PRELIMINARY PREPRINT VERSION – DO NOT CITE
This is not the version of record. The final version will appear in the Proceedings of IJCAI-ECAI 2026.

TS-PEFT: Unveiling Token-Level Redundancy in Parameter-Efficient Fine-Tuning*

Dabiao Ma[1] , **Ziming Dai**[2] , **Zhimin Xin**[1] , **Shu Wang**[1] , **Jian Yang**[1] and **Haojun Fei**[1]
[1]Qfin Holdings, Inc.
[2]Tianjin University
madabiao-jk@qifu.com, phoenixdai@tju.edu.cn,
{xinzhimin-jk,wangshu1-jk,yangjian1-jk,zhangchulan-jk}@qifu.com

Abstract

Current Parameter-Efficient Fine-Tuning (PEFT) methods typically operate under an implicit assumption: Once a target module is selected, every token passing through it contributes equally to the downstream task and requires a parameter update. In this paper, we challenge this convention by revealing a pervasive token-level redundancy in the fine-tuning of large models (LMs). We propose TS-PEFT, a theoretical framework utilizing proximal optimization that acts as a dynamic probe to identify token-level redundancy during the fine-tuning process. Extensive experiments demonstrate that indiscriminately updating all tokens is not only computationally superfluous but often introduces optimization noise. Surprisingly, by discarding 30%-70% of token updates, TS-PEFT consistently matches or exceeds the performance of dense baselines such as LoRA, DoRA. Our in-depth analysis shows that the learned token-level sparsity is a superior indicator of module importance compared to traditional weight criteria, providing a novel data-driven perspective on the intrinsic adaptation mechanism of LMs.

1 Introduction

Large models (LMs) are demonstrating astonishing capabilities in Natural Language Processing (NLP) [Yao *et al.*, 2024; Guo *et al.*, 2025] and Computer Vision (CV) [Croitoru *et al.*, 2023; Liu *et al.*, 2024d]. To effectively use these large-scale models in downstream tasks, the dominant paradigm is Parameter-Efficient Fine-Tuning (PEFT)[Hu *et al.*, 2023; Han *et al.*, 2024]. In these approaches, prompt-based methods such as Prefix-Tuning [Li and Liang, 2021] and Prompt-Tuning [Lester *et al.*, 2021] train additional input tokens to change model behavior and low-rank adaptation (LoRA) [Hu *et al.*, 2022] and its variants [Zhang *et al.*, 2023; Liu *et al.*, 2024c; Liu *et al.*, 2024c] change internal representations by injecting a small number of trainable parameters or corridors, i.e. low-rank components, into the base model. These methods effectively decrease trainable parameters while maintaining similar, or comparable, performance to full parameter fine-tuning [Dai *et al.*, 2026].

However, despite their distinct architectural designs, these methods share a common and implicit assumption that they all employ dense updates. That is, once a module like a self-attention layer is used for fine-tuning, the learnable modification is applied indiscriminately to every token position in the input sequence. Formally, for an input sequence $X = \{x_i\}_{i=1}^{T}$ and a frozen weight matrix W_0, the hidden state at position i is usually written as:

$$h_i = W_0 x_i + \mathrm{M}(x_i), \tag{1}$$

where $\mathrm{M}(\cdot)$ is a PEFT module such as LoRA.

In this paper, we challenge this convention and investigate a fundamental scientific question: ***Is it necessary to apply fine-tuning updates to all token positions?*** Given the strong representational power of pretrained base models, we assume that for a specific downstream sample, only a subset of tokens requires task-specific adaptation, while others retain sufficient information from the pretrained weights. Under this assumption, the standard dense update strategy is suboptimal, primarily because it introduces optimization noise that forces the model to fit gradients on irrelevant tokens, distorting the well-learned features.

Recent works like CFT [Ruan *et al.*, 2025] have demonstrated that masking non-critical tokens during training can improve reasoning. This finding corroborates our intuition that, just as imposing a uniform penalty on all tokens in pretraining may hurt generalization, updating all tokens indiscriminately in PEFT similarly introduces redundancy and noise. To systematically verify this assumption and quantify the potential redundancy, we propose **T**oken-**S**elective **PEFT** (TS-PEFT). Unlike prior approaches, TS-PEFT serves as a learnable probe that dynamically decides whether to update a token or keep its representation frozen. We formulate this selection process as a sparsity-regularized optimization problem, employing proximal optimization and an Adam-style adaptive thresholding mechanism to ensure stable, end-to-end training of the gating decisions.

Our empirical findings unveil a pervasive token-level redundancy. Across many benchmarks, we observe that nearly 30%-70% of token updates are actually redundant. Crucially, we prove that indiscriminately updating these tokens

*The extended version of this paper with appendices is available at https://arxiv.org/abs/2511.16147.

IJCAI-ECAI 2026 PREPRINT

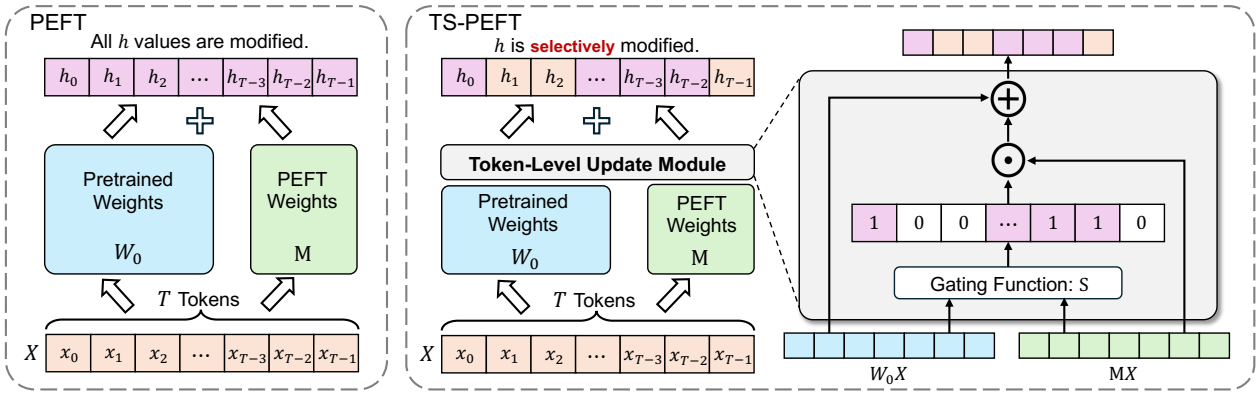

Figure 1: Comparison between standard PEFT and our TS-PEFT framework.

is detrimental. As evidenced by our analysis, forcing updates on these redundant positions leads to a "performance cliff", confirming that redundancy in PEFT acts as harmful noise. By filtering out this noise, TS-PEFT consistently matches or exceeds the performance of dense baselines (e.g., LoRA, DoRA) while utilizing significantly fewer active parameters.

Furthermore, our analysis exposes a profound connection between token sparsity and module importance. We discover that the learned sparsity pattern is not random. The modules that exhibit low sparsity, i.e., requiring frequent updates, are significantly more important to the downstream task than those with high sparsity. This insight allows token-level sparsity to serve as a superior indicator for module selection compared to traditional weight-norm-based metrics. In summary, our contributions are as follows:

- **Unveiling Redundancy:** We identify and quantify that in standard PEFT, nearly 30%-70% of token updates are redundant. We empirically demonstrate that indiscriminately updating these tokens is suboptimal and introduces optimization noise.
- **Methodological Framework:** We propose TS-PEFT, a theoretical framework that employs proximal optimization to dynamically learn token-selective strategies in an end-to-end manner.
- **Mechanism Insight:** We reveal that token-level sparsity provides a strong signal for module importance, offering a novel data-driven perspective on the intrinsic adaptation mechanism of LMs.

2 Related Work

Low-Rank Adaptation. Low-rank adaptation and its derivatives have demonstrated strong potential in reducing the number of trainable parameters while maintaining downstream performance. LoRA [Hu *et al.*, 2022] introduces two low-rank trainable matrices A and B to approximate weight updates, thereby avoiding direct modification of full weight matrices in large pretrained models. After training, these matrices can be merged back into the base model without incurring additional inference cost. Based on this concept, AdaLoRA [Zhang *et al.*, 2023] adaptively redistributes the effective rank across different layers to better meet task-specific requirements. DoRA [Liu *et al.*, 2024c] refines the decomposition of weight updates to improve the expressiveness and stability of low-rank adaptation, and VeRA [Kopiczko *et al.*, 2024] utilizes shared low-rank bases to further reduce the parameter overhead for each task. However, all these methods implicitly assume that once a layer is selected as the PEFT target, every token position passing through that layer should receive an update. That is to say, they perform parameter compression at the module level while still maintaining dense updates along the positional dimension. This may introduce unnecessary redundant modifications and fails to explicitly characterize or exploit potential token-level redundancy.

Sparsity. The methods of sparsity guidance have been widely used to improve the efficiency and interpretability of large language models (LLMs). By reducing the number of active parameters and concentrating computation on core components, these approaches not only make models more efficient but also offer clearer insights into their decision-making processes. Mixture-of-experts (MoE) models, such as Switch Transformers [Fedus *et al.*, 2022] and Mixtral 8x7B [Jiang *et al.*, 2024], employ sparse activation to keep the computational cost roughly constant even with a very large number of parameters. Recently, several works have combined MoE architectures with LoRA-style adapters to further enhance performance [Chen *et al.*, 2024; Liu *et al.*, 2024b; Zadouri *et al.*, 2024]. Another typical application of sparsity is model pruning, where less influential parameters are removed while preserving important ones to maintain accuracy under lower computational budgets [Ma *et al.*, 2023; Xu *et al.*, 2024; Xia *et al.*, 2024]. [Tan *et al.*, 2024] further introduces sparsity-guided techniques as a tool for interpreting the behavior of LLMs.

We observe that these methods all utilize redundancy at different levels. MoE-based approaches typically target modular-level sparsity, whereas pruning focuses on parameter-level sparsity. In contrast, our work explores token-level sparsity within the context of PEFT, introducing a gating mechanism to selectively skip updates for tokens requiring no task-specific adaptation. Therefore, our approach is not constrained by these architectures, and theoretically it can be directly embedded as a detachable module into all LoRA variants, including LoRA-MoE, thereby further elimi-

IJCAI-ECAI 2026 PREPRINT

nating redundant computations.

3 Method

This section introduces the overall theoretical framework and optimization process of TS-PEFT. We first formalize the LoRA-style PEFT procedure and introduce a token-level control function S to determine whether a PEFT update should be applied at each position. Then, we formulate the learning of the scalar threshold τ in S as an optimization problem with a sparsity regularization term. Next, to address the non-differentiability of S with respect to τ and the issue that gradients are nearly zero at most positions, we derive a computable approximate gradient. Finally, we combine this approximate gradient with momentum and adaptive learning rate mechanisms so that it can be seamlessly integrated into the Adam optimizer, and we provide the overall algorithmic description of the TS-PEFT training process.

3.1 Token-level Selective PEFT Formulation

As shown in Eq.(1), once a layer is selected as a PEFT target layer, all token positions passing through that layer will uniformly receive the increment introduced by $\mathrm{M}(x_i)$. To break this assumption, we introduce a token-level gating function S on top of Eq.(1) to determine whether a PEFT update is actually applied at each position. Specifically, the output h_i at position i is rewritten as:

$$h_i = \begin{cases} W_0 x_i & \text{if } \mathrm{S}(W_0 x_i, \mathrm{M}(x_i)) = 0, \\ W_0 x_i + \mathrm{M}(x_i) & \text{if } \mathrm{S}(W_0 x_i, \mathrm{M}(x_i)) = 1. \end{cases} \tag{2}$$

Here, $\mathrm{S}(\cdot) \in \{0, 1\}$ serves as a gating variable. When $\mathrm{S} = 1$, the update is enabled at that position, while when $\mathrm{S} = 0$, the position retains the base model output. To instantiate Eq.(2), we focus on the relative magnitude r_i of the PEFT update with respect to the base model output at each position i: $r_i = \|\mathrm{M}(x_i)\|_2 / \|W_0 x_i\|_2$. This is because the feature norm values at different layers vary significantly. The relative values enable each layer to adaptively adjust according to its own weight change ratio. Intuitively, a larger r_i means that the PEFT update induces a stronger perturbation relative to the base model output at that position and is more likely to be relevant to the task, whereas a small r_i indicates that the modification is minor and can be treated as negligible. Therefore, we introduce a learnable scalar threshold τ for each target layer and define the control function accordingly:

$$\mathrm{S}(W_0, \mathrm{M}(x_i)) = \begin{cases} 0 & \text{if } r_i < \tau, \\ 1 & \text{if } r_i \geq \tau. \end{cases} \tag{3}$$

In other words, if the relative magnitude at a certain position is not lower than τ, we consider the update at that position worth retaining. Otherwise, the update for that position is skipped. This will form a token-level sparse update pattern along the sequence dimension within each selected PEFT module, and the threshold τ controls the overall sparsity level.

However, this threshold-based step gating introduces two training challenges: First, S is a non-smooth step function with respect to τ, so standard backpropagation provides almost no useful gradients. Second, we want S to activate PEFT updates only on a subset of token positions while keeping the remaining ones identical to the base model. This requires explicit control of the overall sparsity level during threshold learning. The next subsections tackle these two issues via a proximal objective and a computable gradient approximation.

3.2 Proximal Optimization for the Threshold

Due to the aforementioned training difficulties, directly using first-order optimization methods cannot effectively update τ. To stably update the threshold while avoiding the train-inference gap typical of differentiable relaxations (e.g., Gumbel-Softmax), we formulate its learning as a proximal optimization problem, enabling it to track the task loss and explicitly favor sparsity. Specifically, our objective function is defined as follows:

$$\min \mathcal{L}(\tau) = \ell(\tau) + \lambda \sum_i \mathbb{1}(r_i \geq \tau), \tag{4}$$

where $\ell(\tau)$ denotes the loss of the target task, and the second term penalizes the number of excessively activated tokens to encourage a sparser update. $\mathbb{1}(\cdot)$ is an indicator function, and λ, as a hyperparameter, is used to balance the sparsity level.

We employ proximal optimization to derive the update rule for parameter τ within a stochastic gradient descent (SGD) framework [Parikh *et al.*, 2014]. Given the threshold τ^k at the k-th iteration, the update τ^{k+1} can be written as:

$$\underset{\tau}{\operatorname{argmin}} \left[\ell(\tau^k) + \frac{\partial \ell}{\partial \tau}(\tau - \tau^k) + \frac{1}{2\alpha_k} \left| \tau - \tau^k \right|^2 + \lambda P(\tau) \right], \tag{5}$$

where α_k is the step size. For convenience in the subsequent derivation, we set $\sum_i \mathbb{1}(r_i \geq \tau)$ to be $P(\tau)$. It can be seen that this subproblem consists of three components: a first-order approximation of the loss at τ^k, a proximal term that constrains the magnitude of the update, and a sparsity regularization term that favors fewer activated positions.

To solve Eq.(5), we need to compute $\partial \ell / \partial \tau$ and $\partial P / \partial \tau$. Thus, we first expand the output of Eq.(2) along the hidden dimension. Let the hidden representation of the i-th token at dimension $j \in \{0, \ldots, H-1\}$ be denoted as:

$$h_{ij} = [W_0 x_i]_j + [\mathrm{M}(x_i)]_j * \mathbb{1}(r_i \geq \tau), \tag{6}$$

where H is the hidden size, and $[\cdot]_j$ denotes the j-th component of a vector. By backpropagation, we finally get:

$$\frac{\partial \ell}{\partial \tau} = \sum_i \mu_i \frac{\partial}{\partial \tau} \mathbb{1}(r_i \geq \tau), \tag{7}$$

$$\frac{\partial P}{\partial \tau} = \sum_i \lambda \frac{\partial}{\partial \tau} \mathbb{1}(r_i \geq \tau), \tag{8}$$

where μ_i aggregates the overall influence of the PEFT update at position i on the loss:

$$\mu_i = \sum_{j=0}^{H-1} \frac{\partial \ell}{\partial h_{ij}} [M(x_i)]_j. \tag{9}$$

IJCAI-ECAI 2026 PREPRINT

Algorithm 1 Forward and Backward

Input: M, α_k, s, λ, β_1, β_2, τ^k.
Output: τ^{k+1}, updated parameters of M.
Forward Pass:
  Forward as in Eq.(2).
Backward Pass:
  Step 1: Perform the usual backward step to update parameters of M.
  Step 2:
    Compute μ_i as in Eq.(9).
    Compute g_k as in Eq.(15).
    Update τ^{k+1} as in Eq.(20).

3.3 Gradient Approximation for the Step Function

In the previous subsection, we obtain the derivative expressions of the loss function and the sparsity penalty term with respect to the threshold τ. However, all these derivatives depend on the partial derivative of the step gating $\mathbb{1}(r_i \geq \tau)$ with respect to τ, and its numerical value is almost zero everywhere, making it impossible to directly perform gradient updates. Therefore, to enable stable learning of the threshold through backpropagation, this section constructs a computable gradient approximation based on the perspective of proximal optimization to replace the non-differentiable derivative of the indicator function.

A straightforward approximation, inspired by [Bengio *et al.*, 2013], is to treat the gradient of the step function as a constant $-s, s > 0$, that is:

$$\frac{\partial}{\partial \tau}\mathbb{1}\left(r_i \geq \tau\right) \approx -s. \tag{10}$$

Substituting this approximation value into Eq.(7)-(9) yields the rough gradients of the loss term and the sparsity term with respect to the threshold:

$$\frac{\partial \ell}{\partial \tau} \approx -s \sum_i \mu_i, \tag{11}$$

$$\frac{\partial P}{\partial \tau} \approx -s \sum_i \lambda. \tag{12}$$

Eq.(10) is a rather rough approximation, which causes highly oscillatory updates of τ and results in poor model performance. We believe this is because Eq.(10) should be 0 for certain value domains, but for the update, we set it to $-s$. Therefore, to limit this effect, we add indicator conditions. We modify Eq.(11) and Eq.(12) as follows:

$$\frac{\partial \ell}{\partial \tau} = -s \sum_i \mathbb{1}\left[\mathbb{1}\left(\mu_i \geq 0\right) = \mathbb{1}\left(r_i \geq \tau\right)\right] \mu_i, \tag{13}$$

$$\frac{\partial P}{\partial \tau} = -s \sum_i \mathbb{1}\left(r_i \geq \tau\right) \lambda. \tag{14}$$

The constraints can be understood as follows: Whenever τ is either large enough ($\mu_i \geq 0$, $r_i < \tau$) or small enough ($\mu_i \leq 0$, $r_i \geq \tau$) to render the update of τ unnecessary, Eq.(13) zeros $\partial \ell / \partial \tau$. Similarly, when τ is large enough ($r_i < \tau$, λ is a positive constant), Eq.(14) nullifies $\partial P / \partial \tau$.

Combining both parts, the approximate gradient for updating the threshold can be unified as:

$$g_k = \sum_i \left\{\mathbb{1}\left[\mathbb{1}\left(\mu_i \geq 0\right) = \mathbb{1}\left(r_i \geq \tau\right)\right] \mu_i + \mathbb{1}\left(r_i \geq \tau\right) \lambda\right\}. \tag{15}$$

Finally, given the learning rate α_k, the threshold is updated as $\tau^{k+1} = \tau^k + \alpha_k s \cdot g_k$, where $-\alpha_k s$ can be regarded as the learning rate.

3.4 Adam-style Threshold Update and Overall Training Procedure

In fact, for the threshold τ, its approximate gradient g_k is highly noisy and varies in scale across training steps. The contributions of tokens change significantly between batches, and the magnitude of g_k can fluctuate during optimization. Directly applying plain gradient descent to τ often leads to oscillations and can slow down or make the convergence unstable. To mitigate this, we adopt a rule similar to Adam to update, which combines momentum and adaptive learning rate [Kingma *et al.*, 2015].

At the k-th iteration, we use the approximate gradient g_k obtained in the previous subsection to update the first and second moments:

$$m_k \leftarrow \beta_1 m_{k-1} + (1-\beta_1) g_k, \tag{16}$$

$$v_k \leftarrow \beta_2 v_{k-1} + (1-\beta_2) g_k^2, \tag{17}$$

$$\hat{m}_k \leftarrow m_k / (1-\beta_1^k), \tag{18}$$

$$\hat{v}_k \leftarrow v_k / (1-\beta_2^k), \tag{19}$$

where β_1 controls the contribution of past gradients and β_2 governs the decay rate of squared gradients. $\hat{m}_k$ and $\hat{v}_k$ denote the bias-corrected first- and second-moment estimates used in the Adam update. Finally, the threshold is updated as:

$$\tau^{k+1} \leftarrow \tau^k + \alpha_k s \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}, \text{ with } \tau^0 = 0, \tag{20}$$

where s is the gradient scaling factor, and ϵ is a minuscule constant to prevent division by zero. We empirically observe that the adaptive moment estimation (Eq.(16)-Eq.(20)) is critical for convergence. Ablation studies in Appendix A.4 demonstrate that removing this component leads to severe threshold oscillation and training divergence, resulting in significant performance degradation on sensitive tasks.

Combining the above components, the training process of TS-PEFT can be summarized as follows: In each training step, during the forward propagation phase, we perform gating based on each token's relative update magnitude r_i and the threshold τ of the corresponding layer, applying updates only at positions that satisfy the condition while keeping the base model output unchanged at all other positions. Then, standard backpropagation is used to update all PEFT parameters, while the base model parameters remain frozen throughout. On this basis, we compute μ_i to construct the approximate gradient g_k for the threshold, and independently update the threshold τ of each layer using Adam-style first- and second-moment estimates. The full training procedure is presented in Algorithm 1.

IJCAI-ECAI 2026 PREPRINT

	Hyperparams	PIQA	BoolQ	HellaSwag	WinoGrande	SIQA	OBQA	ARC-e	ARC-c	Avg
LoRA	$s = 4\text{e}{-5}$	88.5	64.2	95.3	**85.5**	80.8	85.4	**90.3**	**79.9**	83.7
TS-LoRA	$\lambda = 4.5\text{e}{-5}$	**88.6**	**70.1**	**95.5**	84.4	**82.3**	**85.8**	90.1	79.4	**84.5**
Sparsity(%)		58.8	55.6	57.1	54.0	54.9	56.8	59.2	60.0	57.1
DoRA	$s = 4\text{e}{-5}$	87.5	75.0	95.5	85.5	80.1	85.8	90.8	79.5	85.0
TS-DoRA	$\lambda = 1\text{e}{-5}$	**88.8**	**75.2**	95.5	**87.1**	**80.2**	**86.6**	**91.2**	**80.1**	**85.6**
Sparsity(%)		50.0	47.9	50.2	47.1	47.6	48.3	49.6	50.0	48.8
AdaLoRA	$s = 4\text{e}{-5}$	**88.8**	74.2	95.5	85.2	79.6	85.4	**91.2**	79.5	84.9
TS-AdaLoRA	$\lambda = 1\text{e}{-4}$	88.4	**75.5**	**95.8**	**85.6**	**80.9**	**86.2**	90.1	**79.9**	**85.3**
Sparsity(%)		74.3	76.0	65.0	70.8	68.9	72.8	75.5	75.9	67.7

Table 1: Evaluation scores of LLaMA3.1-8B on CSR benchmarks; the last row in each block shows the sparsity of TS-PEFT.

4 Experiments

4.1 Experimental Setup

We evaluate TS-PEFT in multiple representative scenarios: Natural Language Understanding (NLU), Commonsense Reasoning (CSR), Visual Instruction Tuning (VIT), and Natural Language Generation (NLG). Due to space limitations, we present the detailed results and analysis for NLG in Appendix A.3. The experimental setup covers the datasets, baseline models, and hyperparameter configurations to ensure reproducibility and comparability of the results.

Datasets and Models. We select several representative benchmarks in the current PEFT and small-model fine-tuning literature. For CSR, we evaluate on LLaMA-3.1-8B [Dubey *et al.*, 2024] using PIQA, BoolQ, HellaSwag, WinoGrande, SIQA, OBQA, and ARC-e/ARC-c, covering typical language reasoning scenarios such as physical commonsense, boolean question answering, narrative completion, and scientific QA [Liu *et al.*, 2024c]. For VIT, we evaluate LLaVA-1.5-7B [Liu *et al.*, 2023; Liu *et al.*, 2024a] on ScienceQA, POPE, MMBench, GQA, VisWiz, VQAv2, and VQAT, which are generative language tasks with open-ended answers conditioned on images and textual instructions. For NLU, we build on DeBERTaV3-base [He *et al.*, 2021] and use the GLUE benchmark datasets [Wang *et al.*, 2018]. The GLUE tasks follow the official evaluation protocol: CoLA uses the Matthews correlation coefficient, STS-B uses correlation coefficients, and all other tasks use accuracy. Additionally, for the NLG tasks (detailed in Appendix), we evaluate Mistral-7B-v0.1 [Jiang *et al.*, 2023] on the Vicuna [Chiang *et al.*, 2023], LIMA [Zhou *et al.*, 2023] and Dolly [Conover *et al.*, 2023] datasets.

Baselines. In our experiments, we select several mainstream PEFT methods as baselines, covering two major categories: low-rank update methods and adapter-based methods. Specifically, LoRA, DoRA, and AdaLoRA are used to examine whether TS-PEFT can consistently provide additional gains under different low-rank update strategies, while Parallel Adapter [Hu *et al.*, 2023] is included to evaluate the method's scalability on adapter-style architectures. For each baseline, we keep its original implementation, hyperparameters, and rank configuration unchanged, and apply TS-PEFT on top of it. This ensures that the comparison is not affected by factors unrelated to the baseline itself and that the results accurately reflect the impact of token-level selective updates.

Training details. In all experiments, we freeze the base model and update only the PEFT parameters and the threshold τ. We use the AdamW optimizer, where PEFT is the default learning rate and the effective learning rate of τ is further modulated by the scaling factor s. For text tasks, we follow standard instruction-tuning configurations. For vision–language tasks, we adopt the original training pipeline of the corresponding models. Unless otherwise specified, we set $\alpha_k = 1$, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. Each main experiment is repeated at least three times with different random seeds, and we report the mean scores across runs. Additional hyperparameter settings are summarized in Appendix A.6.

4.2 Experimental Results

Commonsense Reasoning. Table 1 reports the results of various PEFT methods and their TS-PEFT variants on the CSR tasks. For a fair comparison, we set the rank of both LoRA and DoRA to 32, and configure AdaLoRA with an initial rank of 64 and a target rank of 32. Under the same rank settings, we observe that TS-LoRA, TS-DoRA, and TS-AdaLoRA achieve performance comparable to or better than their respective baselines on most datasets. This indicates that selectively skipping updates on a subset of tokens does not weaken downstream performance. Instead, it can help alleviate the noise introduced by redundant updates.

The table also reports the sparsity rate of TS-PEFT, which is the proportion of positions in the input sequence that are **not** modified by the PEFT output, as defined in Eq.(21). Across different baselines and tasks, this proportion remains roughly within the 50%–70% range, meaning that only 30% to 50% of the tokens actually receive PEFT updates. Even under this degree of sparsity, TS-PEFT is able to match or outperform the original baselines in commonsense reasoning, suggesting that standard PEFT exhibits substantial redundancy along the token dimension, and that threshold-based token-level selective updating provides a more efficient use of PEFT capacity.

$$\text{Sparsity}(W_0, \text{M}, X) = 1 - \frac{1}{T} \sum_i \text{S}(W_0 x_i, \text{M}(x_i)). \quad (21)$$

IJCAI-ECAI 2026 PREPRINT

	Hyperparams	SQA	POPE	MMBench	GQA	VisWiz	VQAv2	VQAT	Avg
LoRA	$s = 4e{-}5$	68.1	87.3	**64.8**	63.1	46.8	**79.1**	57.3	66.6
TS-LoRA	$\lambda = 2e{-}6$	**68.3**	**87.5**	63.6	**63.3**	**50.4**	78.3	**57.5**	**67.0**
Sparsity(%)		67.6	59.3	62.0	59.3	59.5	59.2	62.2	61.3
DoRA	$s = 4e{-}5$	67.5	87.4	**65.5**	**63.0**	50.5	**78.6**	**57.0**	67.1
TS-DoRA	$\lambda = 1e{-}6$	**69.4**	**88.0**	64.7	62.4	**54.2**	77.8	55.9	**67.4**
Sparsity(%)		70.6	62.6	65.2	62.4	62.9	63.8	65.2	64.7
AdaLoRA	$s = 4e{-}5$	**68.0**	87.3	**64.3**	61.7	48.9	78.3	**56.9**	66.5
TS-AdaLoRA	$\lambda = 5e{-}7$	67.8	**87.5**	63.4	**62.3**	**52.6**	**78.5**	55.2	**66.8**
Sparsity(%)		60.0	50.9	54.1	50.9	51.4	50.9	53.9	53.2

Table 2: Evaluation scores of LLaVA-1.5-7B on VIT benchmarks; the last row in each block shows the sparsity of TS-PEFT.

Method	MNLI	CoLA	QNLI	RTE	MRPC	QQP	SST-2	STS-B	Avg
LoRA	89.9	69.2	94.1	87.1	90.4	**92.2**	95.3	**91.8**	88.8
TS-LoRA	**89.9**	**70.3**	**94.2**	**88.1**	**91.2**	92.0	**95.8**	91.5	**89.1**
Sparsity(%)	71.6	64.1	70.6	42.8	55.4	57.1	67.6	33.9	57.9
λ	1e-5	6e-6	2e-5	1e-5	1e-5	2e-6	2e-5	8e-5	-
AdaLoRA	90.4	71.1	**94.6**	88.6	91.3	91.9	95.9	**91.9**	89.5
TS-AdaLoRA	**90.5**	**71.3**	94.5	**89.0**	**91.6**	91.9	**96.1**	91.6	**89.6**
Sparsity(%)	55.2	53.1	68.1	62.7	71.7	54.7	56.9	68.3	61.3
λ	9e-6	5e-6	4.5e-7	2e-6	2e-8	5e-7	2e-6	5e-6	-
Adapter	**90.4**	**72.1**	**94.2**	86.6	90.0	**92.1**	95.6	**91.4**	89.1
TS-Adapter	90.1	72.0	94.1	**86.8**	**91.4**	91.8	**96.6**	91.3	**89.3**
Sparsity(%)	56.4	59.4	65.5	62.7	55.1	58.0	56.2	67.7	60.1
λ	5e-7	4.5e-7	4.5e-7	2e-6	2e-8	5e-7	2e-6	5e-6	-

Table 3: Evaluation scores of DeBERTaV3-base on GLUE benchmarks; the "Sparsity(%)" rows show the sparsity of TS-PEFT, and the "λ" rows list task-wise thresholds (s = 1e-4).

IJCAI-ECAI 2026 PREPRINT

Visual Instruction Tuning. For visual instruction tuning, we follow the training setup of [Liu *et al.*, 2024c], configuring LoRA and DoRA with rank 128 and setting the initial and target ranks of AdaLoRA to 256 and 128. Table 2 shows the results. In all benchmark tests, TS-LoRA, TS-DoRA, and TS-AdaLoRA achieve consistent improvements of about 0.3–0.4 points, while operating under 50%–70% token sparsity. This demonstrates that token-level redundancy is also present in multimodal generative tasks, and selective token updates remain effective, consistent with our observations in CSR.

Natural Language Understanding. The results are shown in Table 3, where the target ranks of LoRA and AdaLoRA are both set to 4. Overall, TS-LoRA and TS-AdaLoRA achieve average performance that is comparable to or slightly better than their respective baselines, consistent with the trends observed in the previous two task categories. To test whether our approach is tied to a specific low-rank parameterization, we further include Parallel Adapter as a non–LoRA-style PEFT baseline and apply TS-PEFT on top of it to obtain TS-Adapter, using a hidden size of 32. The results show that TS-Adapter performs no worse than the original Adapter, suggesting that TS-PEFT is not bound to LoRA-style weight structures, but can function as a general token-level gating layer that can be stacked on diverse PEFT modules.

TS-PEFT has achieved PEFT updates for only 30% - 70% of the tokens in multiple tasks, yet still maintain or even surpass the baseline performance. This clearly reveals the widespread token-level redundant updates in the current PEFT mode, and these updates also introduce noise that affects model performance. By quantifying this intrinsic redundancy, TS-PEFT establishes a clear blueprint for translating theoretical sparsity into substantial physical efficiency gains in the future.

4.3 Analysis: Token Sparsity Reveals Model Adaptation Mechanism

Sparsity Patterns Reflect Architecture Bias. There are 32 layers in LLaMA3.1-8B. We compute the sparsity of q_proj, k_proj, and v_proj in each layer and take their average to obtain the sparsity distribution of TS-PEFT across modules (detailed results are provided in Appendix A.5). Overall, the sparsity of TS-LoRA and TS-AdaLoRA is relatively evenly distributed across layers, with standard deviations of 9.8 and 11.4, respectively. In contrast, TS-DoRA exhibits a much higher standard deviation of 41.3, which shows a more polarized pattern. It should be noted that LoRA and AdaLoRA share the same weight structure despite having different rank allocation strategies, whereas DoRA adopts a distinct param-

	PIQA	BoolQ	HellaSwag	WinoGrande	SIQA	OBQA	ARC-e	ARC-c	Avg
S_low_50%	**88.6**	**75.7**	95.4	**86.9**	80.6	**87.6**	**90.8**	79.7	**85.7**
S_high_50%	88.2	62.2	92.2	85.2	80.2	86.2	90.4	79.9	83.0
norm_relative_50%	88.4	62.2	95.4	85.0	80.5	83.6	90.1	**80.2**	83.2
norm_abs_50%	88.4	52.7	85.8	83.9	78.7	86.2	89.6	79.0	80.5
random_50%	88.3	73.4	91.2	85.6	80.6	85.7	89.6	78.9	84.2
half_rank_50%	88.1	75.4	95.2	86.3	**81.2**	85.2	89.6	78.7	84.9

Table 4: Performance of LLaMA3.1-8B on the commonsense reasoning task using various selection methods.

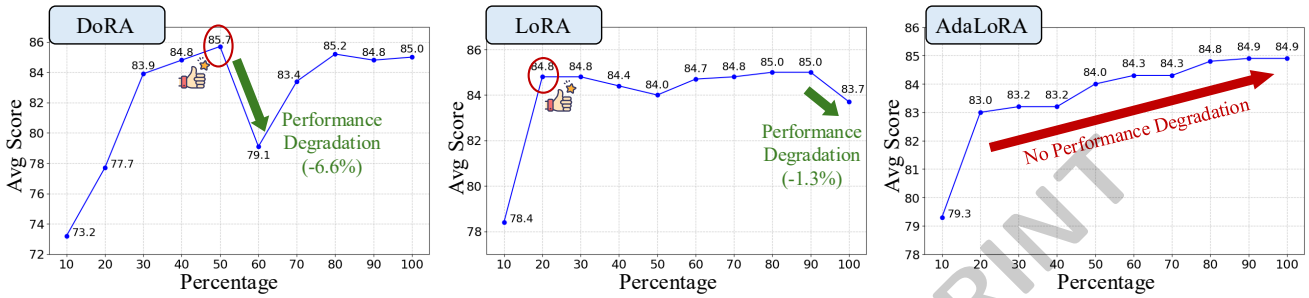

Figure 2: Performance of LLaMA3.1-8B on CSR benchmarks as the percentage of selected modules varies for each PEFT method. Note the performance drops (e.g., DoRA at 60%) when redundant, high-sparsity modules are forced to update, indicating noise injection.

eterization by decomposing weights into magnitude and direction. We hypothesize that this structural difference leads to the stronger selectivity observed in TS-DoRA at the module level, naturally raising the question: ***Are the modules with lower sparsity more critical for downstream tasks?***

Token-level Sparsity as a Module Importance Indicator. To examine whether token-level sparsity can be used to identify important modules, we design a set of selection experiments based on the sparsity patterns learned by TS-DoRA. After completing TS-DoRA fine-tuning, we rank all modules by their average token sparsity and construct two main settings under the same parameter budget: (1) S_low_50%, which fine-tunes only the 50% of modules with the lowest sparsity, and (2) S_high_50%, which fine-tunes only the 50% of modules with the highest sparsity.

We compare these sparsity-based selections against several alternatives: norm_relative_50% (selecting the 50% of modules with the largest average $\|\mathrm{M}(x_i)\|_2/\|W_0 x_i\|_2$), norm_abs_50% (the 50% with the largest average $\|\mathrm{M}(x_i)\|_2$), random_50% (randomly selecting 50% of modules, averaged over three runs), and half_rank_50% (using all modules but halving the rank so that the total number of trainable parameters matches the 50% settings). The results in Table 4 show that S_low_50% not only significantly outperforms S_high_50% under the same parameter budget, but even surpasses fine-tuning all DoRA modules (whose mean score is 85.0). In contrast, norm-based selections, random choices, and half-rank full-module tuning all yield clearly inferior performance. This indicates that, in our setting, the token-level sparsity patterns learned by TS-DoRA provide a more reliable signal of module importance than simple norm statistics or random selection.

Redundant Updates Introduce Optimization Noise. To further validate that the identified high-sparsity modules are indeed redundant and potentially harmful, we conduct a cumulative selection analysis. We expand the set of target modules in ascending order of sparsity. Specifically, we add the modules from the key ones to the redundant ones sequentially. As illustrated in Figure 2, the conventional wisdom that "more trainable parameters imply better performance" does not hold. For DoRA, performance peaks when only 50% of the modules are fine-tuned, while LoRA reaches its optimum at 80%. Remarkably, even utilizing just 20% of the modules yields results comparable to the optimal settings.

Crucially, we observe distinct performance cliffs when redundant modules are forced to update. DoRA suffers a sharp performance drop of 6.6% when the target proportion increases from 50% to 60%. LoRA experiences a 1.3% degradation when expanding from 90% to 100%. This phenomenon provides compelling evidence that updating modules characterized by high sparsity injects negative interference rather than meaningful adaptation. But AdaLoRA exhibits greater stability against these low-importance modules, likely due to its dynamic rank allocation mechanism.

5 Conclusion

This paper revisits PEFT from a token-level perspective and shows that standard PEFT methods exhibit substantial redundancy by uniformly updating all token positions within selected layers. We propose TS-PEFT, which introduces a learnable threshold gate to selectively apply PEFT updates to only about 30%–70% of token positions, yet consistently matches or surpasses strong baselines. Our analyses further show that the resulting token-level sparsity patterns provide a meaningful signal of module importance, enabling more effective module selection under the same parameter budget and offering a theoretical basis for future hardware-software co-design targeting efficient sparse fine-tuning and inference.

IJCAI-ECAI 2026 PREPRINT

Contribution Statement

Dabiao Ma, Ziming Dai, and Zhimin Xin contributed equally to this work. Haojun Fei is the corresponding author.

References

[Bengio *et al.*, 2013] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[Chen *et al.*, 2024] Shaoxiang Chen, Zequn Jie, and Lin Ma. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *arXiv preprint arXiv:2401.16160*, 2024.

[Chiang *et al.*, 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[Conover *et al.*, 2023] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[Croitoru *et al.*, 2023] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023.

[Dai *et al.*, 2026] Ziming Dai, Tuo Zhang, Fei Gao, Xingyi Cai, Xiaofei Wang, Cheng Zhang, Wenyu Wang, and Chengjie Zang. Stratos: An end-to-end distillation pipeline for customized llms under distributed cloud environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 40506–40512, 2026.

[Dubey *et al.*, 2024] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. 2024.

[Fedus *et al.*, 2022] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[Guo *et al.*, 2025] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[Han *et al.*, 2024] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

[He *et al.*, 2021] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2021.

[Hu *et al.*, 2022] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 1(2):3, 2022.

[Hu *et al.*, 2023] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 5254–5276, 2023.

[Jiang *et al.*, 2023] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[Jiang *et al.*, 2024] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[Kingma *et al.*, 2015] Diederik Kingma, Jimmy Ba Adam, et al. A method for stochastic optimization. In *International conference on learning representations (ICLR)*, volume 5. California;, 2015.

[Kopiczko *et al.*, 2024] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

[Lester *et al.*, 2021] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

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

[Liu *et al.*, 2023] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[Liu *et al.*, 2024a] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

IJCAI-ECAI 2026 PREPRINT

[Liu *et al.*, 2024b] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1114, 2024.

[Liu *et al.*, 2024c] 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.

[Liu *et al.*, 2024d] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[Ma *et al.*, 2023] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

[Parikh *et al.*, 2014] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[Ruan *et al.*, 2025] Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. Enhancing large language model reasoning via selective critical token fine-tuning. *arXiv preprint arXiv:2510.10974*, 2025.

[Tan *et al.*, 2024] Zhen Tan, Tianlong Chen, Zhenyu Zhang, and Huan Liu. Sparsity-guided holistic explanation for llms with interpretable inference-time intervention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 21619–21627, 2024.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018.

[Xia *et al.*, 2024] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, 2024.

[Xu *et al.*, 2024] Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *International Conference on Learning Representations*, 2024.

[Yao *et al.*, 2024] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024.

[Zadouri *et al.*, 2024] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In *International Conference on Learning Representations*, 2024.

[Zhang *et al.*, 2023] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview, 2023.

[Zhou *et al.*, 2023] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

IJCAI-ECAI 2026 PREPRINT