tiansheng Huang, and Zheli Liu. Safegrad: Gradient surgery for safe llm fine-tuning. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3326–3330. IEEE, 2026.

[26] Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when finetuning large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[27] Mingjie Li et al. SaLoRA: Safety-alignment preserved low-rank adaptation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Ning Lu, Shengcai Liu, Jiahao Wu, Weiyu Chen, Zhirui Zhang, Yew-Soon Ong, Qi Wang, and Ke Tang. Safe Delta: Consistently preserving safety when fine-tuning LLMs on diverse datasets. In *Forty-second International Conference on Machine Learning*, 2025.

[29] Rishabh Bhardwaj, Duc Anh Do, and Soujanya Poria. Language models are Homer Simpson! safety re-alignment of fine-tuned language models through task arithmetic. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14138–14149, 2024.

[30] Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Joshua Kimball, and Ling Liu. Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning attack. In *Forty-second International Conference on Machine Learning*, 2025.

[31] Shuhao Chen, Weisen Jiang, Yeqi Gong, Shengda Luo, Chengxiang Zhuo, Zang Li, James T Kwok, and Yu Zhang. Spard: Defending harmful fine-tuning attack via safety projection with relevance-diversity data selection. *arXiv preprint arXiv:2605.28030*, 2026.

[32] Jiawen Zhang, Yangfan Hu, Kejia Chen, Lipeng He, Jiachen Ma, Jian Lou, Dan Li, Jian Liu, Xiaohu Yang, and Ruoxi Jia. Understanding and preserving safety in fine-tuned llms. *arXiv preprint arXiv:2601.10141*, 2026.

[33] Bingjie Zhang, Yibo Yang, Ren Zhe, Dandan Guo, Jindong Gu, Philip Torr, and Bernard Ghanem. A guardrail for safety preservation: When safety-sensitive subspace meets harmful-resistant null-space. In *International Conference on Learning Representations*, volume 2026, pages 134322–134347, 2026.

[34] Dongxin Guo, Jikun Wu, and Siu Ming Yiu. Safeanchor: Preventing cumulative safety erosion in continual domain adaptation of large language models. *arXiv preprint arXiv:2604.17691*, 2026.

[35] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025.

[36] Tom Wollschläger, Jannes Elstner, Simon Geisler, Vincent Cohen-Addad, Stephan Günnemann, and Johannes Gasteiger. The geometry of refusal in large language models: Concept cones and representational independence. In *Forty-second International Conference on Machine Learning*, 2025.

[37] Wenbo Pan, Zhichao Liu, Qiguang Chen, Xiangyang Zhou, Yu Haining, and Xiaohua Jia. The hidden dimensions of LLM alignment: A multi-dimensional analysis of orthogonal safety directions. In *Forty-second International Conference on Machine Learning*, 2025.

[38] Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. In *International Conference on Machine Learning*, pages 52588–52610. PMLR, 2024.

[39] Kaustubh Ponkshe, Shaan Shah, Raghav Singhal, and Praneeth Vepakomma. Safety subspaces are not linearly distinct: A fine-tuning case study. In *The Fourteenth International Conference on Learning Representations*, 2026.

[40] Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to LLM security. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[42] Fei Wu, Jia Hu, Geyong Min, and Shiqiang Wang. Efficient orthogonal fine-tuning with principal subspace adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[43] Do-Yeon Kim, Dong-Jun Han, Jun Seo, and Jaekyun Moon. Warping the space: Weight space rotation for class-incremental few-shot learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[45] Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, J Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[46] Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704, 2023.

[47] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[48] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[49] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *arXiv preprint arXiv:2009.13081*, 2020.

[50] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[51] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[53] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[54] Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[55] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[56] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *Forty-first International Conference on Machine Learning*, 2024.

[57] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[58] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

[59] Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*, 2024.

[60] Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *The Thirteenth International Conference on Learning Representations*, 2025.

[61] Guozhi Liu, Weiwei Lin, Tiansheng Huang, Ruichao Mo, Qi Mu, Xiumin Wang, and Li Shen. Surgery: Mitigating harmful fine-tuning for large language models via attention sink. *arXiv preprint arXiv:2602.05228*, 2026.

[62] Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. *Advances in Neural Information Processing Systems*, 37:74058–74088, 2024.

[63] Guozhi Liu, Weiwei Lin, Qi Mu, Tiansheng Huang, Ruichao Mo, Yuren Tao, and Li Shen. Targeted vaccine: Safety alignment for large language models against harmful fine-tuning via layer-wise perturbation. *IEEE Transactions on Information Forensics and Security*, 2025.

[64] Domenic Rosati, Jan Wehner, Kai Williams, Ł ukasz Bartoszcze, David Atanasov, Robie Gonzales, Subhabrata Majumdar, Carsten Maple, Hassan Sajjad, and Frank Rudzicz. Representation noising: A defence mechanism against harmful finetuning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 12636–12676. Curran Associates, Inc., 2024.

[65] Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Tekin, and Ling Liu. Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation. In *International Conference on Learning Representations*, volume 2025, pages 67202–67226, 2025.

[66] Yuhui Wang, Rongyi Zhu, and Ting Wang. Self-destructive language models. In *International Conference on Learning Representations*, volume 2026, pages 15873–15899, 2026.

[67] Biao Yi, Tiansheng Huang, Baolei Zhang, Tong Li, Lihai Nie, Zheli Liu, and Li Shen. Ctrap: Embedding collapse trap to safeguard large language models from harmful fine-tuning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10013–10030, 2026.

[68] Zixuan Chen, Weikai Lu, Xin Lin, and Ziqian Zeng. Sdd: Self-degraded defense against malicious fine-tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29109–29125, 2025.

[69] Amber Yijia Zheng, Cedar Site Bai, Brian Bullins, and Raymond A Yeh. Model immunization from a condition number perspective. *arXiv preprint arXiv:2505.23760*, 2025.

[70] Domenic Rosati, Sebastian Dionicio, Xijie Zeng, Subhabrata Majumdar, Frank Rudzicz, and Hassan Sajjad. Locking open weight models with spectral deformation. In *ICML Workshop on Technical AI Governance (TAIG)*, 2025.

A Experimental Setup

All experiments are conducted on a server equipped with a single NVIDIA B200 GPU with 180GB of memory, AMD EPYC 9365 36-core processor $\times$ 2 (72 cores total), and 2.2TB of system memory. We use bf16 mixed-precision training with a maximum sequence length of 1024 tokens. For training, we use gradient accumulation to obtain an effective batch size of 16.

A.1 Training Hyperparameters

Shared settings. Downstream training uses 3 epochs, an effective batch size of 16, a maximum sequence length of 1024, bf16 precision, and random seed 42. We use AdamW with cosine learning-rate decay and a gradient-clipping norm of 1.0.

Full-parameter adaptation. Safety tuning uses a learning rate of 5×10^{-5} for the Llama and Qwen models and 3×10^{-5} for Gemma-2-9B-IT. Full-parameter downstream training uses a learning rate of 5×10^{-5}, weight decay of 0.01, and a warmup ratio of 0.1. For Gemma-2-9B-IT, the downstream learning rate is 1×10^{-5}. SN-Tune, RSN-Tune, and WSR-Tune target the attention projections `q`, `k`, and `v` and the MLP projections `up` and `down` in all transformer layers.

LoRA-based adaptation. Vanilla LoRA FT, AsFT, Lisa, SafeLoRA, SaLoRA, and WSR-LoRA use rank $r = 16$, scaling parameter $\alpha = 16$ (i.e., $\alpha/r = 1$), and dropout 0.05. Adapters are applied to the same five projections (`q`, `k`, `v`, `up`, and `down`) in all transformer layers. For GSM8K, MATH, MedQA, and ARC-C, all six methods use a learning rate of 3×10^{-4}, weight decay of 0, and a warmup ratio of 0.03. The same settings are used for the BeaverTails-based comparison.

Method-specific settings. Table 7 lists the additional settings for each method. WSR-Tune uses a default freeze ratio of 10%, whereas the reported WSR-LoRA comparisons use 30%. WSR-LoRA uses function-preserving SVD initialization and importance-based masks on both low-rank factors, as detailed in Appendix E.

Table 7: Method-specific settings used in the reported comparisons, unless explicitly varied in an ablation. Common optimization and adapter settings are described above.

Method	Additional settings
Full Params FT	No additional safety constraint.
SafeInstr	Safety examples equal to 10% of the downstream training-set size.
Resta	Safety-vector scaling coefficient $\gamma = 0.3$.
SafeDelta	Safety degradation constraint $s = 0.1$.
SEAL	Select the top 80% of downstream samples.
SN-Tune / RSN-Tune	Safety-neuron or critical-safety-neuron budget of at most 1% of model parameters.
WSR-Tune	Freeze ratio $\rho = 0.1$ in the reparameterized weight space.
Vanilla LoRA FT	No additional safety constraint.
AsFT	Alignment regularization coefficient $\lambda = 1.0$.
Lisa	Proximal coefficient $\rho_{\text{Lisa}} = 1.0$; alternate 100 safety-alignment steps and 900 downstream steps.
SafeLoRA	Cosine-similarity threshold $\tau = 0.3$ for selecting updates to project.
SaLoRA	Safety and task subspace ranks $r_s = r_t = 32$; task-specific initialization using 256 safety examples and 256 downstream examples.
WSR-LoRA	Freeze ratio $\rho = 0.3$; importance masks on both B and $\widetilde{A} = AU$.

B Runtime and Computational Overhead

We analyze the computational overhead of WSR-Tune. All measurements are conducted on the Llama-2-7B-Chat model with GSM8K as the downstream dataset, using a single NVIDIA B200 GPU.

Basis construction cost. In WSR-Tune, an additional preprocessing step is required to construct the safety-conditioned basis. This involves performing SVD on all target layers (32 layers, including attention projections `q`, `k`, `v` and MLP projections `up`, `down`), sharing a single SVD across projections with the same input activations. This step takes approximately 9 minutes 31 seconds and requires around 30 GB of GPU memory. Notably, this cost is incurred only once before downstream fine-tuning.

Table 8: Measured costs of WSR-Tune on Llama-2-7B-Chat with GSM8K using a single NVIDIA B200 GPU. Relative costs compare one-time basis construction with the three-epoch WSR-Tune downstream training stage.

Process	Time	VRAM Cost
Basis construction (one time)	9 min 31 s	30 GB
Downstream training (3 epochs)	39 min 8 s	96 GB
Basis construction / training	≈ 0.24	≈ 0.32

Training time comparison. For downstream adaptation, WSR-Tune requires 39 minutes 8 seconds for 3 epochs (approximately 13 minutes per epoch), with a peak GPU memory usage of 96 GB. As shown in Table 8, basis construction adds approximately 24% relative to this three-epoch fine-tuning time, while requiring approximately $0.32\times$ the GPU memory of the training stage. Also, WSR-LoRA reduces this cost: including the one-time basis construction, it requires approximately 0.32× the end-to-end wall-clock time and 0.55× the peak VRAM of WSR-Tune, and reusing the fixed basis lowers the run-specific time further to 0.18×.

Summary. Overall, WSR-Tune introduces additional computational overhead due to basis construction and reparameterized training. However, since the basis construction is a one-time cost, its relative cost decreases over longer training runs. The method remains practical while providing an improved trade-off between safety and downstream performance.

C Effect of the Output-Side Basis Choice

In the main experiments, we set the output-side basis to $V_l = I$ in Equation (9). To examine the sensitivity of WSR-Tune to this choice, we additionally consider two orthogonal alternatives while keeping all other settings unchanged: a signed permutation matrix V_{perm}, which permutes and sign-flips the output coordinates without mixing them, and a dense Haar-random orthogonal matrix V_{dense}, which fully mixes the output coordinates.

As shown in Table 9, all three choices of V_l yield similar safety–utility trade-offs and substantially improve over full-parameter fine-tuning. Although changing V_l can alter the coordinate-wise importance scores and the resulting mask, the overall performance varies only modestly.

This is consistent with the role of V_l and U_l in $\widetilde{W}_l = V_l^\top W_l U_l$: U_l is constructed from the safety activation covariance and therefore encodes safety-conditioned input directions, whereas V_l only changes the orthonormal coordinates of the output dimension without introducing additional safety information. We therefore use $V_l = I$ throughout the main experiments as the simplest canonical choice that avoids introducing an arbitrary output-side rotation.

Table 9: Effect of the output-side basis V on Llama-2-7B-Chat with GSM8K.

Method	JB AVG ↓	Δ_S ↓	GSM8K ↑	Δ_D ↑	Overall ↑
Full Params FT	20.78	0.00	41.17	0.00	0.00
WSR-Tune ($V = V_{\mathrm{perm}}$)	8.38	-12.40	39.95	-1.22	+11.18
WSR-Tune ($V = V_{\mathrm{dense}}$)	8.21	-12.57	39.65	-1.52	+11.05
WSR-Tune ($V = I$)	**6.90**	**-13.88**	38.99	-2.18	**+11.70**

Table 10: Overall protected activation-energy fractions under a 10% masking budget. Safety and Task denote P_{safe} and P_{task}, respectively; $R = P_{\text{safe}}/P_{\text{task}}$.

Mask space	Safety	Task	R
Original	0.321	0.287	1.12
WSR	**0.530**	0.359	**1.48**

Table 11: Projection-wise safety-to-task activation-energy ratios under a 10% masking budget. Each entry reports the mean per-layer R over 32 transformer layers.

Mask space	q	k	v	up	down
Original	1.08	1.08	1.08	1.08	1.26
WSR	**1.36**	**1.36**	**1.41**	**1.83**	**1.54**

D Mechanistic Effect of Reparameterized Protection

We examine how safety-conditioned reparameterization changes the effect of the same importance-based protection rule, complementing the ablation in Table 5.

Induced update constraints. With $V_l = I$, we have $\widetilde{W}_l = W_l U_l$. Let $S_{l,i} = \{j : M_l(i,j) = 1\}$ denote the protected basis indices for row i. Freezing $\widetilde{W}_l(i,j)$ imposes $e_i^\top \Delta W_l u_j = 0$, where e_i selects the i-th output row and u_j is the j-th column of U_l. Equivalently,

$$P_{l,i} = \sum_{j \in S_{l,i}} u_j u_j^\top, \qquad e_i^\top \Delta W_l P_{l,i} = 0. \tag{13}$$

Different rows can therefore preserve different subsets of the shared basis. Original-space masking is the special case $U_l = I$, which freezes individual weight entries rather than their linear combinations.

Two consequences. First, the protection budget is distributed differently. At $\rho = 10\%$, 99.9% of original weight coordinates participate in the induced constraints, and 98.2% of all coordinates are only partially constrained: they can still change while satisfying the coupled constraints. The reparameterized mask also mixes protected and trainable entries in 99.22% of rows and 95.74% of columns.

Second, since U_l diagonalizes $C_l = H_l H_l^\top$, the protected and complementary directions satisfy

$$P_{l,i} C_l (I - P_{l,i}) = 0. \tag{14}$$

Thus, these directions decouple under the empirical safety activation statistics used to construct the basis, a property that canonical-coordinate masking does not generally satisfy.

Held-out safety selectivity. We evaluate the masks on 256 BeaverTails safety examples and 256 GSM8K task examples, none used to construct the basis or masks. For each dataset D, we measure the protected activation-energy fraction P_D in each mask's own basis across 160 projections (`q`, `k`, `v`, `up`, and `down` over 32 layers), excluding padding tokens. We report $R = P_{\text{safe}}/P_{\text{task}}$, with overall energy fractions and projection-wise ratios shown separately in Table 10 and 11.

WSR increases the overall ratio from 1.12 to 1.48 and achieves higher ratios across all projection types. Here, $R = 1$ indicates equal relative coverage of safety and task activation energy. Together with Table 5, these results support that reparameterization changes what the importance criterion selects and how protection constrains updates, without implying complete separation of safety and task representations.

E Detailed Formulation of WSR-LoRA

We provide the detailed formulation of WSR-LoRA. For clarity, we omit the layer index and the standard LoRA [44] scaling factor; the same procedure is applied independently to each adapted linear projection.

Function-preserving low-rank initialization. Standard LoRA starts from a zero low-rank update, such that the pre-existing safety-aligned structure remains entirely in the frozen weight W_0. To expose a nonzero component of the aligned weight to the WSR protection mechanism, we adopt the function preserving low-rank initialization of PiSSA [58].

Let

$$W_0 = P\Lambda Q^\top \tag{15}$$

be the singular value decomposition of the safety-aligned weight. Using the leading r singular components, we initialize

$$B_0 = P_r\Lambda_r^{1/2}, \qquad A_0 = \Lambda_r^{1/2}Q_r^\top, \tag{16}$$

and define the frozen residual weight as

$$W_{\mathrm{res}} = W_0 - B_0A_0. \tag{17}$$

Therefore,

$$W_0 = W_{\mathrm{res}} + B_0A_0. \tag{18}$$

This initialization is used only to obtain a nonzero trainable factorization that exactly reproduces the original model at initialization; it does not assume that the selected singular components are themselves safety-specific. The safety-aware component of WSR-LoRA is introduced by the subsequent reparameterization and importance-based protection.

Safety-conditioned reparameterization. Let $U \in \mathbb{R}^{n\times n}$ denote the safety-conditioned input basis constructed as in Section 3.3. At downstream optimization step t, we reparameterize the input-side factor as

$$\widetilde{A}_t = A_tU, \qquad A_t = \widetilde{A}_tU^\top. \tag{19}$$

The adapted weight can then be written as

$$W_t = W_{\mathrm{res}} + B_t\widetilde{A}_tU^\top. \tag{20}$$

At initialization ($t = 0$), $\widetilde{A}_0 = A_0U$, and since $UU^\top = I$,

$$W_{\mathrm{res}} + B_0\widetilde{A}_0U^\top = W_{\mathrm{res}} + B_0A_0 = W_0. \tag{21}$$

Thus, WSR-LoRA preserves the original model function at initialization while expressing the trainable input-side factor in the same safety-conditioned coordinates used by WSR-Tune.

Importance estimation and protection. During downstream adaptation, W_{res} and U remain fixed, while B and $\widetilde{A}$ are trainable. Following the same importance criterion as WSR-Tune, we compute

$$G_B = \sum_{x\in D_{\mathrm{safe}}} \left|\frac{\partial\mathcal{L}(x)}{\partial B}\right|, \qquad G_{\widetilde{A}} = \sum_{x\in D_{\mathrm{safe}}} \left|\frac{\partial\mathcal{L}(x)}{\partial \widetilde{A}}\right|. \tag{22}$$

We construct binary masks M_B and $M_{\widetilde{A}}$ by selecting the top-ρ entries according to these scores. During downstream fine-tuning, the selected safety-critical entries of B and $\widetilde{A}$ are kept fixed at their initialized values, while only the remaining entries are updated.

Unlike full-parameter WSR-Tune, the effective low-rank update depends bilinearly on B and $\widetilde{A}$. Therefore, factor-wise protection does not impose the exact coefficient-wise constraints of WSR-Tune; rather, WSR-LoRA transfers the same safety-conditioned protection principle to a low-rank trainable parameterization.

F Additional Experimental Results

This appendix provides additional experimental results omitted from the main text due to space constraints. In the main paper, Table 2 reports average safety changes across models. Here, we provide the full breakdown for each model, including attack-specific safety results for Direct, AutoDAN, PAIR, and PAP, as well as downstream performance on each evaluated dataset.

F.1 Results on Llama-3 Models

Table 12 compares safety-preserving fine-tuning methods on Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct with MATH fine-tuning. WSR-Tune improves Overall by +3.11 and +5.24 points over Full Params FT, respectively, while WSR-LoRA improves Overall by +2.35 and +0.34 points over Vanilla LoRA FT. The full attack-wise breakdown is provided in Table 19.

Table 12: Results on Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct with MATH fine-tuning. Metrics follow Table 2; deltas in the upper block, including AsFT and Lisa, are relative to Full Params FT, while those in the lower block are relative to Vanilla LoRA FT. AsFT and Lisa use LoRA adapters in these experiments. Lower JB AVG and higher MATH accuracy and Overall are better.

	Llama-3.2-3B-Instruct					Llama-3.1-8B-Instruct				
	Safety ↓		Downstream ↑		Overall ↑	Safety ↓		Downstream ↑		Overall ↑
Method	**JB AVG**	Δ_S	**MATH**	Δ_D	$\Delta_{Overall}$	**JB AVG**	Δ_S	**MATH**	Δ_D	$\Delta_{Overall}$
Full Params FT	8.65	0.00	21.52	0.00	0.00	9.28	0.00	12.12	0.00	0.00
SafeInstr	8.23	-0.42	22.26	+0.74	+1.16	6.21	-3.07	13.98	+1.86	+4.93
Resta	8.12	-0.53	20.56	-0.96	-0.43	6.74	-2.54	10.56	-1.56	+0.98
SEAL	8.37	-0.28	21.62	+0.10	+0.38	5.45	-3.83	13.18	+1.06	+4.89
AsFT	7.11	-1.54	7.76	-13.76	-12.22	3.36	-5.92	5.66	-6.46	-0.54
Lisa	6.83	-1.82	8.82	-12.70	-10.88	3.52	-5.76	11.32	-0.80	+4.96
SafeDelta	6.48	-2.17	6.48	-15.04	-12.87	4.55	-4.73	1.18	-10.94	-6.21
SN-Tune	21.84	+13.19	19.36	-2.16	-15.35	54.35	+45.07	13.62	+1.50	-43.57
RSN-Tune	24.49	+15.84	19.62	-1.90	-17.74	55.34	+46.06	14.10	+1.98	-44.08
WSR-Tune (Ours)	6.40	-2.25	22.38	+0.86	**+3.11**	5.62	-3.66	13.70	+1.58	**+5.24**
Vanilla LoRA FT	8.23	0.00	24.80	0.00	0.00	5.82	0.00	25.30	0.00	0.00
SafeLoRA	6.26	-1.97	23.04	-1.76	+0.21	3.94	-1.88	22.04	-3.26	-1.38
SaLoRA	9.35	+1.12	22.38	-2.42	-3.54	6.56	+0.74	23.20	-2.10	-2.84
WSR-LoRA (Ours)	6.76	-1.47	25.68	+0.88	**+2.35**	4.18	-1.64	24.00	-1.30	**+0.34**

F.2 Results on Llama-3.1-8B Base

Table 13 reports full-parameter and PEFT results when downstream adaptation starts from Llama-3.1-8B Base. Deltas are computed relative to Full Params FT for full-parameter methods and Vanilla LoRA FT for PEFT methods. WSR-Tune and WSR-LoRA achieve the best overall trade-offs in their respective adaptation regimes, showing that weight-space reparameterization generalizes to base-to-safe-to-downstream adaptation.

F.3 Robustness to Safety-Data Source and Quantity

We further examine whether WSR-Tune depends strongly on the source or amount of offline safety data. In addition to the default Circuit Breakers (CB) dataset, we construct an alternative BeaverTails-based (BV) safety dataset. We screen the BeaverTails source using Llama-Guard-3-8B and match the number of safety examples to Circuit Breakers for a controlled comparison. As in the main experiments, the safety examples used by WSR consist of harmful prompts with safe refusal responses.

We evaluate three aspects of safety-data sensitivity: (i) replacing Circuit Breakers with BeaverTails as the safety source, (ii) reducing the amount of data used for basis construction or importance estimation, and (iii) using different sources for basis construction and importance estimation. All experiments use Llama-2-7B-Chat with GSM8K downstream fine-tuning.

Alternative safety-data source. Table 14 reports results when the WSR pipeline is constructed using the BeaverTails-based safety source. For full-parameter methods, Δ_S and Δ_D are computed relative to Full Params FT; for PEFT methods, they are computed relative to Vanilla LoRA FT.

Table 13: Results on Llama-3.1-8B Base with GSM8K fine-tuning. Safety is measured by ASR under Direct, AutoDAN, PAIR, and PAP, with JB AVG denoting their mean. Raw scores are percentages and deltas are percentage-point differences. Deltas in the upper comparison block, including AsFT and Lisa, are relative to Full Params FT; those in the lower comparison block are relative to Vanilla LoRA FT. AsFT and Lisa use LoRA adapters. We use $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. Green cells highlight the Overall changes of WSR methods.

	Safety ↓						Downstream ↑		Overall ↑
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	Δ_{S}	**GSM8K**	Δ_{D}	Δ_{Overall}
Full Params FT	0.00	0.19	9.04	26.00	8.81	0.00	57.32	0.00	0.00
SafeInstr	0.00	0.00	2.88	23.35	6.56	-2.25	56.94	-0.38	+1.87
Resta	0.00	0.00	7.69	25.00	8.17	-0.64	48.45	-8.87	-8.23
SEAL	0.00	0.19	8.08	25.00	8.32	-0.49	55.95	-1.37	-0.88
AsFT	0.00	0.00	0.96	15.62	4.14	-4.67	27.14	-30.18	-25.51
Lisa	0.00	0.00	0.19	18.73	4.73	-4.08	30.25	-27.07	-22.99
SafeDelta	0.00	0.00	0.77	15.19	3.99	-4.82	41.32	-16.00	-11.18
SN-Tune	9.04	20.96	48.27	69.12	36.85	+28.04	63.00	+5.68	-22.36
RSN-Tune	23.27	12.12	40.38	38.38	28.54	+19.73	63.53	+6.21	-13.52
WSR-Tune (Ours)	0.00	0.00	1.35	16.50	4.46	-4.35	55.12	-2.20	**+2.15**
Vanilla LoRA FT	1.15	81.73	38.85	68.12	47.46	0.00	57.24	0.00	0.00
SafeLoRA	0.00	0.38	7.31	33.58	10.32	-37.14	48.37	-8.87	+28.27
SaLoRA	1.54	93.27	28.27	70.69	48.44	+0.98	56.41	-0.83	-1.81
WSR-LoRA (Ours)	0.00	5.96	8.08	27.50	10.38	-37.08	59.29	+2.05	**+39.13**

Table 14: Results using the BeaverTails-based safety source on Llama-2-7B-Chat with GSM8K. Safety is measured by ASR under Direct, AutoDAN, PAIR, and PAP, with JB AVG denoting their mean. Raw scores are percentages and deltas are percentage-point differences. Deltas in the upper comparison block, including AsFT and Lisa, are relative to Full Params FT; those in the lower comparison block are relative to Vanilla LoRA FT. AsFT and Lisa use LoRA adapters. We use $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. Green cells highlight the Overall changes of WSR methods.

	Safety ↓						Downstream ↑		Overall ↑
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	Δ_{S}	**GSM8K**	Δ_{D}	Δ_{Overall}
Full Params FT	0.19	6.73	34.04	35.69	19.16	0.00	41.85	0.00	0.00
SafeInstr	0.00	2.88	19.42	13.38	8.92	-10.24	40.33	-1.52	+8.72
Resta	0.19	4.62	36.54	32.88	18.56	-0.60	39.35	-2.50	-1.90
SEAL	0.19	5.58	27.50	26.08	14.84	-4.32	39.04	-2.81	+1.51
AsFT	0.38	26.92	26.35	25.85	19.87	+0.71	21.15	-20.70	-21.41
Lisa	0.00	0.77	16.15	15.19	8.03	-11.13	18.73	-23.12	-11.99
SafeDelta	0.00	0.00	10.19	20.08	7.57	-11.59	24.49	-17.36	-5.77
SN-Tune	0.19	13.27	29.62	43.08	21.54	+2.38	41.09	-0.76	-3.14
RSN-Tune	0.19	13.65	35.00	49.73	24.64	+5.48	41.24	-0.61	-6.09
WSR-Tune (Ours)	0.00	0.19	13.27	15.27	7.18	-11.98	41.09	-0.76	**+11.22**
Vanilla LoRA FT	0.19	0.96	21.92	32.38	13.86	0.00	36.62	0.00	0.00
SafeLoRA	0.19	0.38	20.19	27.31	12.02	-1.84	37.53	+0.91	+2.75
SaLoRA	0.19	18.27	36.15	44.12	24.68	+10.82	36.16	-0.46	-11.28
WSR-LoRA (Ours)	0.19	0.58	18.65	22.65	10.52	-3.34	36.92	+0.30	**+3.64**

WSR-Tune achieves the strongest overall safety–utility trade-off among the evaluated full-parameter methods, reducing JB AVG from 19.16 to 7.18 while retaining GSM8K accuracy of 41.09. In the PEFT setting, WSR-LoRA reduces JB AVG from 13.86 to 10.52 while slightly improving downstream accuracy over Vanilla LoRA, yielding the largest Overall improvement among the evaluated PEFT methods. These results show that the benefit of WSR is not specific to Circuit Breakers.

Sensitivity to basis and importance data. WSR uses the offline safety data for two distinct purposes: constructing the safety-conditioned activation basis and estimating coefficient importance. Table 15 separates these effects by varying the source and quantity of the two datasets independently. Unless otherwise specified, the full same-source safety dataset is used. "Cross" indicates that the basis is constructed from one safety source while importance is estimated using the other source.

The importance estimation is robust to substantial reductions in data. With the Circuit Breakers basis fixed, reducing the importance data to 50% or 10% yields Overall scores of +10.81 and +12.55, respectively. Similarly, with the BeaverTails basis, using only 50% or 10% of the importance data

Table 15: Sensitivity of WSR-Tune to the source and quantity of safety data on Llama-2-7B-Chat with GSM8K. CB and BV denote Circuit Breakers and BeaverTails, respectively. Percentages indicate the fraction of the corresponding safety source used for basis construction or importance estimation. Deltas are computed relative to Full Params FT under the corresponding safety-source setting.

Basis data	Importance data	JB AVG ↓	GSM8K ↑	Δ_S ↓	Δ_D ↑	Overall ↑
Circuit Breakers pipeline						
CB 100%	CB 100%	6.90	38.99	-13.88	-2.18	+11.70
CB 100%	CB 50%	9.09	40.29	-11.69	-0.88	+10.81
CB 100%	CB 10%	7.01	39.95	-13.77	-1.22	+12.55
CB 100%	BV 100%	8.81	41.02	-11.97	-0.15	+11.82
CB 50%	CB 100%	8.79	40.86	-11.99	-0.31	+11.68
CB 25%	CB 100%	6.08	40.94	-14.70	-0.23	+14.47
BeaverTails pipeline						
BV 100%	BV 100%	7.18	41.09	-11.98	-0.76	+11.22
BV 100%	BV 50%	7.88	40.86	-11.28	-0.99	+10.29
BV 100%	BV 10%	8.00	40.86	-11.16	-0.99	+10.17
BV 100%	CB 100%	7.52	40.86	-11.64	-0.99	+10.65
BV 50%	BV 100%	10.38	40.18	-8.78	-1.67	+7.11
BV 25%	BV 100%	9.23	38.21	-9.93	-3.64	+6.29

retains favorable trade-offs (+10.29 and +10.17). Cross-source importance estimation is also stable: using BeaverTails importance with a Circuit Breakers basis gives +11.82 Overall, while the reverse combination gives +10.65.

Basis construction is somewhat more sensitive to data reduction, particularly for BeaverTails. Nevertheless, even with only 25% of the BeaverTails basis data, WSR-Tune reduces JB AVG from 19.16 to 9.23. For Circuit Breakers, the method remains robust even with 25% of the basis data. Overall, these results indicate that WSR-Tune does not depend on a particular safety-data source and is especially robust to the source and quantity of the data used for importance estimation.

F.4 Sensitivity to the Freeze Ratio

Table 16 reports the full freeze-ratio ablation for WSR-Tune on Llama-2-7B-Chat with GSM8K fine-tuning. It extends the 1%–20% range shown in Table 6 to 50%, reporting attack-specific ASR, average ASR, and downstream accuracy.

WSR-Tune maintains low average ASR and competitive downstream accuracy across the 1%–20% range. Compared with the default 10% setting, freezing 40% or 50% of coefficients yields lower average ASR with reduced GSM8K accuracy. The trend is not strictly monotonic: the 30% setting yields both higher average ASR and higher downstream accuracy than the 10% setting.

Table 16: Full freeze-ratio ablation for WSR-Tune on Llama-2-7B-Chat with GSM8K fine-tuning. Safety metrics are attack success rates (ASR), and AVG is the mean over Direct, AutoDAN, PAIR, and PAP. All values are percentages. Lower ASR and higher GSM8K accuracy are better.

	Safety ↓					Downstream ↑
Freeze ratio	Direct	AutoDAN	PAIR	PAP	AVG	GSM8K
1%	0.00	0.00	10.38	16.88	6.82	38.82
5%	0.00	0.00	9.04	16.42	6.37	39.04
10%	0.00	0.00	9.62	18.00	6.90	38.99
15%	0.00	0.00	9.04	18.00	6.76	39.88
20%	0.00	0.00	8.08	14.27	5.59	38.36
30%	0.00	0.00	11.15	21.12	8.07	40.41
40%	0.00	0.00	5.58	15.62	5.30	38.36
50%	0.00	0.00	6.15	14.08	5.06	37.76

F.5 Results on AGNEWS

To evaluate downstream adaptation beyond reasoning tasks, we consider AGNEWS news classification on Llama-2-7B-Chat. Table 17 reports results for selected baselines, including attack-specific ASR, average ASR, and classification accuracy. We compare AsFT and Lisa with WSR-Tune using Full Params FT as the reference, and report the four LoRA-family methods in a separate block using Vanilla LoRA FT as the reference. These results show that WSR-Tune also preserves a favorable safety–utility trade-off on a classification task.

Table 17: AGNEWS results on Llama-2-7B-Chat for selected baselines. Safety is measured by ASR under Direct, AutoDAN, PAIR, and PAP, with JB AVG denoting their mean; downstream performance is AGNEWS accuracy. Raw scores are percentages and deltas are percentage-point differences. Deltas in the upper block, including AsFT and Lisa, are relative to Full Params FT; those in the lower block are relative to Vanilla LoRA FT. We use $\Delta_{\mathrm{Overall}} = -\Delta_{\mathrm{S}} + \Delta_{\mathrm{D}}$, as in Table 2.

	Safety ↓						Downstream ↑		Overall ↑
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\mathrm{S}}}$	**AGNEWS**	$\boldsymbol{\Delta_{\mathrm{D}}}$	$\boldsymbol{\Delta_{\mathrm{Overall}}}$
Full Params FT	0.00	0.00	25.77	32.04	14.45	0.00	94.00	0.00	0.00
Resta	0.00	0.00	14.62	17.54	8.04	-6.41	93.60	-0.40	+6.01
SEAL	0.00	0.00	12.69	14.46	6.79	-7.66	92.50	-1.50	+6.16
AsFT	0.00	0.00	5.58	10.73	4.08	-10.37	91.50	-2.50	+7.87
Lisa	0.00	0.00	6.92	11.65	4.64	-9.81	90.80	-3.20	+6.61
SafeDelta	0.00	0.00	8.65	23.23	7.97	-6.48	91.10	-2.90	+3.58
SN-Tune	0.96	55.96	66.15	89.54	53.15	+38.70	94.00	0.00	-38.70
RSN-Tune	0.96	49.42	65.00	88.62	51.00	+36.55	94.40	+0.40	-36.15
WSR-Tune (Ours)	0.00	0.00	9.04	11.88	5.23	-9.22	93.80	-0.20	**+9.02**
Vanilla LoRA FT	0.00	0.00	10.77	19.31	7.52	0.00	92.80	0.00	0.00
SafeLoRA	0.00	0.00	7.50	15.38	5.72	-1.80	83.20	-9.60	-7.80
SaLoRA	0.00	0.00	14.23	24.62	9.71	+2.19	92.70	-0.10	-2.29
WSR-LoRA (Ours)	0.00	0.00	10.19	21.73	7.98	+0.46	93.30	+0.50	**+0.04**

F.6 Detailed Results for Main Experiments

Table 18 provides the exact attack-wise results corresponding to Table 2 in the main text. While Table 2 reports the average jailbreak score and delta-based trade-off metrics, Table 18 reports the full values for Direct, AutoDAN, PAIR, and PAP, together with downstream accuracy for each model.

Table 20 provides the exact attack-wise results for the additional model-family experiments summarized in Table 4. Similarly, Table 21 reports the complete results for the downstream dataset experiments visualized in Figure 4, including both full-parameter and PEFT methods. These detailed tables show that the averaged improvements reported in the main text are consistent across individual attack settings and downstream datasets.

Table 18: Complete results on Llama-2 models with GSM8K fine-tuning. JB AVG denotes mean ASR over the four attacks. Within each model panel, deltas are relative to Full Params FT above the divider and Vanilla LoRA FT below it. AsFT and Lisa use LoRA adapters. Raw scores are percentages and deltas are percentage-point differences, with $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. Green cells highlight the Overall changes of WSR methods.

	Llama-2-7B-Chat								
	Safety ↓						Downstream ↑		Overall ↑
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.00	8.27	26.35	48.50	20.78	0.00	41.17	0.00	0.00
SafeInstr	0.00	0.38	18.85	34.96	13.55	-7.23	38.44	-2.73	+4.50
Resta	0.00	0.38	15.77	26.08	10.56	-10.22	36.92	-4.25	+5.97
SEAL	0.00	4.23	16.92	29.85	12.75	-8.03	38.89	-2.28	+5.75
AsFT	0.00	0.00	8.46	12.62	5.27	-15.51	19.71	-21.46	-5.95
Lisa	0.00	0.38	16.35	23.08	9.95	-10.83	14.40	-26.77	-15.94
SafeDelta	0.00	0.00	6.92	15.35	5.57	-15.21	19.79	-21.38	-6.17
SN-Tune	0.00	17.69	45.58	45.46	27.18	+6.40	38.99	-2.18	-8.58
RSN-Tune	0.19	5.96	38.85	37.92	20.73	-0.05	40.26	-0.91	-0.86
WSR-Tune (Ours)	0.00	0.00	9.62	18.00	6.90	-13.88	38.99	-2.18	**+11.70**
Vanilla LoRA FT	0.00	2.69	37.31	63.42	25.86	0.00	36.16	0.00	0.00
SafeLoRA	0.00	0.00	13.46	19.62	8.27	-17.59	31.08	-5.08	+12.51
SaLoRA	0.00	5.96	46.15	70.88	30.75	+4.89	38.74	+2.58	-2.31
WSR-LoRA (Ours)	0.00	0.19	12.50	17.00	7.42	-18.44	33.45	-2.71	**+15.73**

	Llama-2-13B-Chat								
	Safety ↓						Downstream ↑		Overall ↑
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.00	0.38	15.58	24.19	10.04	0.00	45.94	0.00	0.00
SafeInstr	0.00	0.00	9.62	13.65	5.82	-4.22	46.55	+0.61	+4.83
Resta	0.19	0.00	15.19	25.96	10.34	+0.30	44.20	-1.74	-2.04
SEAL	0.00	0.96	15.19	24.96	10.28	+0.24	45.19	-0.75	-0.99
AsFT	0.00	0.00	1.73	1.73	0.86	-9.18	28.43	-17.51	-8.33
Lisa	0.00	0.00	4.62	0.96	1.39	-8.65	36.24	-9.70	-1.05
SafeDelta	0.00	0.00	1.54	2.35	0.97	-9.07	28.43	-17.51	-8.44
SN-Tune	0.00	24.42	33.08	58.38	28.97	+18.93	48.22	+2.28	-16.65
RSN-Tune	0.00	17.88	37.88	59.38	28.79	+18.75	49.96	+4.02	-14.73
WSR-Tune (Ours)	0.00	0.00	3.46	1.92	1.35	-8.69	49.58	+3.64	**+12.33**
Vanilla LoRA FT	0.00	0.96	31.35	54.04	21.59	0.00	48.60	0.00	0.00
SafeLoRA	0.00	0.00	10.58	3.96	3.64	-17.95	42.30	-6.30	+11.65
SaLoRA	0.00	0.96	17.88	52.46	17.82	-3.77	38.44	-10.16	-6.39
WSR-LoRA (Ours)	0.00	0.00	7.88	4.12	3.00	-18.59	44.81	-3.79	**+14.80**

Table 19: Complete results on Llama-3 models with MATH fine-tuning. JB AVG denotes mean ASR over the four attacks. Within each model panel, deltas are relative to Full Params FT above the divider and Vanilla LoRA FT below it. AsFT and Lisa use LoRA adapters. Raw scores are percentages and deltas are percentage-point differences, with $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. Green cells highlight the Overall changes of WSR methods.

	Llama-3.2-3B-Instruct								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**MATH**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.19	0.00	5.77	28.62	8.65	0.00	21.52	0.00	0.00
SafeInstr	0.19	0.00	6.54	26.19	8.23	-0.42	22.26	+0.74	+1.16
Resta	0.19	0.00	7.31	24.96	8.12	-0.53	20.56	-0.96	-0.43
SEAL	0.00	0.00	7.31	26.15	8.37	-0.28	21.62	+0.10	+0.38
AsFT	0.00	0.00	5.19	23.23	7.11	-1.54	7.76	-13.76	-12.22
Lisa	0.00	0.00	5.96	21.35	6.83	-1.82	8.82	-12.70	-10.88
SafeDelta	0.00	0.00	5.19	20.73	6.48	-2.17	6.48	-15.04	-12.87
SN-Tune	0.38	0.00	18.46	68.50	21.84	+13.19	19.36	-2.16	-15.35
RSN-Tune	0.96	0.00	26.35	70.65	24.49	+15.84	19.62	-1.90	-17.74
WSR-Tune (Ours)	0.00	0.00	4.23	21.35	6.40	-2.25	22.38	+0.86	**+3.11**
Vanilla LoRA FT	0.00	0.00	7.12	25.81	8.23	0.00	24.80	0.00	0.00
SafeLoRA	0.00	0.00	6.15	18.88	6.26	-1.97	23.04	-1.76	+0.21
SaLoRA	0.19	0.00	7.50	29.69	9.35	+1.12	22.38	-2.42	-3.54
WSR-LoRA (Ours)	0.00	0.00	4.81	22.23	6.76	-1.47	25.68	+0.88	**+2.35**

	Llama-3.1-8B-Instruct								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**MATH**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.00	0.00	4.04	33.08	9.28	0.00	12.12	0.00	0.00
SafeInstr	0.00	0.00	1.35	23.50	6.21	-3.07	13.98	+1.86	+4.93
Resta	0.00	0.00	2.31	24.65	6.74	-2.54	10.56	-1.56	+0.98
SEAL	0.00	0.00	1.46	20.35	5.45	-3.83	13.18	+1.06	+4.89
AsFT	0.00	0.00	0.38	13.08	3.36	-5.92	5.66	-6.46	-0.54
Lisa	0.00	0.00	0.77	13.31	3.52	-5.76	11.32	-0.80	+4.96
SafeDelta	0.00	0.00	0.38	17.81	4.55	-4.73	1.18	-10.94	-6.21
SN-Tune	1.73	74.23	57.50	83.92	54.35	+45.07	13.62	+1.50	-43.57
RSN-Tune	6.54	61.35	68.85	84.62	55.34	+46.06	14.10	+1.98	-44.08
WSR-Tune (Ours)	0.00	0.00	1.15	21.31	5.62	-3.66	13.70	+1.58	**+5.24**
Vanilla LoRA FT	0.00	0.00	1.54	21.73	5.82	0.00	25.30	0.00	0.00
SafeLoRA	0.00	0.00	0.58	15.19	3.94	-1.88	22.04	-3.26	-1.38
SaLoRA	0.00	0.00	0.58	25.65	6.56	+0.74	23.20	-2.10	-2.84
WSR-LoRA (Ours)	0.00	0.00	0.58	16.15	4.18	-1.64	24.00	-1.30	**+0.34**

Table 20: Complete results on Qwen-2.5-7B-Instruct and Gemma-2-9B-IT with GSM8K fine-tuning. Safety is measured by ASR under Direct, AutoDAN, PAIR, and PAP, with JB AVG denoting their mean. Raw scores are percentages and deltas are percentage-point differences. Deltas in the upper comparison block, including AsFT and Lisa, are relative to Full Params FT; those in the lower comparison block are relative to Vanilla LoRA FT. AsFT and Lisa use LoRA adapters. We use $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. Green cells highlight the Overall changes of WSR methods.

	Qwen-2.5-7B-Instruct								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.00	0.00	2.31	12.15	3.62	0.00	67.32	0.00	0.00
SafeInstr	0.00	0.00	4.81	21.81	6.66	+3.04	67.70	+0.38	-2.66
Resta	0.00	0.00	0.96	6.31	1.82	-1.80	66.87	-0.45	+1.35
SEAL	0.00	0.00	3.85	13.50	4.34	+0.72	70.05	+2.73	+2.01
AsFT	0.00	0.00	2.12	12.77	3.72	+0.10	71.57	+4.25	+4.15
Lisa	0.00	0.00	2.88	13.77	4.16	+0.54	70.89	+3.57	+3.03
SafeDelta	0.00	0.00	1.73	13.54	3.82	+0.20	66.87	-0.45	-0.65
SN-Tune	1.15	10.96	53.08	85.00	37.55	+33.93	69.52	+2.20	-31.73
RSN-Tune	1.35	10.19	58.08	83.62	38.31	+34.69	71.34	+4.02	-30.67
WSR-Tune (Ours)	0.00	0.00	1.35	9.38	2.68	-0.94	70.81	+3.49	**+4.43**
Vanilla LoRA FT	0.00	0.00	3.85	8.58	3.11	0.00	71.19	0.00	0.00
SafeLoRA	0.00	0.00	2.69	10.26	3.24	+0.13	73.74	+2.55	+2.42
SaLoRA	0.00	0.00	5.19	12.42	4.40	+1.29	70.66	-0.53	-1.82
WSR-LoRA (Ours)	0.00	0.00	2.12	9.88	3.00	-0.11	73.54	+2.35	**+2.46**

	Gemma-2-9B-IT								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\text{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\text{D}}}$	$\boldsymbol{\Delta_{\text{Overall}}}$
Full Params FT	0.00	0.38	0.96	20.15	5.37	0.00	69.75	0.00	0.00
SafeInstr	0.00	2.12	3.46	28.31	8.47	+3.10	70.89	+1.14	-1.96
Resta	0.00	0.00	1.35	16.96	4.58	-0.79	57.85	-11.90	-11.11
SEAL	0.00	0.38	0.77	19.61	5.19	-0.18	68.31	-1.44	-1.26
AsFT	0.00	0.00	0.77	18.31	4.77	-0.60	36.62	-33.13	-32.53
Lisa	0.00	0.00	0.19	11.92	3.03	-2.34	65.28	-4.47	-2.13
SafeDelta	0.00	0.00	0.96	17.12	4.52	-0.85	64.37	-5.38	-4.53
SN-Tune	0.00	50.38	41.54	55.54	36.87	+31.50	74.37	+4.62	-26.88
RSN-Tune	0.38	64.23	39.80	60.18	41.15	+35.78	71.57	+1.82	-33.96
WSR-Tune (Ours)	0.00	0.00	0.77	19.19	4.99	-0.38	70.81	+1.06	**+1.44**
Vanilla LoRA FT	0.00	14.23	7.50	29.96	12.92	0.00	70.74	0.00	0.00
SafeLoRA	0.19	0.00	1.35	18.23	4.94	-7.98	69.29	-1.45	+6.53
SaLoRA	0.00	1.92	10.38	32.96	11.32	-1.60	68.23	-2.51	-0.91
WSR-LoRA (Ours)	0.00	0.77	1.35	18.69	5.20	-7.72	70.81	+0.07	**+7.79**

Table 21: Complete results on Llama-2-7B-Chat with MedQA and ARC-C fine-tuning. Safety is measured by ASR under Direct, AutoDAN, PAIR, and PAP, with JB AVG denoting their mean. Raw scores are percentages and deltas are percentage-point differences. Deltas in the upper comparison block, including AsFT and Lisa, are relative to Full Params FT; those in the lower comparison block are relative to Vanilla LoRA FT. AsFT and Lisa use LoRA adapters. We use $\Delta_{\mathrm{Overall}} = -\Delta_{\mathrm{S}} + \Delta_{\mathrm{D}}$. Green cells highlight the Overall changes of WSR methods.

	MedQA								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\mathrm{S}}}$	**MedQA**	$\boldsymbol{\Delta_{\mathrm{D}}}$	$\boldsymbol{\Delta_{\mathrm{Overall}}}$
Full Params FT	0.00	0.00	10.00	21.08	7.77	0.00	44.30	0.00	0.00
SafeInstr	0.00	0.00	10.00	20.27	7.57	-0.20	43.83	-0.47	-0.27
Resta	0.00	0.00	7.50	17.35	6.21	-1.56	41.01	-3.29	-1.73
SEAL	0.38	13.27	45.58	75.31	33.64	+25.87	35.74	-8.56	-34.43
AsFT	0.00	0.00	5.19	10.23	3.86	-3.91	32.84	-11.46	-7.55
Lisa	0.00	0.00	12.12	15.73	6.96	-0.81	31.81	-12.49	-11.68
SafeDelta	0.00	0.00	6.54	15.46	5.50	-2.27	33.62	-10.68	-8.41
SN-Tune	0.19	0.96	25.00	49.58	18.93	+11.16	42.89	-1.41	-12.57
RSN-Tune	0.00	0.58	22.31	41.15	16.01	+8.24	43.44	-0.86	-9.10
WSR-Tune (Ours)	0.00	0.00	7.12	14.69	5.45	-2.32	43.28	-1.02	**+1.30**
Vanilla LoRA FT	0.00	0.19	23.46	40.81	16.12	0.00	45.95	0.00	0.00
SafeLoRA	0.00	0.19	20.38	38.77	14.84	-1.28	45.95	0.00	+1.28
SaLoRA	0.00	1.15	29.23	69.54	24.98	+8.86	42.11	-3.84	-12.70
WSR-LoRA (Ours)	0.00	0.00	13.65	26.42	10.02	-6.10	45.17	-0.78	**+5.32**

	ARC-C								
	Safety ↓						**Downstream ↑**		**Overall ↑**
Method	**Direct**	**AutoDAN**	**PAIR**	**PAP**	**JB AVG**	$\boldsymbol{\Delta_{\mathrm{S}}}$	**ARC-C**	$\boldsymbol{\Delta_{\mathrm{D}}}$	$\boldsymbol{\Delta_{\mathrm{Overall}}}$
Full Params FT	0.00	0.00	31.54	51.00	20.64	0.00	56.31	0.00	0.00
SafeInstr	0.00	0.00	22.88	22.00	11.22	-9.42	58.45	+2.14	+11.56
Resta	0.00	0.00	25.19	39.12	16.08	-4.56	56.48	+0.17	+4.73
SEAL	0.00	0.58	29.23	53.54	20.84	+0.20	60.67	+4.36	+4.16
AsFT	0.00	0.00	5.96	18.08	6.01	-14.63	38.74	-17.57	-2.94
Lisa	0.00	0.00	9.42	27.23	9.16	-11.48	31.57	-24.74	-13.26
SafeDelta	0.00	0.00	10.77	22.27	8.26	-12.38	36.52	-19.79	-7.41
SN-Tune	0.00	64.42	30.77	81.23	44.11	+23.47	59.73	+3.42	-20.05
RSN-Tune	0.19	18.85	32.88	75.38	31.83	+11.19	59.13	+2.82	-8.37
WSR-Tune (Ours)	0.00	0.00	18.27	26.88	11.29	-9.35	58.53	+2.22	**+11.57**
Vanilla LoRA FT	0.00	0.00	20.00	36.23	14.06	0.00	61.09	0.00	0.00
SafeLoRA	0.00	0.00	18.46	33.12	12.90	-1.16	61.95	+0.86	+2.02
SaLoRA	0.00	0.38	21.35	36.85	14.64	+0.58	62.97	+1.88	+1.30
WSR-LoRA (Ours)	0.00	0.00	18.85	29.69	12.13	-1.93	61.57	+0.48	**+2.41**

G Visualization of Baseline Compatibility Results

Figure 5 visualizes the results in Table 3. Each point represents the safety-downstream trade-off of an existing baseline, and each arrow indicates the change after applying WSR-Tune. The figure shows consistent improvements for SN-Tune, SafeInstr, and SEAL on Llama-2-7B-Chat and Llama-2-13B-Chat.

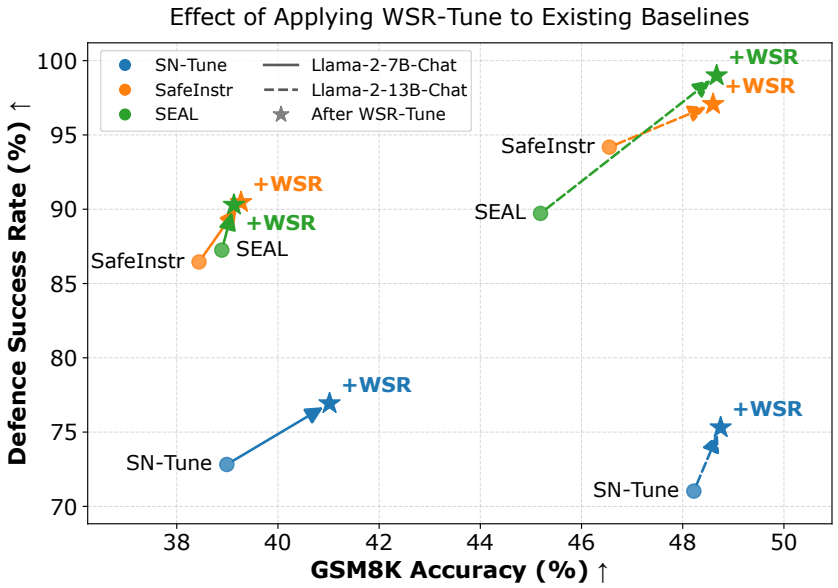

Figure 5: Visualization of the results in Table 3. Each arrow indicates the change from an existing baseline to its WSR-Tune-enhanced variant, where "+WSR" denotes applying WSR-Tune on top of the corresponding baseline. The x-axis denotes GSM8K accuracy, and the y-axis denotes defense success rate. Moving toward the upper-right direction indicates a better safety–downstream trade-off.

H Robustness of the Safety Metric: An LLM-Based Refusal Judge

We additionally evaluate the generated responses using WildGuard [57], an LLM-based classifier, on Llama-2-7B-Chat and Llama-2-13B-Chat after GSM8K fine-tuning. We use WildGuard's response-refusal predictions, counting a non-refusal response as a successful attack. This evaluation uses the same attack prompts, test examples, system prompt, and generated responses as the main experiments.

Table 22 summarizes the results. WSR-Tune achieves the largest overall improvement among the compared methods. These results are consistent with the favorable safety–utility trade-off observed in our main experiments, showing that this trend also holds under LLM-based classification.

I Comparison with Null space projection in Original Space

A common way to separate safety-relevant and task-relevant components is to construct subspaces in the original parameterization and explicitly project weights or updates onto, or away from, these subspaces. In particular, null-space projection can remove components aligned with a utility subspace from a safety-related subspace, yielding a more safety-specific component. WSR-Tune differs in that it does not directly project the original weight matrix onto a predefined safety subspace. Instead, it first constructs a safety-conditioned coordinate system and performs importance estimation and selective protection within that reparameterized space.

Activation-based subspace analysis [38] provides a representative example of this projection-based approach: it derives safety- and utility-related subspaces from projected activations and isolates safety-specific components by projecting the safety subspace onto the null space of the utility subspace.

Table 22: Comparison under an **LLM-based refusal judge** (WildGuard) instead of refusal-keyword matching, on Llama-2-7B-Chat and Llama-2-13B-Chat with GSM8K fine-tuning. The generations, attacks, and system prompt are identical to Table 2; only the judge is replaced. Safety is measured by Jailbreak (JB) AVG (averaged over Direct, AutoDAN, PAIR, and PAP; lower is better); downstream performance is task accuracy (higher is better). We report changes relative to Full Params FT: Δ_{S} (safety, lower is better) and Δ_{D} (downstream, higher is better), and $\Delta_{\mathrm{Overall}} = -\Delta_{\mathrm{S}} + \Delta_{\mathrm{D}}$. Raw scores are percentages and deltas are percentage-point differences. Green cells highlight the Overall changes of WSR-Tune.

	Llama-2-7B-Chat					**Llama-2-13B-Chat**				
	Safety ↓		**Downstream ↑**		**Overall ↑**	**Safety ↓**		**Downstream ↑**		**Overall ↑**
Method	**JB AVG**	$\boldsymbol{\Delta_{\mathrm{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\mathrm{D}}}$	$\boldsymbol{\Delta_{\mathrm{Overall}}}$	**JB AVG**	$\boldsymbol{\Delta_{\mathrm{S}}}$	**GSM8K**	$\boldsymbol{\Delta_{\mathrm{D}}}$	$\boldsymbol{\Delta_{\mathrm{Overall}}}$
Full Params FT	17.64	0.00	41.17	0.00	0.00	8.64	0.00	45.94	0.00	0.00
SafeInstr	15.26	-2.38	38.44	-2.73	-0.35	5.82	-2.82	46.55	+0.61	+3.43
Resta	10.46	-7.18	36.92	-4.25	+2.93	8.33	-0.31	44.20	-1.74	-1.43
SEAL	12.12	-5.52	38.89	-2.28	+3.24	9.08	+0.44	45.19	-0.75	-1.19
AsFT	5.29	-12.35	19.71	-21.46	-9.11	0.98	-7.66	28.43	-17.51	-9.85
Lisa	8.47	-9.17	14.40	-26.77	-17.60	1.53	-7.11	36.24	-9.70	-2.59
SafeDelta	5.95	-11.69	19.79	-21.38	-9.69	1.01	-7.63	28.43	-17.51	-9.88
SN-Tune	25.16	+7.52	38.99	-2.18	-9.70	21.43	+12.79	48.22	+2.28	-10.51
RSN-Tune	20.64	+3.00	40.26	-0.91	-3.91	21.18	+12.54	49.96	+4.02	-8.52
WSR-Tune (Ours)	7.44	-10.20	38.99	-2.18	**+8.02**	1.46	-7.18	49.58	+3.64	**+10.82**

Original-space projection. Let $W_l \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ denote a linear projection and $X_l \in \mathbb{R}^{d_{\mathrm{in}} \times N}$ its input activations. This approach performs SVD on the projected activations,

$$W_l X_l \in \mathbb{R}^{d_{\mathrm{out}} \times N}, \tag{23}$$

and uses the leading output singular vectors $U_{l,r} \in \mathbb{R}^{d_{\mathrm{out}} \times r}$ to construct

$$W_l^c = U_{l,r}(U_{l,r})^\top W_l. \tag{24}$$

It further isolates safety-specific modifications through projections such as

$$\Delta W_l = (I - \Pi_l^u)\Pi_l^s W_l. \tag{25}$$

Thus, this approach explicitly separates safety-specific components in the original parameterization by first projecting onto the safety subspace and then removing components shared with the utility subspace.

WSR-Tune. WSR-Tune instead constructs a complete basis from the covariance of safety input activations,

$$H_l H_l^\top = U_l \Sigma_l U_l^\top, \tag{26}$$

and reparameterizes the weight matrix as

$$\widetilde{W}_l = V_l^\top W_l U_l. \tag{27}$$

Safety importance is then evaluated for each coefficient $\widetilde{W}_l(i,j)$, and downstream adaptation is restricted by

$$\Delta W_l = V_l \left(\Delta \widetilde{W}_l \odot (1 - M_l)\right) U_l^\top. \tag{28}$$

Therefore, WSR-Tune does not preserve an entire rank component. Its preservation unit is an individual coefficient in the reparameterized basis, and the resulting update is not required to be low-rank.

With the default $V_l = I$, let

$$S_{l,i} = \{j : M_l(i,j) = 1\} \tag{29}$$

denote the basis directions protected for output row i. The corresponding projector is

$$P_{l,i} = \sum_{j \in S_{l,i}} u_j u_j^\top. \tag{30}$$

Hence, different output rows can preserve different subsets of the safety-conditioned input basis. In contrast, rank-level projection couples parameters through shared output-rank components. This provides WSR-Tune with a finer-grained preservation mechanism than rank-level projection.

Table 23: Comparison with rank- and neuron-level fine-tuning defenses on Llama-2-7B-Chat with GSM8K. JB AVG denotes average ASR and downstream performance is GSM8K accuracy. Raw scores are percentages and deltas are percentage-point differences relative to Full Params FT. We use $\Delta_{\text{Overall}} = -\Delta_{\text{S}} + \Delta_{\text{D}}$. The green cell highlights the Overall change of WSR-Tune.

	Safety ↓		Downstream ↑		Overall ↑
Method	**JB AVG**	$\mathbf{\Delta_S}$	**GSM8K**	$\mathbf{\Delta_D}$	$\mathbf{\Delta_{Overall}}$
Rank-level protection	15.11	-5.67	39.42	-1.75	+3.92
Neuron-level protection	15.77	-5.01	40.33	-0.84	+4.17
WSR-Tune	6.90	-13.88	38.99	-2.18	**+11.70**

The distinction also applies to importance scoring. Since

$$\frac{\partial L}{\partial \widetilde{W}_l} = V_l^\top \frac{\partial L}{\partial W_l} U_l, \tag{31}$$

coordinate-wise importance ranking depends on the chosen basis. Thus, applying the same gradient-based criterion after reparameterization generally identifies a different protected subspace from applying it directly in the original parameterization.

Fine-tuning-stage comparison. To compare these protection strategies, we adapt the component-identification procedures of [38] into downstream fine-tuning-stage defenses at both the rank and neuron levels. Rank-level protection preserves the safety-relevant output-rank components identified from

$$W_0 X_{\text{safe}} = U \Sigma V^\top, \tag{32}$$

whereas neuron-level protection freezes the identified safety-critical neurons. Table 23 reports the results under the same Llama-2-7B-Chat and GSM8K setting as our main experiments.

WSR-Tune substantially improves safety preservation over both original-space protection variants while maintaining comparable downstream performance. These results support the benefit of coefficient-level selection in a safety-conditioned reparameterized space, rather than preserving coupled rank or neuron components in the original parameterization.

J Additional Related Work on Harmful Fine-Tuning Defenses

We complement Section 2 with a broader discussion of harmful fine-tuning defenses, emphasizing parameter preservation and update constraints. Following the stage-based taxonomy of [59], we distinguish alignment-stage preparation, interventions during downstream fine-tuning, and post-tuning restoration. The mechanism-based groups below are not mutually exclusive and do not assume identical threat models or access to safety data.

Safety-critical components and parameter freezing. Safety-related components have been investigated at different granularities. [40] identify middle layers important for safety and propose SPPFT, which freezes these layers while updating the remaining parameters. [38] examine safety-critical neurons and low-rank components, including activation-derived safety and utility projections in their low-rank analysis. Their analysis also shows that freezing identified safety-critical neurons does not generally prevent subsequent fine-tuning attacks. These findings distinguish identifying components responsible for existing safety behavior from ensuring its preservation throughout adaptation.

Optimization and gradient constraints. Several fine-tuning-stage methods control how parameters change rather than freezing entire layers. AsFT [24] uses the weight difference between aligned and unaligned models as an alignment direction and penalizes update components orthogonal to it, motivated by a narrow safety basin. Lisa [8] alternates safety and task optimization, using proximal penalties relative to the preceding switching checkpoint to limit drift. SafeGrad [25] removes task-gradient components that conflict with an alignment gradient before combining the two gradients; its alignment objective uses KL divergence to a frozen safety-aligned reference model. SPF [32]

instead extracts a low-rank subspace from safety-loss gradients and, when safety and utility gradients conflict, projects the utility gradient onto its orthogonal complement. Non-conflicting updates are retained. These methods therefore differ in the safety signal used to constrain adaptation: checkpoint differences, optimization proximity, reference distributions, or gradient-derived subspaces.

Low-rank and continual adaptation. SaLoRA [27] combines task-specific adapter initialization with a fixed projection that restricts adapter-induced feature changes to directions orthogonal to estimated safety features. GuardSpace [33] uses covariance-preconditioned weight decomposition to initialize adapters from low-sensitivity components and additionally projects adapter updates into a null space derived from harmful-prompt activations. Its initialization and null-space restriction are separate components. SafeAnchor [34] addresses cumulative safety erosion across sequential domain adaptations: it estimates a Fisher-based safety subspace in LoRA parameter space, incrementally updates this subspace, and combines gradient projection with safety-triggered corrective replay. Thus, low-rank parameterization alone does not specify a common defense mechanism; feature preservation, null-space restrictions, and continual safety monitoring impose different constraints.

Safety supervision and data selection. SafeInstr [21] augments downstream training with safety examples. SEAL [22] instead learns sample weights through bilevel optimization, evaluating the safety loss of a model trained on weighted downstream data, and then selects high-ranked downstream samples for fine-tuning. SPARD [31] selects a compact safety dataset using a relevance–diversity determinantal point process. After each utility update, it applies a trust-region-limited correction based on a linearized safety constraint when the safety loss exceeds a threshold. SEAL selects downstream training samples, whereas SPARD selects safety examples used to constrain optimization.

Token- and attention-level constraints. [60] show that refusal alignment can be concentrated in early response tokens and investigate both deeper safety alignment and constrained fine-tuning. Their constrained objective limits deviations from the aligned model with token-dependent regularization, especially at the beginning of a response. Surgery [61] operates on attention behavior, penalizing positive attention-sink divergence measured between harmful and safe responses to harmful prompts. These methods constrain output or attention behavior during optimization rather than directly freezing a selected block of weight coefficients.

Perturbation and representation-based alignment. Alignment-stage defenses seek to make subsequent harmful adaptation less effective. Vaccine [62] optimizes alignment under adversarial perturbations of hidden embeddings. Targeted Vaccine [63] concentrates perturbation and training on selected layers according to their importance, rather than perturbing all layers uniformly. Representation Noising [64] combines benign-behavior retention with suppression of harmful outputs and noise-distribution matching of harmful intermediate representations. Booster [65] penalizes the reduction in harmful loss after a simulated harmful parameter perturbation while optimizing the alignment objective. These approaches modify the model's preparation for subsequent adaptation, rather than necessarily changing the downstream optimizer.

Conditional degradation under malicious adaptation. Other defenses deliberately couple harmful adaptation with loss of model capability. Self-destructive language models [66] encourage opposing gradients on benign and harmful data and incorporate harmful-knowledge unlearning, aiming to make harmful fine-tuning degrade general utility. CTRAP [67] plants a collapse objective evaluated after a simulated harmful update, encouraging context-independent fixed-token outputs when the trap is triggered. SDD [68] trains on harmful instructions with unrelated, high-quality benign responses, so that malicious fine-tuning is intended to impair general capabilities. These methods seek to retain benign usability while accepting capability degradation under malicious adaptation as part of the defense.

Conditioning and model locking. A related but broader perspective changes the difficulty of subsequent optimization. [69] study model immunization through the condition number of the downstream linear-probing Hessian, shaping representations to impede a designated harmful task while preserving pre-training performance. Their linear-probing analysis and vision experiments provide an optimization perspective rather than direct evidence of safety preservation in aligned LLMs. Spectral Deformation [70] enlarges leading singular values of weight matrices while optimizing a retention loss, aiming to make further training impractically slow without substantially degrading

the original task. This model-locking objective restricts subsequent trainability more broadly than safety-preserving task adaptation.

Post-tuning restoration. Post-tuning methods modify the result of adaptation rather than its training trajectory. SafeLoRA [26] applies alignment-aware projection to learned low-rank updates, while SafeDelta [28] adjusts task deltas with safety-aware compensation and RESTA [29] adds a safety vector through task arithmetic. These interventions differ in timing from constraints enforced throughout fine-tuning.

Relation to WSR-Tune. WSR-Tune belongs to the fine-tuning-stage family and shares the goal of protecting safety-relevant structure while permitting task adaptation. Its distinction is the combination of a fixed, complete basis derived from safety activations with importance-based selection and protection of reparameterized coefficients. Unlike online gradient-conflict correction, the basis and mask remain fixed during downstream training. Coefficient freezing in this basis induces row-specific linear constraints in the original weights, so it is related to projection-based methods rather than mathematically opposed to them. WSR-LoRA extends this principle through input-factor reparameterization and importance masks on both adapter factors, without equating factor-wise protection with exact freezing of their product. Our contribution is this reparameterization–selection composition, not a new freezing primitive. Detailed geometric and PEFT formulations are provided in Appendices D and E.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main claim that WSR-Tune preserves safety during downstream adaptation by constructing a safety-conditioned reparameterized weight space and freezing safety-critical coefficients. The claims are supported by the method description and experimental comparisons across multiple models and downstream tasks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses computational overhead as a limitation, including the cost of SVD-based basis construction and safety-critical coefficient estimation. Runtime details are provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results requiring theorem statements or proofs. The mathematical formulations are used to describe the optimization objective, weight-space reparameterization, and masking procedure.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the main experimental setup, including safety and downstream datasets, evaluation metrics, model families, baselines, target modules, and the freeze ratio used for WSR-Tune. The experimental hyperparameters are summarized in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available in a GitHub repository, with instructions for running WSR-Tune and reproducing the main experimental results. The paper also describes the datasets, model backbones, evaluation protocols, and hyperparameter settings needed for reproduction.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the safety dataset, downstream datasets, evaluation metrics, attack settings, baselines, target modules, and key implementation choices. Training hyperparameters such as epochs and learning rates are described as being provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The current version reports point estimates for safety and downstream performance but does not yet include error bars, confidence intervals, or statistical significance tests.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the compute environment, including GPU type (NVIDIA B200), CPU configuration, memory, and training setup in the Appendix. We also report runtime comparisons between standard fine-tuning and WSR-Tune.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is intended to improve the safety preservation of LLMs during downstream fine-tuning and does not involve deceptive data collection, human-subject experiments, or privacy-sensitive user data. The work uses standard public benchmarks and safety evaluation datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The current version does not include a dedicated broader impacts discussion. However, the work is directly related to LLM safety, and we will include a discussion of potential benefits, such as safer downstream adaptation, and risks, such as limitations or misuse of safety mechanisms, in the final version.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new high-risk pretrained model, scraped dataset, or deployed system.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited the original paper that produced datasets used in our work in Section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce any new dataset, model, or benchmark.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects or human-participant data collection.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use an LLM as a non-standard component for generating scientific claims or performing the proposed algorithm. LLMs are the objects of study and evaluation; any use of LLMs for writing or editing assistance does not affect the scientific methodology or results.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.