Past, Future, All at Once: Mitigating Stability-Plasticity Dilemma via Post-hoc JANUS Rectification

Zhilong Zheng[1,2,*], **Letian Tao**[1,2,*], **Yujie Yang**[1,2], **Wei Xiong**[2], **Kehua Sheng**[2]
Bo Zhang[2], **Jingliang Duan**[1,2], **Keqiang Li**[1], **Yang Guan**[1,2,†], **Shengbo Eben Li**[1,†]
[1]School of Vehicle and Mobility & College of AI, Tsinghua University
[2]DiDi Voyager Labs, DiDi Autonomous Driving
*Equal contribution, [†]Corresponding authors: {yguan,lishbo}@tsinghua.edu.cn
https://github.com/zhengzl18/JANUS

Abstract

Fine-tuning foundation models on new tasks inevitably suffer from catastrophic forgetting. While existing works attempt to mitigate this on the basis of parameter-efficient fine-tuning methods, they adopted an overly restrictive Subspace Orthogonality condition. In this paper, we introduce a purely *post-hoc* and *tuning-agnostic* weight rectification framework that achieves *Parameter Space Orthogonality*, which is the necessary and sufficient condition for preserving historical performance to the first order. By projecting parameter updates into the JAcobian NUll Space (JANUS), our method significantly recovers compromised historical knowledge without interfering with the underlying fine-tuning process. To overcome the local validity of the Jacobian approximation, we further propose a Multi-step Adaptive Rectification mechanism that utilizes the JANUS shift to dynamically verify the valid trust region and adjust step sizes. Coupled with our proposed ghost projection, ghost orientation comparison, and sequence-level singular value decomposition compression techniques, JANUS also achieves great temporal and spatial efficiency. Experiments demonstrate that JANUS seamlessly integrates with various fine-tuning methods, significantly mitigating the stability-plasticity dilemma by recovering historical knowledge while preserving downstream task adaptation.

1 Introduction

In the era of large-scale models, fine-tuning (FT), ranging from supervised learning [1] to reinforcement learning [2, 3], has become a fundamental paradigm. Adapting foundation models to downstream tasks have achieved unprecedented success across a wide range of domains, including question-answering [4], math [5], code [6] and instruction following [7, 8]. However, FT on new tasks inevitably suffers from catastrophic forgetting (CF) [9], where the model significantly loses its previously acquired general knowledge and abilities while learning new skills. This poses the well-known stability-plasticity dilemma: trading off between retaining historical knowledge (stability) and subsuming new information (plasticity).

Concurrently, Parameter-Efficient Fine-Tuning (PEFT) is also essential due to the massive scale of modern large models [10]. Low-Rank Adaptation (LoRA) [11] is arguably the most widely adopted PEFT method, where the weight update $\Delta \boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is given by the product of two low rank matrices $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$. Recent literature has extensively investigated strategies to alleviate CF within LoRA-based FT frameworks. One paradigm is to pursue *output invariance*, ensuring that for every linear layer, the weight update $\Delta \boldsymbol{W}$ does not alter the layer's

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

Constraint \ Target	Output invariance $\boldsymbol{M} = \boldsymbol{X}$	Loss invariance $\boldsymbol{M} = \nabla_{\boldsymbol{W}^\top} \mathcal{L}_A$
Subspace orthogonality $\Delta \boldsymbol{W} \boldsymbol{M} = \boldsymbol{0}$ Suff. ✓, Nec. ✗	[12, 13, 14] [15, 16, 17] [18, 19]	[20, 21]
Parameter space orthogonality $\mathrm{Tr}(\Delta \boldsymbol{W} \boldsymbol{M}) = 0$ Suff. ✓, Nec. ✓	–	JANUS

Table 1: Comparison of forgetting-free conditions. **Suff.**: sufficient; **Nec.**: necessary. $\mathcal{L}_A$: past task loss.

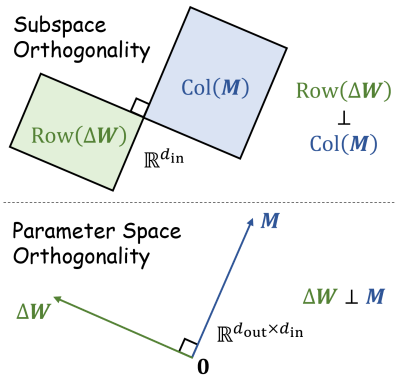


Figure 1: Comparison of two kinds of orthogonality.

output [12, 13, 14, 15, 16, 17, 18, 19]. Another paradigm, which aligns more closely with the ultimate objective of forgetting-free FT, focuses on *loss invariance*, ensuring that the $\Delta \boldsymbol{W}$ of any linear layer does not increase the loss on historical tasks [20, 21]. Their shared core idea is to enforce orthogonality, either achieved by projection [12, 13, 17, 18], initialization [14, 15, 16, 19], or by penalty [20, 21]. However, all these methods essentially strive for *Subspace Orthogonality*. They aim to make the row space of $\Delta \boldsymbol{W}$ orthogonal to the column space of a specific matrix $\boldsymbol{M}$, i.e., $\Delta \boldsymbol{W} \boldsymbol{M} = 0$. For output invariance, the matrix $\boldsymbol{M}$ comes from the layer inputs $\boldsymbol{X}$ of historical tasks [12, 13, 14, 15, 16, 17, 18, 19]. For loss invariance, $\boldsymbol{M}$ is the parameter updates from historical tasks which is assumed to be a surrogate for the historical gradients [20, 21]. The problem is that the subspace orthogonality constraint $\Delta \boldsymbol{W} \boldsymbol{M} = \boldsymbol{0}$ is overly restrictive; it is a sufficient but unnecessary condition for forgetting-free FT, making it extremely difficult to satisfy in practice. Furthermore, methods forcing this orthogonality through initialization face a dilemma: the orthogonality guarantee relies on a specific initialization of the matrix $\boldsymbol{A}$, which requires it to be frozen, while a frozen $\boldsymbol{A}$ hinders plasticity. Moreover, these special initializations are inherently incompatible with more advanced LoRA initialization strategies, such as PiSSA [22].

To address these theoretical and practical bottlenecks, we introduce the *Parameter Space Orthogonality* condition ($\mathrm{Tr}(\Delta \boldsymbol{W} \boldsymbol{M}) = \boldsymbol{0}$) that is necessary and sufficient for first-order forgetting-free FT. Tab. 1 and Fig. 1 gives a comparison of different forgetting-free conditions. Given the massive parameter space of large models, there is a vast space for this condition to hold, while maintaining plasticity. Leveraging this insight, we propose JANUS (JAcobian NUll Space projection), a purely post-hoc and tuning-agnostic weight rectification framework. JANUS rectifies the parameter updates $\Delta \boldsymbol{W}$ after the FT process is complete, and imposes no assumption or requirement on the FT method. In other words, it recovers the historical knowledge that might have been compromised during the unconstrained FT phase. The advantages of this post-hoc and tuning-agnostic nature are threefold: (1) it is inherently compatible with any FT method; (2) it can be applied directly to off-the-shelf fine-tuned models without requiring re-training; and (3) it adds no overhead to the FT process. To overcome the local validity of the Jacobian approximation, we introduce the *JANUS shift* as a proxy metric for forgetting. Based on this metric, we design a Multi-step Adaptive Rectification mechanism that safely navigates the parameter space along the loss contour, progressively advancing toward the region of high plasticity. To ensure practical scalability, we develop "ghost projection" and "ghost orientation comparison" techniques and sequence-level Singular Value Decomposition (SVD) compression, which relieve the burden of instantiating the full Jacobian matrix or performing per-sample backpropagation. Consequently, we can execute the post-rectification of a LLaMA-2-7b model in just 30 minutes using a single NVIDIA A100 GPU.

The main contributions of our work are summarized as follows:

- **Parameter Space Orthogonality Theory:** We systematically reveal that the widely adopted subspace orthogonality ($\Delta \boldsymbol{W} \boldsymbol{M} = \boldsymbol{0}$) is overly strict and practically infeasible. Instead, we introduce the parameter space orthogonality ($\mathrm{Tr}(\Delta \boldsymbol{W} \boldsymbol{M}) = \boldsymbol{0}$), the equivalent condition for preventing CF, which is highly feasible for large models.

- **Post-hoc Multi-step Adaptive Rectification in JANUS:** We propose the JANUS shift as a proxy indicator of forgetting and design a plug-and-play, tuning-agnostic algorithm which adaptively and gradually rectifies parameter updates into the JANUS through projection to recover historical knowledge. It incurs no training overhead and seamlessly integrates with arbitrary FT methods. Furthermore, our proposed "ghost" operations and SVD compression ensure both temporal and spatial efficiency.
- **Empirical Validation of the Stability-Plasticity Breakthrough:** Extensive experiments across diverse models and tasks confirm that JANUS significantly mitigates the stability-plasticity dilemma. Our method consistently pushes the Pareto front outward, achieving near-perfect knowledge recovery with negligible degradation to plasticity. Additionally, we empirically validate the use of JANUS shift as an indicator of forgetting by revealing a statistically significant positive correlation between this metric and model stability.

2 Related Works

Parameter-Efficient Fine-Tuning With the exponential growth in the parameter scale of large models, full FT has become computationally prohibitive [10]. To address this, PEFT techniques adapt models to downstream tasks with only a marginal fraction of parameters, such as partial FT [23, 24, 25, 26, 27] and parameter-efficient adaptation [11, 28, 29, 30]. Among these, LoRA [11] has emerged as the standard paradigm by injecting trainable low-rank matrices into the frozen pretrained weights. Building upon LoRA, a series of work seek to enhance representational capacity [22, 31, 32, 33]. For instance, PiSSA [22] leverages SVD to identify and separate the essential low-rank parts of the model to be the initializations of the adapters, achieving better plasticity and quicker convergence. Being purely post-hoc and tuning-agnostic, our proposed weight rectification in JANUS method seamlessly integrates any PEFT technique, even full FT.

Mitigating Catastrophic Forgetting Catastrophic forgetting is a foundational challenge in continual learning [9, 34], where models drastically degrade in historical task performance when adapting to new data. Plenty of methods have been proposed to mitigate this problem, including regularization-based [35, 36, 37, 38], optimization-based [39, 40, 41, 42] and architecture-based [43, 44, 45, 46] approaches. Regularization methods constrain updates during training, usually through an old-task penalty, and introduce an additional coefficient to tune. Orthogonality-based methods instead treat the forgetting-free condition as a constraint, avoiding the trade-off introduced by tuning a penalty coefficient. Architecture-based methods preserve old tasks by allocating additional task-dependent capacity. In general, it is preferred that the model's parameter count does not increase as the number of tasks grows. To be applied on modern large-scale models, the ability to integrate with PEFT methods is also essential. Consequently, many works seek to mitigate CF built upon LoRA [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. These methods emphasize subspace orthogonality to prevent interference, which poses restriction on both stability and plasticity. In contrast, our work emphasizes parameter space orthogonality, establishing a highly compatible and efficient mechanism for unlocking the potential of models and mitigating the stability-plasticity dilemma.

3 Theoretical Analysis of Forgetting-Free Mechanisms

3.1 Problem Formulation

Given network parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^n$ pretrained on a past task $\mathcal{T}_A$ with dataset $\mathcal{D}_A$, where n is the total number of parameters, our goal is to fine-tune the network to obtain updated parameters $\hat{\boldsymbol{\theta}}$ on a new task $\mathcal{T}_B$ while ensuring performance stability on $\mathcal{T}_A$. Mathematically, this requires the loss on the previous task to remain invariant, i.e., $\mathcal{L}_A(\hat{\boldsymbol{\theta}}; \mathcal{D}_A) \approx \mathcal{L}_A(\boldsymbol{\theta}_0; \mathcal{D}_A)$.

3.2 The Conditions Towards Forgetting-Free Fine-Tuning

Consider a specific linear layer in the network parameterized by $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Let $\boldsymbol{X} \in \mathbb{R}^{d_{\text{in}} \times T}$ denote its layer input under a specific data point $d_{A,i} \in \mathcal{D}_A$, and $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} \in \mathbb{R}^{d_{\text{out}} \times T}$ denote its corresponding output, where T is the sequence length. Based on the first-order Taylor expansion,

for a small weight perturbation $\Delta \boldsymbol{W}$, the change in the loss function $\mathcal{L}_A^{(i)}$ evaluated on $d_{A,i}$ can be approximated as [47]

$$\Delta \mathcal{L}_A^{(i)} \approx \text{Tr}\left(\Delta \boldsymbol{W} \nabla_{\boldsymbol{W}^\top} \mathcal{L}_A^{(i)}\right) = \text{Tr}\left(\Delta \boldsymbol{W} \boldsymbol{X} \nabla_{\boldsymbol{Y}^\top} \mathcal{L}_A^{(i)}\right), \tag{1}$$

where $\nabla_{\boldsymbol{W}} \mathcal{L}_A^{(i)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and $\nabla_{\boldsymbol{Y}} \mathcal{L}_A^{(i)} \in \mathbb{R}^{d_{\text{out}} \times T}$ are the gradients of the loss w.r.t. the weights and the layer outputs, respectively, and we utilize the chain rule $\nabla_{\boldsymbol{W}} \mathcal{L}_A^{(i)} = \nabla_{\boldsymbol{Y}} \mathcal{L}_A^{(i)} \boldsymbol{X}^\top$. Consequently, the necessary and sufficient condition for a first-order forgetting-free update (i.e., $\Delta \mathcal{L}_A^{(i)} \approx 0$) is:

$$\text{Tr}\left(\Delta \boldsymbol{W} \boldsymbol{X} \nabla_{\boldsymbol{Y}^\top} \mathcal{L}_A^{(i)}\right) = 0. \tag{2}$$

Existing methods attempt to achieve this either by restricting $\Delta \boldsymbol{W} \boldsymbol{X} = 0$ (as in [12, 13, 14, 15, 16, 20]), or by forcing $\Delta \boldsymbol{W} \boldsymbol{X} \nabla_{\boldsymbol{Y}^\top} \mathcal{L}_A^{(i)} = 0$ (as in [17, 18]). Clearly, both of these formulations are sufficient but not necessary conditions for Eq. (2). They impose excessively strong constraints that can hardly be satisfied in practice. To illustrate this, let $\boldsymbol{M} \in \mathbb{R}^{d_{\text{in}} \times p}$ serve as a generalized representation for either the layer input $\boldsymbol{X} \in \mathbb{R}^{d_{\text{in}} \times T}$ or the gradient matrix $\nabla_{\boldsymbol{W}^\top} \mathcal{L}_A^{(i)} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$. The general formulation of these existing works can be abstracted as:

$$\Delta \boldsymbol{W} \boldsymbol{M} = 0. \tag{3}$$

This strict subspace orthogonality requires finding d_{out} distinct directions within $\mathbb{R}^{d_{\text{in}}}$, each of which is orthogonal to p directions. This is mathematically feasible only when $d_{\text{in}} - \text{rank}\,(\boldsymbol{M}) \geq d_{\text{out}}$. Unfortunately, this requirement is rarely met. The situation exacerbates when considering a batch of m data points, where each sample corresponds to a matrix $\boldsymbol{M}_i \in \mathbb{R}^{d_{\text{in}} \times p}$, collectively forming a joint matrix $\boldsymbol{M} \in \mathbb{R}^{d_{\text{in}} \times mp}$. This joint matrix can easily have full row rank and thus no non-trivial left null space. Intuitively speaking, prior works attempt to mitigate forgetting by rigorously constraining every single row of every linear layer individually. Not only does this overly restrictive constraint hinder the model's plasticity for adapting to the new task, but the inevitable violation of this strict condition in practice also undermines the theoretical foundation of these mechanisms.

However, if we revert to the necessary and sufficient condition defined in Eq. (2), the feasible space for $\Delta \boldsymbol{W}$ is vastly expanded. Specifically, Eq. (2) can be reformulated into a vectorized form:

$$\text{rvec}\left(\nabla_{\boldsymbol{W}} \mathcal{L}_A^{(i)}\right)^\top \text{rvec}\,(\Delta \boldsymbol{W}) = 0, \tag{4}$$

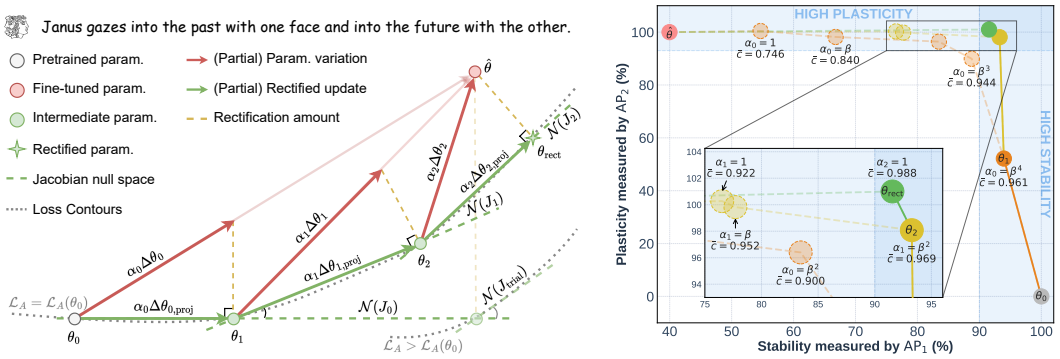
where $\text{rvec}(\cdot)$ is the row-major vectorization operator: $\text{rvec}\,(\boldsymbol{A}) \triangleq \left[\boldsymbol{a}_1^\top, \boldsymbol{a}_2^\top, \ldots, \boldsymbol{a}_p^\top\right]^\top \in \mathbb{R}^{pq}, \quad \forall \boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_p]^\top \in \mathbb{R}^{p \times q}$. In its vectorized form, Eq. (4) considers all the parameters in $\Delta \boldsymbol{W}$ as a whole rather than row by row. It merely requires finding a single direction in the parameter space $\mathbb{R}^{d_{\text{in}} d_{\text{out}}}$ that is orthogonal to one specific gradient direction. This is why it is termed parameter space orthogonality. Even when aggregating over m data points, satisfying this condition remains highly tractable due to the massive dimensionality of $d_{\text{in}} d_{\text{out}}$ in modern large-scale models.

By shifting our perspective from the strict subspace orthogonality characterized by Eq. (3) to the relaxed parameter space orthogonality formulated in Eq. (4), we provide a vast optimization space that fully unlocks the model's capacity to learn new tasks. Simultaneously, this relaxation renders the orthogonality constraint practically achievable, thereby guaranteeing stability.

4 Methods

4.1 Post-Hoc JANUS Rectification

To rectify the parameter updates, we replay a small subset of m data points from the previous tasks and compute the individual gradients to construct the Jacobian matrix for each linear layer in the network. Suppose the network consists of L linear layers. For the simplicity of notation, we assume that all linear layers share the same shape $(d_{\text{out}}, d_{\text{in}})$. The weight matrix of the j-th layer is denoted



(a) Schematic diagram (b) Visualization of the rectification process

Figure 2: Illustration of the Multi-step Adaptive Rectification in JANUS.

as $\boldsymbol{W}^{(j)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, and we define $\boldsymbol{\theta}^{(j)} \triangleq \text{rvec}(\boldsymbol{W}^{(j)}) \in \mathbb{R}^{d_{\text{in}} d_{\text{out}}}$ as its row-major vectorization. Let the gradient of the loss on the i-th data point with respect to $\boldsymbol{\theta}^{(j)}$ be denoted as $\boldsymbol{g}^{(i,j)} \triangleq \nabla_{\boldsymbol{\theta}^{(j)}} \mathcal{L}_A^{(i)}$. The Jacobian matrix for the j-th layer can then be constructed by stacking these individual gradient vectors:

$$\boldsymbol{J}^{(j)} = \left[\boldsymbol{g}^{(1,j)}, \boldsymbol{g}^{(2,j)}, \ldots, \boldsymbol{g}^{(m,j)}\right]^\top \in \mathbb{R}^{m \times d_{\text{in}} d_{\text{out}}}. \tag{5}$$

Upon the full completion of the FT process, we extract the overall parameter update $\Delta\boldsymbol{\theta}^{(j)} \triangleq \hat{\boldsymbol{\theta}}^{(j)} - \boldsymbol{\theta}_0^{(j)}$ between the initial pretrained weights $\boldsymbol{\theta}_0^{(j)}$ and the final adapted weights $\hat{\boldsymbol{\theta}}^{(j)}$. Subsequently, we perform the post-hoc *JANUS projection* on this update:

$$\Delta\boldsymbol{\theta}_{\text{proj}}^{(j)} = \Delta\boldsymbol{\theta}^{(j)} - \boldsymbol{J}^{(j)\top} \left(\boldsymbol{J}^{(j)} \boldsymbol{J}^{(j)\top}\right)^{-1} \boldsymbol{J}^{(j)} \Delta\boldsymbol{\theta}^{(j)}. \tag{6}$$

After this projection-based rectification, the updated parameter strictly satisfies:

$$\boldsymbol{g}^{(i,j)\top} \Delta\boldsymbol{\theta}_{\text{proj}}^{(j)} = 0, \quad \forall i \in [1, m] \cap \mathbb{Z},\ \forall j \in [1, L] \cap \mathbb{Z}. \tag{7}$$

Note that Eq. (7) is a reformulation of the parameter space orthogonality condition Eq. (4), which serves as the necessary and sufficient condition for achieving performance invariance at the layer level. Building upon this layer-wise property, we can further formally establish the global forgetting-free guarantee across the entire network.

Theorem 1 (Global Forgetting-Free Guarantee via Layer-wise Projection)**.** *Applying the post-hoc JANUS projection independently to each linear layer guarantees that the overall loss change on the replayed past task samples approximates zero to the first-order, i.e.,* $\Delta\mathcal{L}_A^{(i)} \approx 0, \forall i = 1, 2, ..., m.$

The proof is deferred to Appendix A.1. Thm. 1 mathematically solidifies the effectiveness of our approach. Crucially, both the calculation of the Jacobian $\boldsymbol{J}^{(j)}$ and the subsequent projection procedure can be executed in a purely post-hoc manner, thereby adding no overload to the FT stage. Meanwhile, this rectification procedure is tuning-agnostic: it merely requires the parameters before and after FT, imposing no assumptions on the underlying FT method.

4.2 Multi-step Adaptive Rectification

While JANUS identifies the directions along which the model can move with minimal degradation in performance on past tasks, it provides only local information. Even when parameter updates are strictly constrained within the JANUS at the initial point $\boldsymbol{\theta}_0$, a substantially large update distance may cause the parameters to deviate from the valid approximation region of the local Jacobian. To address this limitation, we propose a Multi-step Adaptive Rectification mechanism, which effectively navigates the parameter space along the loss contour, as illustrated in Fig. 2a.

Suppose that after t rectification steps, the current parameters are denoted as $\boldsymbol{\theta}_t^{(j)}$. The remaining parameter variation to be rectified is defined as $\Delta\boldsymbol{\theta}_t^{(j)} \triangleq \hat{\boldsymbol{\theta}}^{(j)} - \boldsymbol{\theta}_t^{(j)}$. For this variation, we compute

Algorithm 1 Multi-step Adaptive Rectification in JANUS

Input: Pretrained parameters $\boldsymbol{\theta}_0$, fine-tuned parameters $\hat{\boldsymbol{\theta}}$, decay factor β, acceptance threshold τ.
Output: Final rectified parameters $\boldsymbol{\theta}_{\text{rect}}$.

Initialize step $t \leftarrow 0$, current parameters $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_0$.
while True **do**
  for all j **do** ▷ Ghost projection (see Sec. 4.1 and Sec. 4.3)
    Compute $\boldsymbol{J}_t^{(j)}$ at $\boldsymbol{\theta}_t^{(j)}$ and the projected update $\Delta\boldsymbol{\theta}_{t,\text{proj}}^{(j)}$ via Eq. (6).
  end for
  Initialize trial step size: $\alpha_t \leftarrow 1$.
  while True **do**
    Let $\boldsymbol{\theta}_{\text{trial}} \leftarrow \boldsymbol{\theta}_t + \alpha_t \Delta\boldsymbol{\theta}_{t,\text{proj}}$
    for all j **do** ▷ Ghost orientation comparison (see Sec. 4.2 and Sec. 4.3)
      Compute trial Jacobian $\boldsymbol{J}_{\text{trial}}^{(j)}$ and avg. cosine $c^{(j)}$.
    end for
    if $\bar{c} \triangleq \text{mean}_j\{c^{(j)}\} \geq \tau$ **then**
      break ▷ Step size is within confidence region
    end if
    $\alpha_t \leftarrow \beta \cdot \alpha_t$ ▷ Decay step size and retry
  end while
  Update parameters: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_{\text{trial}}$.
  if $\alpha_t = 1$ **then**
    break ▷ Successfully rectified
  end if
  $t \leftarrow t + 1$
end while
return $\boldsymbol{\theta}_{\text{rect}} \equiv \boldsymbol{\theta}_{t+1}$

the Jacobian $\boldsymbol{J}_t^{(j)}$ at $\boldsymbol{\theta}_t^{(j)}$ and then the projected update $\Delta\boldsymbol{\theta}_{t,\text{proj}}^{(j)}$ according to Eq. (6). Subsequently, we initialize the step size $\alpha_t = 1$ to perform a trial rectification, yielding the trial parameters $\boldsymbol{\theta}_{\text{trial}}^{(j)} = \boldsymbol{\theta}_t^{(j)} + \alpha_t \Delta\boldsymbol{\theta}_{t,\text{proj}}^{(j)}$. To determine whether the current step size is overly large, we evaluate the Jacobian matrix $\boldsymbol{J}_{\text{trial}}^{(j)}$ at the trial point for every layer and then compute the average cosine of the principal angles [48] between the row spaces of $\boldsymbol{J}_{\text{trial}}^{(j)}$ and the current Jacobian $\boldsymbol{J}_t^{(j)}$, denoted as $c^{(j)}$. This procedure is termed *JANUS orientation comparison*, and we adopt $\bar{c} \triangleq \text{mean}_j \left\{c^{(j)}\right\}$ to quantify the JANUS shift. If $\bar{c}$ exceeds a predefined threshold τ, i.e., the JANUS shift is acceptable, we accept the current step size and set $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_{\text{trial}}$. Otherwise, we shrink the step size by a decay factor $\beta \in (0, 1)$ and repeat the trial rectification until the criterion is satisfied. This adaptive step size ensures that the intermediate parameters closely track the loss-invariant contour. When a rectification step is accepted with a full step size $\alpha_t = 1$, all remaining variation has been rectified, and the rectification phase is complete. If $\alpha_t < 1$, the mechanism proceeds to the next iterative step.

4.3 Efficient Calculation and Storage of the Jacobian

Directly computing and storing the full Jacobian $\boldsymbol{J}^{(j)} \in \mathbb{R}^{m \times d_{\text{in}} d_{\text{out}}}$ for all layers is computationally and spatially prohibitive. Temporally, it would require m separate backward passes to get the gradients for each sample, and Algorithm 1 demands repeated evaluations of the Jacobian at various parameter locations. Spatially, the complete Jacobian entails an $\mathcal{O}(Lmd^2)$ memory complexity in total (assuming $d_{\text{in}} \approx d_{\text{out}} \approx d$ for simplicity), posing a severe bottleneck even for offline storage.

Addressing the temporal challenges, we propose ghost projection and ghost orientation comparison techniques inspired by [49, 50]. By utilizing intermediate tensors (i.e., the layer inputs $\boldsymbol{X}^{(j)} \in \mathbb{R}^{m \times T \times d_{\text{in}}}$ and the pre-activation gradients $\boldsymbol{A}^{(j)} \in \mathbb{R}^{m \times T \times d_{\text{out}}}$) from batch-wise backpropagation, we bypass per-sample backpropagation and implicitly perform JANUS projection and JANUS orientation comparison without ever instantiating the high-dimensional Jacobian matrices. This approach enables the computation of all necessary quantities in a single backward pass, significantly reducing the computational overhead.

Regarding the spatial bottleneck, caching all $\boldsymbol{X}^{(j)}$ and $\boldsymbol{A}^{(j)}$ still requires $\mathcal{O}(2LmTd)$ space, which can still be spatially prohibitive under a large T. Fortunately, it has been demonstrated that they tend to exhibit strong correlation along the sequence dimension [51]. Leveraging this intrinsic low-rank property, we can significantly compress these tensors along the sequence dimension using rank-r SVD. This yields two low-dimensional tensors $\hat{\boldsymbol{X}}^{(j)} \in \mathbb{R}^{m\times r\times d_{\text{in}}}$ and $\hat{\boldsymbol{A}}^{(j)} \in \mathbb{R}^{m\times r\times d_{\text{out}}}$, reducing the spatial complexity to $\mathcal{O}(2Lmrd)$ ($r \ll \min\{d, T\}$) while preserving the principal gradient information. The details can be found in Appendix B.

In the experiments, with a batch size of $m = 256$ and a compressed rank of $r = 32$, these techniques enables the rectification of a LLaMA-2-7b model on a single A100 GPU within 30 minutes (the exact time depending on the number of rectification iterations). For a detailed time cost profile, please refer to Appendix B.4. Furthermore, caching a complete set of $\hat{\boldsymbol{X}}^{(j)}$ and $\hat{\boldsymbol{A}}^{(j)}$ across all layers in `float16` format requires only less than 40 GB of disk storage.

5 Experiments

5.1 Experimental Setup

Tasks To evaluate the effectiveness of our method, we fine-tune LLaMA-2-7b [52] and LLaMA-3-8b [53] across three tasks: Math, Code, and Instruction Following (IF). Following the protocol established by [14], the world knowledge is evaluated by TriviaQA [54], NQ open [55], and WebQS [56] (collectively termed knowledge datasets). The Math, Code, and IF tasks are evaluated by GSM8k/Math [57, 5], HumanEval/MBPP [58, 59], and MTBench [7], respectively.

Compared Methods We benchmark our approach against several baselines: (1) Full Fine-tuning (FF); (2) LoRA [11]; (3) PiSSA [22]; (4) CorDA [14]; (5) MiLoRA [15]; and (6) LoRA-Null [16]. Our post-hoc rectification mechanism is applied to the resulting parameters of FF, LoRA, and PiSSA, which we denote as **JANUS-F**, **JANUS-L**, and **JANUS-P**, respectively.

Metrics In addition to the raw scores, we define three aggregated metrics to comprehensively evaluate the stability-plasticity tradeoff: AP_1 for stability, AP_2 for plasticity, and AP for overall performance. Specifically, AP_1 is the average of the performance scores across knowledge datasets, each normalized relative to the pretrained model. AP_2 is the average of the performance scores across task-specific datasets, each normalized relative to the FF model. The overall metric AP is the arithmetic mean of AP_1 and AP_2.

Further details are provided in the Appendix C.

5.2 Main results

Tabs. 2 and 3, together with Fig. 3, compare the performance of various methods. To demonstrate compatibility with different FT methods, we highlight **JANUS-P** on LLaMA-2-7b and **JANUS-L** on LLaMA-3-8b (full results in Appendix D). To enhance reliability, we evaluate PiSSA and JANUS-P on LLaMA-2-7b, as well as LoRA and JANUS-L on LLaMA-3-8b, across three random seeds (233, 234, and 235). All results presented without error bars (including those in ablation studies) are obtained using seed 233.

Across both models, standard PEFT methods (LoRA, PiSSA) exhibit high plasticity but suffer from severe forgetting, while knowledge-preserving methods provide varying tradeoffs between stability and plasticity. JANUS substantially improves this balance by recovering historical knowledge while largely retaining the adaptation achieved during FT. On LLaMA-2-7b, JANUS-P raises AP_1 from 67.55–79.71% to 100.27–109.83% across the three tasks while largely retaining new-task performance. On LLaMA-3-8b, JANUS-L raises AP_1 from 57.02–75.96% to 87.07–91.23%, while retaining over 98% of LoRA's AP_2 on every task.

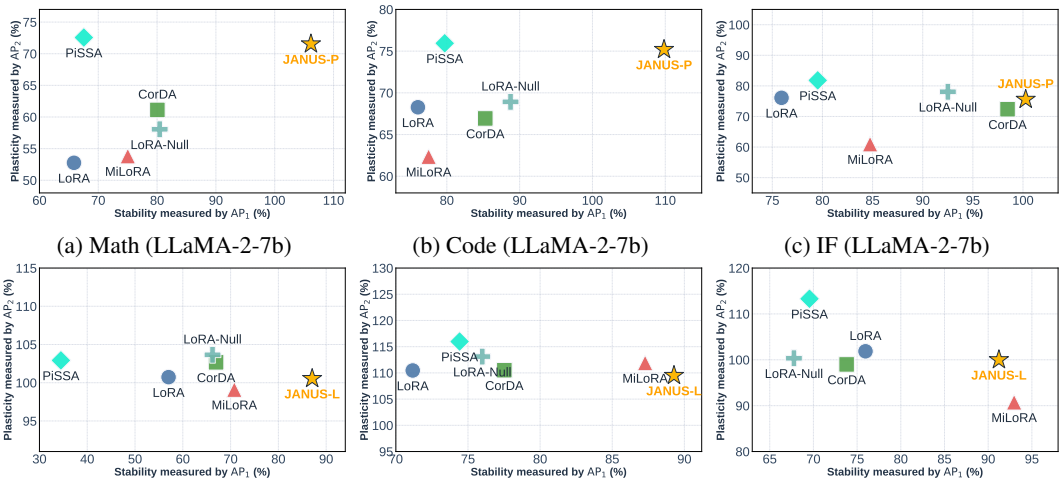
When JANUS offers only a limited advantage over knowledge-preserving methods on new tasks, this largely reflects the performance of the underlying FT method. For example, JANUS-L rectifies LoRA-fine-tuned models, which themselves do not outperform knowledge-preserving methods on several new-task benchmarks. Relative to LoRA, however, JANUS-L substantially improves historical knowledge retention while preserving nearly all new-task performance. As shown in Fig. 4,

Table 2: Performance comparison of various methods on LLaMA-2-7b. For each column, the **bold** and underlined values indicate the highest and second-highest scores, respectively. The scores used for normalization are marked in gray and excluded from the ranking.

Method	#Param	TriviaQA	NQ open	WebQS	$AP_1(\%)$	Bench 1	Bench 2	$AP_2(\%)$	$AP(\%)$
LLaMA-2-7b	–	52.52	18.95	5.81	100.00	–	–	–	–
Task: Math (Bench 1: GSM8k, Bench 2: Math)									
FF	6.7B	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
LoRA	320M	41.78	1.50	6.40	65.87	40.71	4.76	52.76	59.32
PiSSA	320M	41.57±1.50	3.34±0.18	6.15±0.52	67.55±3.85	**51.73**±0.29	**7.43**±0.35	**72.56**±1.37	70.05±2.34
CorDA	320M	41.94	7.09	7.14	80.05	43.97	6.18	61.12	70.59
MiLoRA	320M	44.98	3.13	7.14	75.02	40.64	5.04	53.82	64.42
LoRA-Null	320M	44.64	6.04	7.23	80.44	42.30	5.76	58.06	69.25
JANUS-P	320M	**47.31**±0.57	**16.05**±0.65	**8.35**±0.61	**106.16**±4.98	51.20±0.57	7.29±0.29	71.54±1.49	**88.85**±3.01
Task: Code (Bench 1: HumanEval, Bench 2: MBPP)									
FF	6.7B	37.61	6.76	6.59	73.57	33.88	28.41	100.00	86.78
LoRA	320M	43.52	9.22	5.61	76.03	18.22	23.51	68.27	72.15
PiSSA	320M	44.93±0.25	10.43±0.65	5.73±0.18	79.71±2.20	**21.51**±0.51	**25.11**±0.30	**75.94**±1.24	77.83±1.61
CorDA	320M	44.23	13.57	5.81	85.28	19.25	21.89	66.93	76.10
MiLoRA	320M	42.58	11.83	5.17	77.50	16.74	21.39	62.35	69.92
LoRA-Null	320M	46.03	14.57	5.91	88.75	18.59	23.57	68.92	78.83
JANUS-P	320M	**50.21**±0.45	**16.39**±0.43	**8.56**±0.31	**109.83**±2.51	21.01±0.88	25.10±0.48	75.19±0.80	**92.51**±0.98
Task: IF (Bench 1: MTBench, Bench 2: –)									
FF	6.7B	20.38	5.01	4.97	50.26	4.56	–	100.00	75.13
LoRA	320M	43.79	8.25	5.86	75.92	3.47	–	76.10	76.01
PiSSA	320M	43.23±0.70	8.75±0.45	6.40±0.32	79.53±2.88	**3.73**±0.54	–	**81.80**±11.94	80.67±6.34
CorDA	320M	45.63	**17.04**	6.89	98.46	3.30	–	72.37	85.42
MiLoRA	320M	45.02	10.28	6.64	84.75	2.78	–	60.96	72.86
LoRA-Null	320M	47.55	12.96	6.89	92.51	3.56	–	78.07	85.29
JANUS-P	320M	**48.06**±0.20	15.58±0.82	**7.38**±0.25	**100.27**±1.18	3.45±0.25	–	75.58±5.59	**87.93**±2.26

Table 3: Performance comparison of various methods on LLaMA-3-8b. For each column, the **bold** and underlined values indicate the highest and second-highest scores, respectively. The scores used for normalization are marked in gray and excluded from the ranking.

Method	#Param	TriviaQA	NQ open	WebQS	$AP_1(\%)$	Bench 1	Bench 2	$AP_2(\%)$	$AP(\%)$
LLaMA-3-8b	–	61.66	21.86	9.94	100.00	–	–	–	–
Task: Math (Bench 1: GSM8k, Bench 2: Math)									
FF	8.37B	41.43	5.21	4.13	44.19	75.36	24.84	100.00	72.10
LoRA	336M	54.66±0.58	7.90±0.78	4.60±0.10	57.02±1.33	75.49±0.57	25.16±0.33	100.73±0.41	78.87±0.65
PiSSA	336M	30.22	3.57	3.79	34.49	**76.72**	25.84	102.92	68.70
CorDA	336M	49.76	16.09	4.63	66.96	76.35	25.84	102.67	84.82
MiLoRA	336M	54.15	15.32	5.41	70.78	75.28	24.42	99.10	84.94
LoRA-Null	336M	53.93	11.86	5.66	66.22	76.50	**26.28**	**103.65**	84.94
JANUS-L	336M	**58.49**±0.77	**20.53**±0.11	**7.20**±0.63	**87.07**±2.67	75.44±0.92	25.08±0.16	100.54±0.37	**93.80**±1.32
Task: Code (Bench 1: HumanEval, Bench 2: MBPP)									
FF	8.37B	56.29	14.02	6.89	74.91	43.11	45.69	100.00	87.46
LoRA	336M	58.54±0.33	13.30±0.68	5.74±0.13	71.17±0.97	50.05±0.97	47.89±0.51	110.46±0.89	90.82±0.90
PiSSA	336M	54.43	15.32	6.45	74.42	**56.52**	46.08	**115.98**	95.20
CorDA	336M	56.26	17.37	6.15	77.52	51.99	45.88	110.51	94.02
MiLoRA	336M	60.28	18.12	**8.07**	87.28	50.24	**49.01**	111.90	**99.59**
LoRA-Null	336M	56.96	15.15	6.59	75.99	53.44	46.72	113.11	94.55
JANUS-L	336M	**60.69**±0.13	**19.93**±0.18	7.78±0.15	**89.28**±0.27	49.66±1.13	47.43±0.40	109.50±0.98	99.39±0.50
Task: IF (Bench 1: MTBench, Bench 2: –)									
FF	8.37B	42.69	10.08	3.94	51.66	5.94	–	100.00	75.83
LoRA	336M	58.78±0.62	15.38±0.24	6.18±0.25	75.96±0.93	6.05±0.21	–	101.85±3.50	88.91±1.55
PiSSA	336M	48.40	14.71	6.25	69.55	**6.73**	–	**113.30**	91.43
CorDA	336M	53.18	19.25	4.68	73.80	5.88	–	98.99	86.39
MiLoRA	336M	**62.49**	17.92	**9.50**	**92.97**	5.39	–	90.74	91.85
LoRA-Null	336M	53.42	15.32	4.63	67.77	5.96	–	100.34	84.05
JANUS-L	336M	61.47±0.39	**20.43**±0.29	8.00±0.62	91.23±2.47	5.94±0.06	–	100.00±1.05	**95.61**±0.71

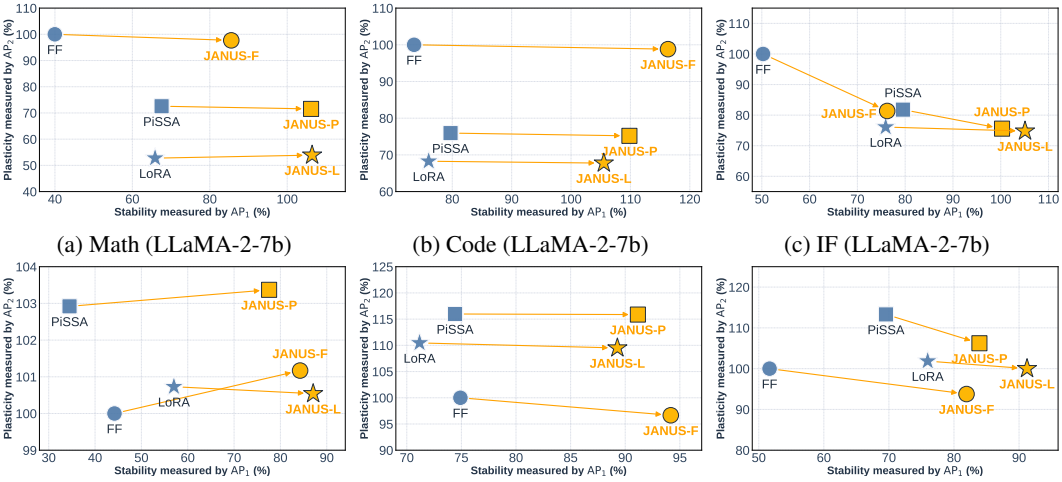


(a) Math (LLaMA-2-7b) (b) Code (LLaMA-2-7b) (c) IF (LLaMA-2-7b)

(d) Math (LLaMA-3-8b) (e) Code (LLaMA-3-8b) (f) IF (LLaMA-3-8b)

Figure 3: Stability-Plasticity comparison. The methods used for normalization are excluded.

this improvement over the corresponding FT checkpoint extends across all evaluated FT methods, downstream tasks, and base models, supporting JANUS as an effective plug-and-play rectification module. Notably, all rectifications use the same acceptance threshold $\tau = 0.95$ and decay factor $\beta = 0.7$, without task-specific tuning.



(a) Math (LLaMA-2-7b) (b) Code (LLaMA-2-7b) (c) IF (LLaMA-2-7b)

(d) Math (LLaMA-3-8b) (e) Code (LLaMA-3-8b) (f) IF (LLaMA-3-8b)

Figure 4: Recovery in stability across three FT methods.

5.3 Ablation

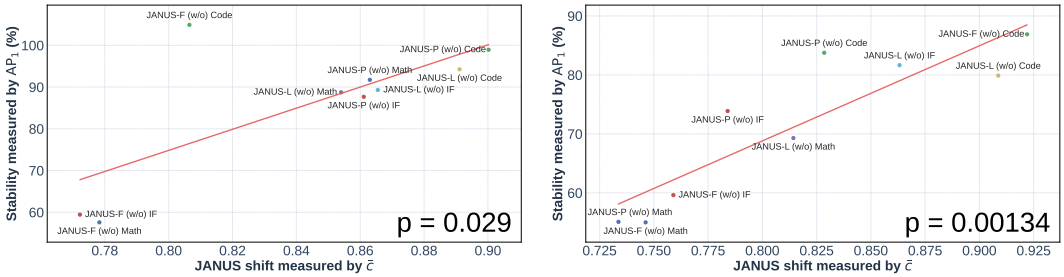
To evaluate the multi-step adaptive rectification, we ablate the mechanism by setting τ to zero, reducing it to single-step rectification (denoted as w/o). As demonstrated in Tab. 4, bypassing this mechanism drastically degrades stability across all tasks. We also plot the recovered stability AP_1 after a single-step rectification and the corresponding JANUS shift $\bar{c}$ in Fig. 5. Crucially, it reveals a statistically significant positive correlation ($p < 0.05$) between AP_1 and $\bar{c}$. This finding provides a solid foundation for our Multi-step Adaptive Rectification mechanism: by monitoring the JANUS shift, we can dynamically determine a proper step size to maximize stability recovery. Fig. 2b provides a visualization of the complete multi-step rectification process for JANUS-F on LLaMA-3-8b and the Math task, using an acceptance threshold of $\tau = 0.96$ and a decay factor of $\beta = 0.8$. It illustrates how

our method navigates the parameter space along the loss contour, progressively advancing toward the region of high plasticity. Complete results can be found in Appendix D.

For more ablations on the acceptance threshold τ, decay factor β, compressed rank r, replay size m, and replay distribution, please refer to Appendix E.

Table 4: Ablation study on the multi-step adaptive rectification mechanism.

Dataset	LLaMA-2-7b						LLaMA-3-8b					
	Math		Code		IF		Math		Code		IF	
	JANUS-P (w/)	JANUS-P (w/o)	JANUS-P (w/)	JANUS-P (w/o)	JANUS-P (w/)	JANUS-P (w/o)	JANUS-L (w/)	JANUS-L (w/o)	JANUS-L (w/)	JANUS-L (w/o)	JANUS-L (w/)	JANUS-L (w/o)
TriviaQA	**47.95**	45.98	**50.60**	49.53	**48.28**	46.81	**59.38**	56.87	**60.78**	59.65	**61.91**	60.85
NQ open	**16.73**	11.16	**16.26**	13.16	**14.82**	10.97	**20.64**	13.38	**19.94**	17.06	**20.66**	17.15
WebQS	**9.01**	7.48	**8.32**	7.73	**7.38**	6.74	**7.92**	5.41	**7.78**	6.45	**8.66**	6.74
AP_1(%)	**111.55**	91.73	**108.45**	98.93	**99.05**	87.67	**90.13**	69.29	**89.35**	79.89	**94.01**	81.65



(a) LLaMA-2-7b (b) LLaMA-3-8b

Figure 5: Correlation between the average cosine of principal angles and stability.

6 Conclusion and Limitation

In this paper, we introduced JANUS, a purely post-hoc and tuning-agnostic weight rectification framework designed to mitigate catastrophic forgetting in large-scale models. Grounded in a theoretical paradigm shift from subspace orthogonality to parameter space orthogonality, JANUS effectively recovers compromised knowledge by projecting parameter updates onto the historical Jacobian null space. We further introduced JANUS shift, defined as the average cosine of principal angles between Jacobian row spaces, which effectively quantifies the extent of forgetting and determines when it is necessary to recompute the JANUS. Leveraging this metric, we developed a Multi-step Adaptive Rectification mechanism which effectively overcomes the locality of Jacobian approximation. Powered by our proposed ghost operations and SVD compression, the rectification process is both temporally and spatially efficient. Empirical evaluations confirm that JANUS serves as a highly efficient, plug-and-play module for diverse FT methods, significantly mitigating the stability-plasticity dilemma by recovering historical knowledge while preserving downstream task adaptation.

The theoretical guarantee of all orthogonality-based methods relies on local first-order approximations. Although our multi-step mechanism mitigates this by monitoring trust regions, navigating highly rugged loss landscapes under extreme domain shifts may necessitate numerous conservative, small step sizes, which inherently extends the overall processing time. Additionally, while our current study focuses on the pretraining-to-finetuning setting using LLaMA-family models, extending JANUS to multi-task continual learning and broader architectures remains a promising direction for future work.

7 Acknowledgments and Disclosures

This study is supported by the National Natural Science Foundation of China under Grant No. 92582205 and the Beijing Natural Science Foundation (L257002). The authors declare no competing interests.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[2] Shengbo Eben Li. Reinforcement learning for sequential decision and optimal control. 2023.

[3] Yang Guan, Shengbo Eben Li, Jingliang Duan, Jie Li, Yangang Ren, Qi Sun, and Bo Cheng. Direct and indirect reinforcement learning. *International Journal of Intelligent Systems*, 36(8):4439–4467, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[5] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[6] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

[7] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[8] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2023.

[9] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

[11] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[12] Jingyang Qiao, Zhizhong Zhang, Xin Tan, Yanyun Qu, Wensheng Zhang, Zhi Han, and Yuan Xie. Gradient projection for continual parameter-efficient tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[13] Mao-Lin Luo, Zi-Hao Zhou, Yi-Lin Zhang, Yuanyu Wan, Tong Wei, and Min-Ling Zhang. Keeplora: Continual learning with residual gradient adaptation. *arXiv preprint arXiv:2601.19659*, 2026.

[14] Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen L Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. Corda: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:71768–71791, 2024.

[15] Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4823–4836, 2025.

[16] Pengwei Tang, Xiaolin Hu, Yong Liu, Lizhong Ding, Dongjie Zhang, Xing Wu, and Debing Zhang. Put the space of lora initialization to the extreme to preserve pre-trained knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33232–33240, 2026.

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

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

[19] Minrui Luo, Fuhang Kuang, Yu Wang, Zirui Liu, and Tianxing He. Sc-lora: Balancing efficient fine-tuning and knowledge preservation via subspace-constrained lora. *arXiv preprint arXiv:2505.23724*, 2025.

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

[21] Kefan Cao and Shuaicheng Wu. Orthogonal low-rank adaptation in lie groups for continual learning of large language models. *arXiv preprint arXiv:2509.06100*, 2025.

[22] 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.

[23] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, 2022.

[24] Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, 2020.

[25] Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

[26] Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, 2022.

[27] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 12799–12807, 2023.

[28] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[29] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, 2020.

[30] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328, 2021.

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

[32] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR, 2023.

[33] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

[34] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

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

[36] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

[37] Xialei Liu, Marc Masana, Luis Herranz, Joost van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *Proceedings-International Conference on Pattern Recognition*, 2018.

[38] Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. Afec: Active forgetting of negative transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22379–22391, 2021.

[39] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[40] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 9634–9643, 2021.

[41] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[42] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pages 3762–3773. PMLR, 2020.

[43] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

[44] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.

[45] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018.

[46] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[47] Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

[48] Camille Jordan. Essai sur la géométrie à n dimensions. *Bulletin de la Société mathématique de France*, 3:103–174, 1875.

[49] Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021.

[50] Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024.

[51] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE international symposium on high-performance computer architecture (HPCA)*, pages 97–110. IEEE, 2021.

[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] AI@Meta. Llama 3 model card. 2024.

[54] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[55] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 6086–6096, 2019.

[56] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[57] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[58] 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.

[59] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[60] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[61] Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12834–12859, 2024.

A Further Theoretical Analysis

A.1 Proof of Theorem 1

Theorem 1 (Global Forgetting-Free Guarantee via Layer-wise Projection)**.** *Applying the post-hoc JANUS projection independently to each linear layer guarantees that the overall loss change on the replayed past task samples approximates zero to the first-order, i.e.,* $\Delta\mathcal{L}_A^{(i)} \approx 0, \forall i = 1, 2, ..., m.$

Proof. Considering the parameter perturbations across all L layers simultaneously, the total change in loss for the i-th data point ($1 \leq i \leq m$) can be approximated as:

$$\begin{aligned}
\Delta\mathcal{L}_A^{(i)} &\approx \begin{bmatrix} \boldsymbol{g}^{(i,1)\top} & \boldsymbol{g}^{(i,2)\top} & \dots & \boldsymbol{g}^{(i,L)\top} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\theta}_{\text{proj}}^{(1)} \\ \Delta\boldsymbol{\theta}_{\text{proj}}^{(2)} \\ \vdots \\ \Delta\boldsymbol{\theta}_{\text{proj}}^{(L)} \end{bmatrix} \\
&= \sum_{j=1}^{L} \boldsymbol{g}^{(i,j)\top} \Delta\boldsymbol{\theta}_{\text{proj}}^{(j)} \\
&= 0. \qquad \text{(According to Eq. (7))}
\end{aligned}$$

This completes the proof. □

A.2 Beyond First-Order: A Gauss-Newton Perspective of JANUS

A natural question arises regarding our formulation: why should we project the parameter updates into the null space of the full Jacobian matrix J, which consists of individual gradients for each data point, rather than simply projecting them to be orthogonal to the average gradient? We seek to answer this question by formulating an equivalent least-squares problem to the original loss.

Let $\mathcal{L}_1(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_A^{(i)}(\boldsymbol{\theta})$ be the original loss, and let $\boldsymbol{\theta}^*$ denote the optimal solution of $\min_{\boldsymbol{\theta}} \mathcal{L}_1(\boldsymbol{\theta})$. A corresponding loss function in the least-squares form can be formulated as

$$\mathcal{L}_2(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m} \left(\mathcal{L}_A^{(i)}(\boldsymbol{\theta}) - \mathcal{L}_A^{(i)}(\boldsymbol{\theta}^*)\right)^2. \tag{8}$$

This formulation is equivalent to $\mathcal{L}_1$ in the sense that the optimal solution of $\min_{\boldsymbol{\theta}} \mathcal{L}_2(\boldsymbol{\theta})$ is also strictly $\boldsymbol{\theta}^*$, since $\mathcal{L}_2(\boldsymbol{\theta}^*) = 0$. Given that the initial parameter $\boldsymbol{\theta}_0$ is the outcome of pretraining on the past task $\mathcal{T}_A$, we can assume that $\mathcal{L}_A^{(i)}(\boldsymbol{\theta}_0) \approx \mathcal{L}_A^{(i)}(\boldsymbol{\theta}^*)$. Consequently, according to the Gauss-Newton approximation [60], the Hessian matrix of $\mathcal{L}_2(\boldsymbol{\theta})$ around $\boldsymbol{\theta}_0$ can be derived as

$$\begin{aligned}
\boldsymbol{H}_{\mathcal{L}_2}(\boldsymbol{\theta}) &= \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{g}^{(i)}(\boldsymbol{\theta})\, \boldsymbol{g}^{(i)}(\boldsymbol{\theta})^\top + \frac{1}{m}\sum_{i=1}^{m} \left(\mathcal{L}_A^{(i)}(\boldsymbol{\theta}) - \mathcal{L}_A^{(i)}(\boldsymbol{\theta}^*)\right) \boldsymbol{H}_{\mathcal{L}_A^{(i)}}(\boldsymbol{\theta}) \\
&\approx \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{g}^{(i)}(\boldsymbol{\theta})\, \boldsymbol{g}^{(i)}(\boldsymbol{\theta})^\top \\
&= \frac{1}{m}\boldsymbol{J}^\top\boldsymbol{J},
\end{aligned} \tag{9}$$

where $\boldsymbol{g}^{(i)} = \left[\boldsymbol{g}^{(i,1)\top}, \boldsymbol{g}^{(i,2)\top}, \dots, \boldsymbol{g}^{(i,L)\top}\right]^\top \in \mathbb{R}^n$ and $\boldsymbol{J} = \left[\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)}, \dots, \boldsymbol{g}^{(m)}\right]^\top \in \mathbb{R}^{m\times n}$ are the global gradient and Jacobian, respectively.

Eq. (9) indicates that the null space of the Jacobian J closely approximates that of the equivalent Hessian $\boldsymbol{H}_{\mathcal{L}_2}$. Therefore, by rectifying the parameter update $\Delta\boldsymbol{\theta}$ into $\mathcal{N}(\boldsymbol{J})$, we effectively constrain the update to move along the flat loss valleys of the equivalent problem. While directly computing the true Hessian $\boldsymbol{H}_{\mathcal{L}_1}(\boldsymbol{\theta})$ incurs a prohibitive $O\left(n^2\right)$ memory complexity for large models, our Jacobian projection requires only first-order computation and memory. In essence, it is derived *from* first-order information, yet operates *beyond* first-order limitations.

The ablation study in Tab. 5 highlights the necessity of using Jacobian matrices. When simplified to average gradients, the stability of JANUS-P collapses to the level of PiSSA, confirming that individual gradient information is critical for precise weight rectification.

Table 5: Ablation on the use of Jacobian matrices instead of average gradient.

Method	#Param	TriviaQA	NQ open	WebQS	AP_1(%)	GSM8k	Math	AP_2(%)	AP(%)
LLaMA-2-7b	–	52.52	18.95	5.81	100.00	–	–	–	–
FF	6.7B	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
PiSSA	320M	42.92	3.21	6.69	71.27	51.86	7.76	73.96	72.62
JANUS-P	320M	47.95	16.73	9.01	111.55	51.86	7.52	73.01	92.28
JANUS-P (avg. grad)	320M	42.92	3.27	6.59	70.80	51.78	7.54	73.02	71.91

B Details of Efficient Calculation and Storage of the Jacobian

B.1 Ghost Projection

Given a batch of m data points, the input of the j-th linear layer can be formulated as a tensor $\boldsymbol{X}^{(j)} = \left[\boldsymbol{x}_{i,k}^{(j)}\right] \in \mathbb{R}^{m\times T\times d_{\text{in}}}$, where $\boldsymbol{x}_{i,k}^{(j)} \in \mathbb{R}^{d_{\text{in}}}$ denotes the input vector corresponding to the k-th token of the i-th sample. Similarly, we denote the corresponding output matrix of this layer as $\boldsymbol{Y}^{(j)} = \left[\boldsymbol{y}_{i,k}^{(j)}\right] \in \mathbb{R}^{m\times T\times d_{\text{out}}}$, comprising the output vectors $\boldsymbol{y}_{i,k}^{(j)} \in \mathbb{R}^{d_{\text{out}}}$. According to the chain rule and the typical loss formulations in LLMs, the gradient of the individual sample loss $\mathcal{L}_A^{(i)}$ with respect to the weight $\boldsymbol{\theta}^{(j)}$ can be computed as:

$$\begin{aligned}
\boldsymbol{g}^{(i,j)} &= \text{rvec}\left(\nabla_{\boldsymbol{W}^{(j)}}\mathcal{L}_A^{(i)}\right) \\
&= \text{rvec}\left(\sum_{k=1}^{T}\nabla_{\boldsymbol{y}_{i,k}^{(j)}}\mathcal{L}_A^{(i)}\left(\boldsymbol{x}_{i,k}^{(j)}\right)^\top\right) \\
&= \text{rvec}\left(\sum_{k=1}^{T}\nabla_{\boldsymbol{y}_{i,k}^{(j)}}\mathcal{L}_1\left(\boldsymbol{x}_{i,k}^{(j)}\right)^\top\right) \\
&\triangleq \text{rvec}\left(\sum_{k=1}^{T}\boldsymbol{a}_{i,k}^{(j)}\left(\boldsymbol{x}_{i,k}^{(j)}\right)^\top\right),
\end{aligned} \tag{10}$$

where $\mathcal{L}_1(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_A^{(i)}(\boldsymbol{\theta})$ is the batch-level loss, and the last step holds because $\boldsymbol{y}_{i,k}^{(j)}$ has no dependency on the losses of the other data points.

In standard LLM frameworks, the default output loss is typically averaged over all valid (non-padded) tokens. Directly backpropagating this averaged loss would scale the gradients down by a factor of $1/T_{\text{valid}}$, causing the magnitude of the resulting Jacobian matrix to fluctuate arbitrarily depending on the varying sequence lengths of different batches. To strictly adhere to our mathematical derivation which requires the additive accumulation of token-wise gradients, we explicitly "un-reduce" the loss by multiplying it by the total number of valid tokens before backpropagation. By performing a single backward pass with respect to this unreduced loss, we can efficiently obtain the exact pre-activation gradients $\boldsymbol{A}^{(j)} = \left[\boldsymbol{a}_{i,k}^{(j)}\right] \in \mathbb{R}^{m\times T\times d_{\text{out}}}$ along with the layer inputs $\boldsymbol{X}^{(j)}$.

These two tensors jointly enable the exact construction of the per-sample gradients, which conceptually form the full Jacobian matrix:

$$\boldsymbol{J}^{(j)} = \begin{bmatrix}\boldsymbol{g}^{(1,j)\top}\\ \boldsymbol{g}^{(2,j)\top}\\ \vdots \\ \boldsymbol{g}^{(m,j)\top}\end{bmatrix} = \begin{bmatrix}\text{rvec}\left(\sum_{k=1}^{T}\boldsymbol{a}_{1,k}^{(j)}\left(\boldsymbol{x}_{1,k}^{(j)}\right)^\top\right)^\top \\ \text{rvec}\left(\sum_{k=1}^{T}\boldsymbol{a}_{2,k}^{(j)}\left(\boldsymbol{x}_{2,k}^{(j)}\right)^\top\right)^\top \\ \vdots \\ \text{rvec}\left(\sum_{k=1}^{T}\boldsymbol{a}_{m,k}^{(j)}\left(\boldsymbol{x}_{m,k}^{(j)}\right)^\top\right)^\top\end{bmatrix}. \tag{11}$$

Crucially, all necessary computations for the JANUS projection and orientation comparison can be executed directly utilizing the tensors $\boldsymbol{X}^{(j)}$ and $\boldsymbol{A}^{(j)}$, without the need to instantiate and store the full Jacobian at all.

Specifically, the JANUS projection computes the parameter update as follows:

$$\Delta\boldsymbol{\theta}_{\text{proj}}^{(j)} = \Delta\boldsymbol{\theta}^{(j)} - \boldsymbol{J}^{(j)\top}\underbrace{\left(\boldsymbol{J}^{(j)}\boldsymbol{J}^{(j)\top}\right)^{-1}\boldsymbol{J}^{(j)}\Delta\boldsymbol{\theta}^{(j)}}_{\boldsymbol{v}^{(j)}}. \tag{12}$$

This relies on three core operations: the evaluation of the Gram matrix $\boldsymbol{J}^{(j)}\boldsymbol{J}^{(j)\top} \in \mathbb{R}^{m\times m}$, the Jacobian-vector product (JVP) $\boldsymbol{J}^{(j)}\Delta\boldsymbol{\theta}^{(j)} \in \mathbb{R}^{m}$, and the (transposed) vector-Jacobian product (VJP) $\boldsymbol{J}^{(j)\top}\boldsymbol{v}^{(j)} \in \mathbb{R}^{d_{\text{in}}d_{\text{out}}}$.

First, the (u, v)-th entry of the Gram matrix can be decoupled into the Hadamard product of two smaller dot products:

$$\left(\boldsymbol{J}^{(j)}\boldsymbol{J}^{(j)\top}\right)_{u,v} = \boldsymbol{g}^{(u,j)\top}\boldsymbol{g}^{(v,j)} = \sum_{k=1}^{T}\sum_{k'=1}^{T}\left(\left(\boldsymbol{a}_{u,k}^{(j)}\right)^{\top}\boldsymbol{a}_{v,k'}^{(j)}\right)\left(\left(\boldsymbol{x}_{u,k}^{(j)}\right)^{\top}\boldsymbol{x}_{v,k'}^{(j)}\right). \tag{13}$$

For improved numerical precision, we compute the inverse of the Gram matrix in `float64` precision. Second, the u-th element of the JVP vector can be efficiently computed via vector-matrix-vector multiplication:

$$\left(\boldsymbol{J}^{(j)}\Delta\boldsymbol{\theta}^{(j)}\right)_{u} = \boldsymbol{g}^{(u,j)\top}\Delta\boldsymbol{\theta}^{(j)} = \sum_{k=1}^{T}\left(\boldsymbol{a}_{u,k}^{(j)}\right)^{\top}\Delta\boldsymbol{W}^{(j)}\boldsymbol{x}_{u,k}^{(j)}. \tag{14}$$

Finally, given the intermediate vector $\boldsymbol{v}^{(j)} = \left[v_1^{(j)}, \ldots, v_m^{(j)}\right]^{\top} \in \mathbb{R}^m$, the VJP vector is inherently a linear combination of the per-sample gradient vectors. Let $\boldsymbol{P}^{(j)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the matrix form of the VJP vector. It can be computed as

$$\boldsymbol{P}^{(j)} = \sum_{i=1}^{m} v_i^{(j)} \sum_{k=1}^{T} \boldsymbol{a}_{i,k}^{(j)}\left(\boldsymbol{x}_{i,k}^{(j)}\right)^{\top}. \tag{15}$$

In practice, these formulations are highly parallelizable and can be strictly reduced to hardware-efficient tensor contractions (e.g., `torch.einsum`).

B.2 Ghost Orientation Comparison

The JANUS orientation comparison quantifies the orientation shift between the current and trial Jacobian matrices, denoted as $\boldsymbol{J}_1^{(j)}$ and $\boldsymbol{J}_2^{(j)}$, by computing the mean cosine of the principal angles between their respective row spaces. Let $\boldsymbol{G}_1^{(j)} = \boldsymbol{J}_1^{(j)}\boldsymbol{J}_1^{(j)\top}$ and $\boldsymbol{G}_2^{(j)} = \boldsymbol{J}_2^{(j)}\boldsymbol{J}_2^{(j)\top}$ denote the individual Gram matrices, and $\boldsymbol{G}_{12}^{(j)} = \boldsymbol{J}_1^{(j)}\boldsymbol{J}_2^{(j)\top}$ be the cross-Gram matrix. All these matrices can be evaluated efficiently without instantiating the Jacobians, as derived in Eq. (13).

Next, letting $\boldsymbol{G}_1^{(j)} = \boldsymbol{L}_1^{(j)}\boldsymbol{\Lambda}_1^{(j)}\boldsymbol{L}_1^{(j)\top}$ and $\boldsymbol{G}_2^{(j)} = \boldsymbol{L}_2^{(j)}\boldsymbol{\Lambda}_2^{(j)}\boldsymbol{L}_2^{(j)\top}$ represent the eigendecompositions of the individual Gram matrices, the cosine of the i-th principal angle corresponds to the i-th singular value of the interaction matrix:

$$\boldsymbol{M}^{(j)} = \left(\boldsymbol{\Lambda}_1^{(j)}\right)^{-1/2}\boldsymbol{L}_1^{(j)\top}\boldsymbol{G}_{12}^{(j)}\boldsymbol{L}_2^{(j)}\left(\boldsymbol{\Lambda}_2^{(j)}\right)^{-1/2}. \tag{16}$$

Since $\boldsymbol{G}_1^{(j)}$, $\boldsymbol{G}_2^{(j)}$, $\boldsymbol{G}_{12}^{(j)}$, and $\boldsymbol{M}^{(j)}$ are all low-dimensional matrices of size $m \times m$, performing these eigendecompositions and the subsequent SVD incurs negligible computational overhead.

B.3 Sequence-level SVD Compression

Caching the dense tensors $\boldsymbol{X}^{(j)}$ and $\boldsymbol{A}^{(j)}$ for all layers across a large sequence length T can still pose a substantial memory burden. However, leveraging the intrinsic low-rank property of $\boldsymbol{X}^{(j)}$ and $\boldsymbol{A}^{(j)}$, we can significantly compress these tensors along the sequence dimension using rank-r Truncated SVD.

Specifically, let $\boldsymbol{X}_i^{(j)} \in \mathbb{R}^{T \times d_{\text{in}}}$ and $\boldsymbol{A}_i^{(j)} \in \mathbb{R}^{T \times d_{\text{out}}}$ be the layer inputs and pre-activation gradients for the i-th sample. We first perform QR decomposition on their transposes:

$$\boldsymbol{X}_i^{(j)\top} = \boldsymbol{Q}_X\boldsymbol{R}_X, \quad \boldsymbol{A}_i^{(j)\top} = \boldsymbol{Q}_A\boldsymbol{R}_A, \tag{17}$$

where $\boldsymbol{Q}_X, \boldsymbol{Q}_A$ are orthogonal matrices and $\boldsymbol{R}_X, \boldsymbol{R}_A$ are upper triangular matrices. The gradient for the i-th sample is therefore $\boldsymbol{g}^{(i,j)} = \text{rvec}\left(\boldsymbol{A}_i^{(j)\top}\boldsymbol{X}_i^{(j)}\right) = \text{rvec}\left(\boldsymbol{Q}_A\left(\boldsymbol{R}_A\boldsymbol{R}_X^{\top}\right)\boldsymbol{Q}_X^{\top}\right)$. By computing the rank-$r$ truncated SVD on the core matrix $\boldsymbol{M} = \boldsymbol{R}_A\boldsymbol{R}_X^{\top} \approx \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^{\top}$, we can

symmetrically distribute the singular values to form two compressed, low-dimensional tensors $\hat{\boldsymbol{X}}^{(j)} \in \mathbb{R}^{m\times r\times d_{\text{in}}}$ and $\hat{\boldsymbol{A}}^{(j)} \in \mathbb{R}^{m\times r\times d_{\text{out}}}$. For the i-th sample, the compressed slices are strictly defined as:

$$\hat{\boldsymbol{X}}_i^{(j)} = \left(\boldsymbol{Q}_X \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{1/2}\right)^\top, \quad \hat{\boldsymbol{A}}_i^{(j)} = \left(\boldsymbol{Q}_A \boldsymbol{U}_r \boldsymbol{\Sigma}_r^{1/2}\right)^\top. \tag{18}$$

By construction, their inner product directly recovers the optimal rank-r approximation of the original gradient matrix, successfully preserving the principal information along the sequence dimension:

$$\left(\hat{\boldsymbol{A}}_i^{(j)}\right)^\top \hat{\boldsymbol{X}}_i^{(j)} = \boldsymbol{Q}_A \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top \boldsymbol{Q}_X^\top \approx \boldsymbol{Q}_A \left(\boldsymbol{R}_A \boldsymbol{R}_X^\top\right) \boldsymbol{Q}_X^\top = \left(\boldsymbol{A}_i^{(j)}\right)^\top \boldsymbol{X}_i^{(j)}. \tag{19}$$

B.4 Time Cost Profiling

To evaluate computational efficiency, we break down the total execution time of the rectification process for JANUS-P on LLaMA-2-7b and the Math task into four key components: (1) forward and backward passes, (2) ghost projection and orientation comparison, (3) SVD compression, and (4) disk I/O. The results are listed in Tab. 6. The first thing worth noting is that forward and backward passes account for less than 5% of the total execution time. This efficiency is directly attributed to our ghost operations, which bypass the need for expensive per-sample backpropagation. On the other hand, disk I/O constitutes a significant portion of the overall time cost (28.19%). This is because in our current implementation, the compressed tensors $\hat{\boldsymbol{X}}^{(j)}$ and $\hat{\boldsymbol{A}}^{(j)}$ are temporarily cached to disk. However, this overhead could be substantially mitigated in future optimizations through parallel loading or memory offloading. Furthermore, while our current implementation employs exact SVD for tensor compression, adopting randomized approximate SVD may have the potential to significantly accelerate this stage while preserving performance.

Table 6: Time cost profile of JANUS-P on LLaMA-2-7b and the Math task.

Component	Time (s)	Percentage (%)
SVD compression	524.85	36.07
Disk I/O	410.29	28.19
Ghost projection and orientation comparison	351.60	24.16
Others	103.62	7.12
Forward and backward passes	64.84	4.46
Total	1455.20	100.00

C Implementation Details

C.1 Datasets and Benchmarks

Throughout our experiments, we utilize the training dataset released by PiSSA[1] [22]. This dataset comprises several subsets, including "metamath", "python", and "conversation", which correspond to the Math, Code, and IF tasks, respectively. Specifically, the training data for the Math task is sourced from MetaMathQA [5], the Code task from CodeFeedback [61], and the IF task from WizardLM-Evol-Instruct [8].

For evaluation, we employ publicly available benchmarks: TriviaQA, NQ open, and WebQS for world knowledge[2]; GSM8k and MATH for the Math task[1]; HumanEval and MBPP for the Code task[3]; and MTBench for the IF task[4].

C.2 Training Configuration

For LLaMA-2-7b, we follow the experimental setup established in [14, 16, 22]. Specifically, models are optimized using the AdamW optimizer with a batch size of 128, a maximum sequence length of 512, and a learning rate of 2×10^{-5} regulated by a cosine annealing schedule (warmup ratio of 0.03). No weight decay is applied.

On LLaMA-3-8b, we observed that a learning rate of 2×10^{-5} was insufficient to induce significant forgetting. To evaluate the robustness of our method under more substantial weight shifts, we increased the learning rate to 5×10^{-5} for all methods except FF. For FF on LLaMA-3-8b, the training process is highly unstable under large learning rates. Hence, we employed a learning rate of 5×10^{-6} for the Math and IF tasks, and 2×10^{-6} for the Code task.

All models are trained on the first 100,000 conversations of the dataset for one epoch, with the loss calculated exclusively on the response tokens. Training and evaluation are performed on 4 NVIDIA A100 80GB GPUs, while the rectification process is executed on a single NVIDIA A100 80GB GPU. We use `float16` precision for FF and `bfloat16` for the base weights of other FT methods, with adapter parameters maintained in `float32`.

C.3 Evaluation Configuration

We follow the default evaluation protocols for all benchmarks. For world knowledge and the Math task, no sensitive hyperparameters are involved in the evaluation process.

For the Code and IF tasks, we explicitly specify the standard settings used to ensure transparency. For the Code task, we strictly adhere to the officially recommended configurations[5]: specifically, a temperature of 0.2 with $n = 200$ samples for HumanEval, and a temperature of 0.1 with $n = 15$ samples for MBPP. All reported metrics for these benchmarks are `pass@1`.

For the IF task, GPT-4 is employed as the judge model. Although MTBench supports multi-turn dialogues, we report performance based only on the first turn, as the training data consists exclusively of single-turn conversations[1].

To facilitate reproduction, an evaluation script for each benchmark is provided in our repository.

C.4 Data Replay

For CorDA and LoRA-Null, the replay buffer consists of 256 samples randomly drawn from NQ open with a maximum length of 1024 tokens, consistent with their original implementations. It should be noted that since individual entries in NQ open are typically much shorter than 1024 tokens, each sample comprises multiple concatenated data points. For JANUS, we replay exactly 256 randomly selected data points from the same source.

[1] `https://huggingface.co/datasets/fxmeng/pissa-dataset`

[2] `https://github.com/EleutherAI/lm-evaluation-harness`

[3] `https://github.com/bigcode-project/bigcode-evaluation-harness`

[4] `https://github.com/lm-sys/FastChat`

[5] `https://github.com/bigcode-project/bigcode-evaluation-harness/blob/main/docs/README.md`

D Complete Main Results

Listed in Tab. 7 and Tab. 8 are the complete results on LLaMA-2-7b and LLaMA-3-8b, respectively. These results indicate that, despite being influenced by the baseline performance of the fine-tuned models, our JANUS rectification consistently achieves superior overall performance, significantly mitigating the stability-plasticity dilemma.

Tab. 9 provides the full results of the ablation study regarding the multi-step adaptive rectification mechanism. Across various base models, downstream tasks, and fine-tuning methods, this mechanism demonstrates a universal capability to enhance stability.

Table 7: Complete performance comparison of various methods on LLaMA-2-7b across Math, Code, and IF tasks. For each column within a task, **bold** and underlined values indicate the highest and second-highest scores among fine-tuned methods.

Method	#Param	TriviaQA	NQ open	WebQS	$AP_1(\%)$	Bench 1	Bench 2	$AP_2(\%)$	$AP(\%)$
LLaMA-2-7b	–	52.52	18.95	5.81	100.00	–	–	–	–
Task: Math (Bench 1: GSM8k, Bench 2: Math)									
FF	6.7B	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
LoRA	320M	41.78	1.50	6.40	65.87	40.71	4.76	52.76	59.32
PiSSA	320M	41.57±1.50	3.34±0.18	6.15±0.52	67.55±3.85	51.73±0.29	7.43±0.35	72.56±1.37	70.05±2.34
CorDA	320M	41.94	7.09	7.14	80.05	43.97	6.18	61.12	70.59
MiLoRA	320M	44.98	3.13	7.14	75.02	40.64	5.04	53.82	64.42
LoRA-Null	320M	44.64	6.04	7.23	80.44	42.30	5.76	58.06	69.25
JANUS-F	6.7B	35.57	9.97	7.92	85.55	**58.76**	**12.28**	**97.69**	**91.62**
JANUS-L	320M	**47.73**	**17.78**	7.82	**106.43**	40.94	5.00	53.91	80.17
JANUS-P	320M	47.31±0.57	16.05±0.65	**8.35**±0.61	106.16±4.98	51.20±0.57	7.29±0.29	71.54±1.49	88.85±3.01
Task: Code (Bench 1: HumanEval, Bench 2: MBPP)									
FF	6.7B	37.61	6.76	6.59	73.57	33.88	28.41	100.00	86.78
LoRA	320M	43.52	9.22	5.61	76.03	18.22	23.51	68.27	72.15
PiSSA	320M	44.93±0.25	10.43±0.65	5.73±0.18	79.71±2.20	21.51±0.51	25.11±0.30	75.94±1.24	77.83±1.61
CorDA	320M	44.23	13.57	5.81	85.28	19.25	21.89	66.93	76.10
MiLoRA	320M	42.58	11.83	5.17	77.50	16.74	21.39	62.35	69.92
LoRA-Null	320M	46.03	14.57	5.91	88.75	18.59	23.57	68.92	78.83
JANUS-F	6.7B	48.56	**16.68**	**9.79**	**116.33**	**33.05**	**28.43**	**98.81**	**107.57**
JANUS-L	320M	**50.40**	16.43	7.78	105.52	17.35	23.95	67.76	86.64
JANUS-P	320M	50.21±0.45	16.39±0.43	8.56±0.31	109.83±2.51	21.01±0.88	25.10±0.48	75.19±0.80	92.51±0.98
Task: IF (Bench 1: MTBench, Bench 2: –)									
FF	6.7B	20.38	5.01	4.97	50.26	4.56	–	100.00	75.13
LoRA	320M	43.79	8.25	5.86	75.92	3.47	–	76.10	76.01
PiSSA	320M	43.23±0.70	8.75±0.45	6.40±0.32	79.53±2.88	**3.73**±0.54	–	**81.80**±11.94	80.67±6.34
CorDA	320M	45.63	17.04	6.89	98.46	3.30	–	72.37	85.42
MiLoRA	320M	45.02	10.28	6.64	84.75	2.78	–	60.96	72.86
LoRA-Null	320M	47.55	12.96	6.89	92.51	3.56	–	78.07	85.29
JANUS-F	6.7B	32.78	10.64	6.40	76.24	3.71	–	81.36	78.80
JANUS-L	320M	**49.59**	**17.45**	**7.48**	**105.08**	3.41	–	74.78	**89.93**
JANUS-P	320M	48.06±0.20	15.58±0.82	7.38±0.25	100.27±1.18	3.45±0.25	–	75.58±5.59	87.93±2.26

Table 8: Complete performance comparison of various methods on LLaMA-3-8b across Math, Code, and IF tasks. For each column within a task, **bold** and underlined values indicate the highest and second-highest scores among fine-tuned methods.

Method	#Param	TriviaQA	NQ open	WebQS	$AP_1(\%)$	Bench 1	Bench 2	$AP_2(\%)$	$AP(\%)$
LLaMA-3-8b	–	61.66	21.86	9.94	100.00	–	–	–	–
Task: Math (Bench 1: GSM8k, Bench 2: Math)									
FF	8.37B	41.43	5.21	4.13	44.19	75.36	24.84	100.00	72.10
LoRA	336M	54.66±0.58	7.90±0.78	4.60±0.10	57.02±1.33	75.49±0.57	25.16±0.33	100.73±0.41	78.87±0.65
PiSSA	336M	30.22	3.57	3.79	34.49	76.72	25.84	102.92	68.70
CorDA	336M	49.76	16.09	4.63	66.96	76.35	25.84	102.67	84.82
MiLoRA	336M	54.15	15.32	5.41	70.78	75.28	24.42	99.10	84.94
LoRA-Null	336M	53.93	11.86	5.66	66.22	76.50	**26.28**	**103.65**	84.94
JANUS-F	8.37B	**59.21**	19.00	6.94	84.25	76.57	25.02	101.17	92.71
JANUS-L	336M	58.49±0.77	**20.53**±0.11	**7.20**±0.63	**87.07**±2.67	75.44±0.92	25.08±0.16	100.54±0.37	**93.80**±1.32
JANUS-P	336M	53.00	17.89	6.45	77.56	**77.71**	25.74	103.37	90.47
Task: Code (Bench 1: HumanEval, Bench 2: MBPP)									
FF	8.37B	56.29	14.02	6.89	74.91	43.11	45.69	100.00	87.46
LoRA	336M	58.54±0.33	13.30±0.68	5.74±0.13	71.17±0.97	50.05±0.97	47.89±0.51	110.46±0.89	90.82±0.90
PiSSA	336M	54.43	15.32	6.45	74.42	56.52	46.08	**115.98**	95.20
CorDA	336M	56.26	17.37	6.15	77.52	51.99	45.88	110.51	94.02
MiLoRA	336M	60.28	18.12	8.07	87.28	50.24	**49.01**	111.90	99.59
LoRA-Null	336M	56.96	15.15	6.59	75.99	53.44	46.72	113.11	94.55
JANUS-F	8.37B	59.25	**20.30**	**9.30**	**94.17**	41.22	44.64	96.66	95.42
JANUS-L	336M	**60.69**±0.13	19.93±0.18	7.78±0.15	89.28±0.27	49.66±1.13	47.43±0.40	109.50±0.98	99.39±0.50
JANUS-P	336M	57.42	19.94	8.86	91.16	**56.87**	45.61	115.87	**103.52**
Task: IF (Bench 1: MTBench, Bench 2: –)									
FF	8.37B	42.69	10.08	3.94	51.66	5.94	–	100.00	75.83
LoRA	336M	58.78±0.62	15.38±0.24	6.18±0.25	75.96±0.93	6.05±0.21	–	101.85±3.50	88.91±1.55
PiSSA	336M	48.40	14.71	6.25	69.55	**6.73**	–	**113.30**	91.43
CorDA	336M	53.18	19.25	4.68	73.80	5.88	–	98.99	86.39
MiLoRA	336M	**62.49**	17.92	**9.50**	**92.97**	5.39	–	90.74	91.85
LoRA-Null	336M	53.42	15.32	4.63	67.77	5.96	–	100.34	84.05
JANUS-F	8.37B	56.21	20.17	6.20	81.93	5.57	–	93.77	87.85
JANUS-L	336M	61.47±0.39	**20.43**±0.29	8.00±0.62	91.23±2.47	5.94±0.06	–	100.00±1.05	**95.61**±0.71
JANUS-P	336M	55.30	18.75	7.58	83.91	6.31	–	106.23	95.07

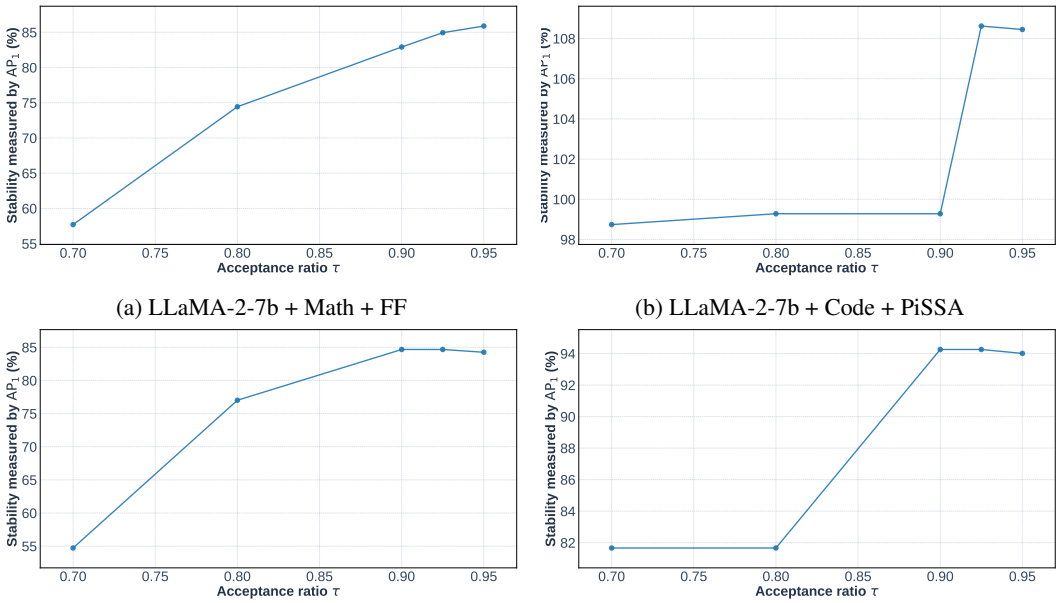
Table 9: Complete results of the ablation study on the multi-step adaptive rectification mechanism.

Methods	LLaMA-2-7b				LLaMA-3-8b			
	TriviaQA	NQ open	WebQS	AP_1(%)	TriviaQA	NQ open	WebQS	AP_1(%)
Task: Math								
JANUS-F (w/)	**35.57**	**9.97**	**7.92**	**85.55**	**59.21**	**19.00**	**6.94**	**84.25**
JANUS-F (w/o)	28.01	4.49	5.56	57.57	50.72	8.34	4.43	54.99
JANUS-L (w/)	**47.73**	**17.78**	**7.82**	**106.43**	**59.38**	**20.64**	**7.92**	**90.13**
JANUS-L (w/o)	45.24	11.00	7.09	88.74	56.87	13.38	5.41	69.29
JANUS-P (w/)	**47.95**	**16.73**	**9.01**	**111.55**	**53.00**	**17.89**	**6.45**	**77.56**
JANUS-P (w/o)	45.98	11.16	7.48	91.73	45.04	9.86	4.68	55.08
Task: Code								
JANUS-F (w/)	**48.56**	**16.68**	**9.79**	**116.33**	**59.25**	**20.30**	**9.30**	**94.17**
JANUS-F (w/o)	46.50	12.85	9.20	104.90	58.30	18.12	8.27	86.88
JANUS-L (w/)	**50.40**	**16.43**	**7.78**	**105.52**	**60.78**	**19.94**	**7.78**	**89.35**
JANUS-L (w/o)	49.20	12.38	7.19	94.25	59.65	17.06	6.45	79.89
JANUS-P (w/)	**50.60**	**16.26**	**8.32**	**108.45**	**57.42**	**19.94**	**8.86**	**91.16**
JANUS-P (w/o)	49.53	13.16	7.73	98.93	56.25	19.09	7.23	83.76
Task: IF								
JANUS-F (w/)	**32.78**	**10.64**	**6.40**	**76.24**	**56.21**	**20.17**	**6.20**	**81.93**
JANUS-F (w/o)	27.78	6.12	5.41	59.43	49.33	12.63	4.08	59.61
JANUS-L (w/)	**49.59**	**17.45**	**7.48**	**105.08**	**61.91**	**20.66**	**8.66**	**94.01**
JANUS-L (w/o)	47.29	11.88	6.69	89.29	60.85	17.15	6.74	81.65
JANUS-P (w/)	**48.28**	**14.82**	**7.38**	**99.05**	**55.30**	**18.75**	**7.58**	**83.91**
JANUS-P (w/o)	46.81	10.97	6.74	87.67	50.77	16.81	6.20	73.87

E More ablations

E.1 Acceptance threshold τ and decay factor β

As illustrated in Fig. 5, a statistically significant positive correlation exists between the recovered stability and the JANUS shift. To further validate the effectiveness of JANUS shift as a metric of the trust region, we apply the JANUS rectification with various acceptance thresholds τ on a series of settings, including (1) LLaMA-2-7b + Math + FF, (2) LLaMA-2-7b + Code + PiSSA, (3) LLaMA-3-8b + Math + FF, and (4) LLaMA-3-8b + IF + LoRA, covering across two models, three tasks, and three FT methods. The performance comparison of these rectified models are summarized in Tab. 10 and Fig. 6. The results demonstrate that increasing τ effectively enhances stability while maintaining task-specific performance, which justifies adaptively choosing proper step size under the guide of the JANUS shift.



(a) LLaMA-2-7b + Math + FF

(b) LLaMA-2-7b + Code + PiSSA

(c) LLaMA-3-8b + Math + FF

(d) LLaMA-3-8b + IF + LoRA

Figure 6: Impact of the acceptance threshold τ on the stability of the rectified model.

Regarding the decay factor β, Tab. 11 demonstrates that our method is highly robust to variations in this parameter, with performance remaining consistently high across the wide tested range.

E.2 Compressed rank r

We apply the JANUS rectification on a LLaMA-2-7b model fine-tuned on the Math task via FF with various compressed ranks r. The results in Tab. 12 show a noticeable stability effect only at $r = 4$. During compression, we need to perform SVD and approximation on the matrix $\boldsymbol{M} = \boldsymbol{R}_A \boldsymbol{R}_X^\top$; we thus analyze its singular value spectrum. We save the complete layer inputs $\boldsymbol{X}$ and pre-activation gradients $\boldsymbol{A}$ for `q_proj` and `up_proj` in layers 0, 15, and 31. The energy ratios captured by the first r singular values, $\sum_{i=1}^{r} \sigma_i^2 / \sum_i \sigma_i^2$, are summarized in Tab. 13. The results show that the singular values decay rapidly and that low ranks preserve most of the matrix energy.

E.3 Replay size

We apply the JANUS rectification on a LLaMA-2-7b model fine-tuned on the Math task via FF with various replay sizes m, as summarized in Tab. 14. As the replay-buffer size decreases, recovery of old-task performance also decreases, as expected: 256 samples are already only a small batch, and further reduction weakens the buffer's representativeness of the full task distribution. We also inspect the singular value spectrum of the full Jacobians constructed by 256 samples from layers 0,

Table 10: Performance comparison of different rectified models under varying acceptance threshold.

(a) LLaMA-2-7b + Math + FF

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	GSM8k	Math	$AP_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
JANUS-F ($\tau = 0.7$)	28.00	4.57	5.56	57.71	58.45	12.32	97.59	77.65
JANUS-F ($\tau = 0.8$)	32.91	7.48	7.04	74.43	58.61	12.02	96.53	85.48
JANUS-F ($\tau = 0.9$)	34.95	9.64	7.63	82.91	58.83	12.24	97.59	90.25
JANUS-F ($\tau = 0.925$)	35.58	9.78	7.87	84.94	58.83	12.26	97.67	91.30
JANUS-F ($\tau = 0.95$)	35.57	9.97	7.92	85.55	58.76	12.28	97.69	91.62

(b) LLaMA-2-7b + Code + PiSSA

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	HumanEval	MBPP	$AP_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	33.88	28.41	100.00	86.78
JANUS-P ($\tau = 0.7$)	49.47	13.07	7.73	98.74	22.05	25.55	77.51	88.12
JANUS-P ($\tau = 0.8$)	49.47	13.38	7.73	99.28	21.86	24.73	75.78	87.53
JANUS-P ($\tau = 0.9$)	49.47	13.38	7.73	99.28	21.86	24.73	75.78	87.53
JANUS-P ($\tau = 0.925$)	50.65	16.34	8.32	108.62	21.67	25.13	76.21	92.42
JANUS-P ($\tau = 0.95$)	50.60	16.26	8.32	108.45	21.99	24.77	76.05	92.25

(c) LLaMA-3-8b + Math + FF

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	GSM8k	Math	$AP_2(\%)$	AP(%)
LLaMA-3-8b	61.66	21.86	9.94	100.00	–	–	–	–
FF	41.43	5.21	4.13	44.19	75.36	24.84	100.00	72.10
JANUS-F ($\tau = 0.7$)	50.62	8.31	4.38	54.72	76.35	24.84	100.66	77.69
JANUS-F ($\tau = 0.8$)	57.74	16.84	6.00	77.01	76.35	24.38	99.73	88.37
JANUS-F ($\tau = 0.9$)	59.42	19.09	6.99	84.67	76.65	24.98	101.14	92.91
JANUS-F ($\tau = 0.925$)	59.42	19.09	6.99	84.67	76.65	24.98	101.14	92.91
JANUS-F ($\tau = 0.95$)	59.21	19.00	6.94	84.25	76.57	25.02	101.17	92.71

(d) LLaMA-3-8b + IF + LoRA

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	MTBench	$AP_2(\%)$	AP(%)
LLaMA-3-8b	61.66	21.86	9.94	100.00	–	–	–
FF	42.69	10.08	3.94	51.66	5.94	100.00	75.83
JANUS-L ($\tau = 0.7$)	60.95	17.12	6.74	81.66	5.6	94.28	87.97
JANUS-L ($\tau = 0.8$)	60.95	17.12	6.74	81.66	5.61	94.44	88.05
JANUS-L ($\tau = 0.9$)	61.71	20.78	8.71	94.26	5.59	94.11	94.18
JANUS-L ($\tau = 0.925$)	61.71	20.78	8.71	94.26	5.68	95.62	94.94
JANUS-L ($\tau = 0.95$)	61.91	20.66	8.66	94.01	5.87	98.82	96.42

Table 11: Performance comparison of different rectified models under varying decay factor β.

(a) LLaMA-2-7b + Math + FF

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	GSM8k	Math	$AP_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
JANUS-F ($\beta = 0.6$)	35.62	9.81	7.78	84.50	58.61	12.20	97.25	90.87
JANUS-F ($\beta = 0.7$)	35.57	9.97	7.92	85.55	58.76	12.28	97.69	91.62
JANUS-F ($\beta = 0.8$)	34.64	8.95	7.63	81.50	58.98	12.10	97.16	89.33
JANUS-F ($\beta = 0.9$)	35.46	9.58	7.78	83.99	58.76	12.06	96.81	90.40

(b) LLaMA-2-7b + Code + PiSSA

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	HumanEval	MBPP	$AP_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	33.88	28.41	100.00	86.78
JANUS-P ($\beta = 0.6$)	50.50	15.62	8.22	106.69	21.88	25.28	76.78	91.73
JANUS-P ($\beta = 0.7$)	50.60	16.26	8.32	108.45	21.99	24.77	76.05	92.25
JANUS-P ($\beta = 0.8$)	50.53	16.15	8.42	108.79	21.61	24.27	74.61	91.70
JANUS-P ($\beta = 0.9$)	50.39	15.10	8.07	104.84	21.86	24.63	75.61	90.23

(c) LLaMA-3-8b + Math + FF

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	GSM8k	Math	$AP_2(\%)$	AP(%)
LLaMA-3-8b	61.66	21.86	9.94	100.00	–	–	–	–
FF	41.43	5.21	4.13	44.19	75.36	24.84	100.00	72.10
JANUS-F ($\beta = 0.6$)	59.09	18.06	6.89	82.59	76.35	24.72	100.42	91.50
JANUS-F ($\beta = 0.7$)	59.21	19.00	6.94	84.25	76.57	25.02	101.17	92.71
JANUS-F ($\beta = 0.8$)	59.43	18.17	7.53	85.09	75.74	24.76	100.09	92.59
JANUS-F ($\beta = 0.9$)	59.06	17.78	6.94	82.31	76.19	24.80	100.47	91.39

(d) LLaMA-3-8b + IF + LoRA

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	MTBench	$AP_2(\%)$	AP(%)
LLaMA-3-8b	61.66	21.86	9.94	100.00	–	–	–
FF	42.69	10.08	3.94	51.66	5.94	100.00	75.83
JANUS-L ($\beta = 0.6$)	61.93	20.06	8.66	93.11	5.69	95.79	94.45
JANUS-L ($\beta = 0.7$)	61.91	20.66	8.66	94.01	5.87	98.82	96.42
JANUS-L ($\beta = 0.8$)	61.79	20.33	8.81	93.95	5.76	96.97	95.46
JANUS-L ($\beta = 0.9$)	61.66	19.78	8.46	91.87	5.73	96.46	94.16

Table 12: Performance comparison of different rectified models under varying compressed rank r.

Method	TriviaQA	NQ open	WebQS	$AP_1(\%)$	GSM8k	Math	$AP_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
JANUS-F ($r = 32$)	35.57	9.97	7.92	85.55	58.76	12.28	97.69	91.62
JANUS-F ($r = 16$)	34.75	9.14	7.73	82.48	58.91	12.06	96.94	89.71
JANUS-F ($r = 8$)	34.77	9.17	7.78	82.83	58.98	12.26	97.79	90.31
JANUS-F ($r = 4$)	33.04	7.59	7.19	75.57	58.45	12.24	97.27	86.42

Table 13: Energy ratios of $\boldsymbol{M} = \boldsymbol{R}_A \boldsymbol{R}_X^\top$ captured by the first r singular values.

Layer	$r = 4$ (%)	$r = 8$ (%)	$r = 16$ (%)	$r = 32$ (%)
`0-q_proj`	98.916	99.855	99.995	99.999
`0-up_proj`	88.351	95.574	99.412	99.998
`15-q_proj`	80.505	93.770	97.909	99.999
`15-up_proj`	68.238	86.126	97.909	99.997
`31-q_proj`	83.841	93.876	99.428	99.998
`31-up_proj`	60.587	81.848	97.056	99.999

15, and 31 for `q_proj` and `up_proj`. The energy ratios captured by the first r singular values are listed in Tab. 15. The results empirically show that, gradients across samples are highly correlated, so a small-sample Jacobian can capture dominant directions of the full Jacobian.

Table 14: Performance comparison of different rectified models under varying replay size m.

Method	TriviaQA	NQ open	WebQS	$\text{AP}_1(\%)$	GSM8k	Math	$\text{AP}_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
JANUS-F ($m = 256$)	35.57	9.97	7.92	85.55	58.76	12.28	97.69	91.62
JANUS-F ($m = 128$)	32.91	7.59	7.53	77.44	58.68	12.26	97.54	87.49
JANUS-F ($m = 64$)	30.00	4.35	6.40	63.41	58.83	12.14	97.19	80.30

Table 15: Energy ratios of $\boldsymbol{J}$ captured by the first r singular values.

Layer	$r = 16$ (%)	$r = 32$ (%)	$r = 64$ (%)	$r = 128$ (%)
`0-q_proj`	97.10	98.46	99.32	99.81
`0-up_proj`	79.77	86.27	92.17	97.08
`15-q_proj`	57.21	66.89	77.67	89.80
`15-up_proj`	57.51	64.12	73.50	86.32
`31-q_proj`	72.81	77.87	84.49	92.43
`31-up_proj`	58.08	63.86	72.75	85.60

E.4 Replay distribution

We apply the JANUS rectification on a LLaMA-2-7b model fine-tuned on the Math task via FF with varying replay buffers sampled from TriviaQA and WebQS, as listed in Tab. 16. The results show that sampling replay data from any one of the three tasks protects that task reasonably well, while cross-task generalization varies with the source distribution.

Table 16: Performance comparison of different rectified models under varying replay distribution.

Method	TriviaQA	NQ open	WebQS	$\text{AP}_1(\%)$	GSM8k	Math	$\text{AP}_2(\%)$	AP(%)
LLaMA-2-7b	52.52	18.95	5.81	100.00	–	–	–	–
FF	19.19	0.86	4.58	39.97	60.20	12.56	100.00	69.98
JANUS-F (NQ open)	35.57	9.97	7.92	85.55	58.76	12.28	97.69	91.62
JANUS-F (TriviaQA)	35.66	5.21	6.30	67.94	59.29	12.12	97.49	82.72
JANUS-F (WebQS)	28.43	2.33	6.00	56.57	59.14	12.22	97.77	77.17

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: 1

   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: 6

   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: [Yes]

Justification: 3, A.1

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: 5.1, C

   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: 5.1, C

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: 5.1, C

   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: [Yes]

   Justification: 5.2, D

   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: C

   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: This paper conforms with the NeurIPS Code of Ethics in every respect.

   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: [N/A]

    Justification: Our study addresses the catastrophic forgetting problem in fine-tuning large models. We only use public available datasets. There is no societal impact of the work performed.

    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: This paper poses no such risks.

    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: 5.1, C

    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: This paper does not release new assets.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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 LLM is used only for writing. This paper does not involve LLMs as any important, original, or non-standard components.

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.