TARE: Lightweight Token-Aware Representation Editing for Fine-tuning Transformer-like Models

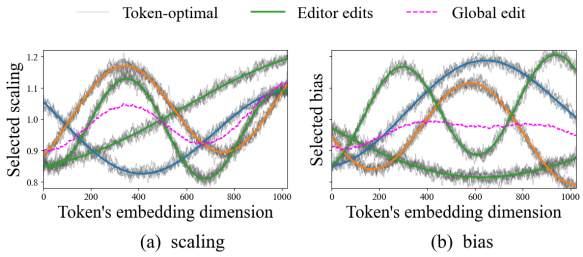
Yulong Wang and Siyu Zhao
School of Computer Science, Beijing University of Posts and Telecommunications, China
{wyl, zsy2023110857}@bupt.edu.cn

Abstract

Parameter-efficient fine-tuning (PEFT) must balance effectiveness and efficiency: low-rank methods can be costly, while global representation edits often underfit token-level contexts. We propose **Token-Aware Representation Editing (TARE)**, a PEFT method that performs fine-grained, token-specific edits with a small additional inference overhead and minimal tuning. After each FFN block in a transformer-like model, we adopt a lightweight selector that scores a small pool of hidden representation editors for each token, activates only the top-k editors, and mixes their element-wise scaling/bias updates. This design achieves superior performance while maintaining computational efficiency, yielding a more favorable Pareto frontier compared to the state-of-the-art (SOTA) methods. Across LLaMA-3-8B (eight knowledge reasoning and seven mathematical reasoning tasks) and RoBERTa-base/large (GLUE), TARE outperforms SOTAs (LoRA, DoRA, MiLoRA, LoReFT, and RED), achieving 86.7% (knowledge reasoning), 76.7% (mathematical reasoning), and 88.3% (GLUE) while tuning only 0.0392% of parameters using about 20 GiB peak GPU memory (during training). An implementation is available at: https://github.com/PatriciaPulec/tare.

1 Introduction

Parameter–efficient fine–tuning (PEFT) has become a central paradigm for adapting large Transformer-like models (e.g., LLaMA and RoBERTa) under tight compute and memory budgets: it aims to reach strong task performance by training only a tiny fraction of parameters while keeping the backbone frozen. Existing PEFT families include weight–space adapters (e.g., LoRA (Hu et al., 2021), DoRA (Liu et al., 2024), MiLoRA (Wang et al., 2024a)), representation–space editing and gating (e.g., RED (Wu et al., 2024a), LoReFT (Wu et al., 2024b), IA^3 (Liu et al., 2022), BitFit (Ben Zaken et al., 2021)). Although TARE uses token-wise Top-k routing and a load-balancing loss, it is not an MoE-based adapter or an FFN replacement. Instead, it mixes a few lightweight diagonal post-FFN hidden representation editors within a PEFT regime. Despite clear efficiency gains, a key open problem remains: how to attain fine-grained, token-aware adaptation while keeping the added inference overhead low and the hyperparameter burden modest.



(a) scaling (b) bias

Figure 1: Token-wise optimal scaling/bias (Token-optimal) forms a few modes. A single global scaling/bias (Global edit) underfits, a small set of scalings/biases (Editor edits) covers dominant modes.

Across PEFT methods, a common limitation is the tension between expressiveness and efficiency. Low-rank approaches such as LoRA (Hu et al., 2021), DoRA (Liu et al., 2024), and MiLoRA (Wang et al., 2024a) manage to fine-tune a model under a low-rank subspace but still demand a considerable number of updated learnable parameters and a proper choice of hyperparameters such as ranks and scaling factors, which can complicate tuning across layers and tasks. Representation-editing methods that are highly efficient at inference often apply a single, shared transformation to all tokens—e.g., RED (Wu et al., 2024a) learns one global per-feature scaling/bias; IA^3 (Liu et al., 2022) gates channels uniformly; BitFit (Ben Zaken et al., 2021) updates only biases—thereby limiting capacity to capture fine-grained context. LoReFT (Wu et al., 2024b) performs low-rank projections

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18452–18471

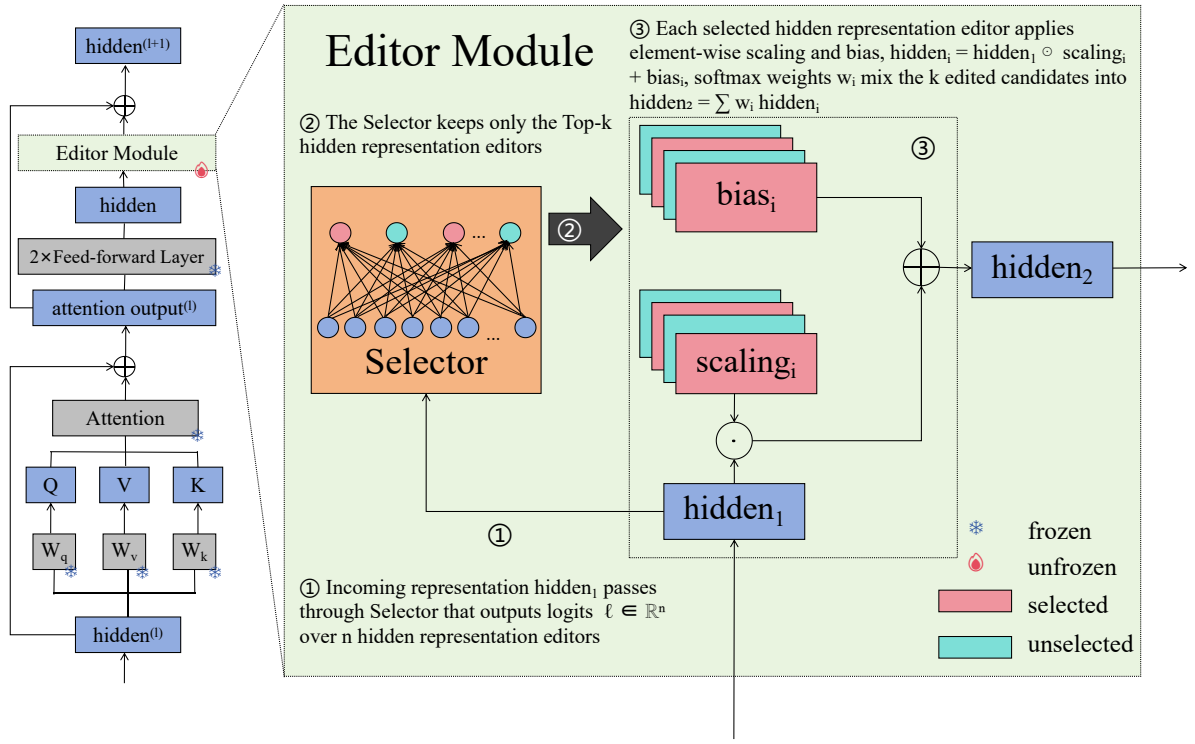


Figure 2: Schematic of the proposed TARE method. TARE inserts an Editor Module after each FFN block. Given the incoming hidden representation $\mathbf{h}$, a lightweight Selector produces logits $\boldsymbol{\ell} \in \mathbb{R}^n$ over n hidden representation editors (①), keeps only the Top-k editors (②), and each selected editor applies element-wise scaling and bias to form candidates $\mathbf{h}_i = \mathbf{h} \odot \boldsymbol{\gamma}_i + \mathbf{b}_i$. Softmax weights w_i then mix the selected candidates into the updated representation $\mathbf{h}' = \sum_i w_i \mathbf{h}_i$ (③). The backbone Transformer is frozen, while the Selector and editor parameters are trainable.

in representation space but still uses the same projection for every token and inherits rank-selection overhead. In summary, existing methods either impose a uniform editor that underfits token-level variability, or they rely on hyperparameters (e.g., ranks/scales), motivating a token-aware representation editor that preserves inference efficiency while capturing per-token context.

As shown in Figure 1, for a single layer, the per-dimension scaling (top) and bias (bottom) that would be individually optimal for different tokens (thin solid curves). Two regularities emerge. First, token requirements are highly heterogeneous across embedding dimensions: the thin curves span roughly 0.8–1.2 for both scaling and bias and exhibit clear phase shifts, indicating that different tokens prefer amplifying/suppressing different feature bands. Second, despite this heterogeneity, the thin curves concentrate around a small number of prototypical shapes (thick solid curves); most token-specific curves closely follow one of these smooth templates up to modest perturbations. In contrast, the single global edit (thin dashed) is essentially the per-dimension average; it flattens peaks and valleys and therefore underfits wherever tokens require opposite adjustments (e.g., around the mid- and high-dimensional regions where one mode rises while another falls). The same multi-modal pattern appears simultaneously in both scaling and bias, and the two often exhibit slight phase misalignment, suggesting that accurate edits must coordinate the pair rather than rely on either alone. This analysis implies that token-level edits are necessary to capture fine-grained semantics, and only a few hidden representation editors are sufficient to cover the dominant modes.

Consequently, we propose **Token-Aware Representation Editing** (*TARE*), which adopts a token-aware hidden representation editing scheme. TARE inserts a hidden representation editor module after each block's FFN: for each token, a lightweight selector produces logits over n diagonal editors and activates only the Top-k hidden representation editors; each selected hidden representation editor maintains element-wise scale and bias vectors (γ_i, b_i) to form candidate edits $h_i = h_1 \odot$

$\gamma_i + b_i$, which are then linearly mixed by softmax-normalized weights to update the representation. Because the selected units are lightweight diagonal post-FFN editors rather than expert FFNs, TARE is not an MoE-based adapter. The operations are feature-wise and sparse, so the added inference overhead remains small in practice; the backbone network of the model under fine-tuning is frozen, and only (n, k) need to be set—no rank/scale hyperparameters are introduced.

The main contributions of this work are as follows:

- We propose TARE, a new PEFT method that replaces one-size-fits-all edits with per-token, per-dimension adjustments. Conceptually, TARE uses sparse routing as a PEFT representation-editing mechanism rather than as an MoE replacement of Transformer FFN blocks. A lightweight selector scores a small pool of hidden representation editors and mixes only a few of them for each token, yielding fine-grained context adaptivity while keeping computation strictly diagonal and sparse. This directly tackles the key challenge raised above—achieving token-level expressiveness with only a small additional inference cost and without introducing rank/scale hyperparameters.

- We show that token-optimal edits cluster into a handful of smooth modes; the proposed TARE method's selector–template co-design exploits this structure by projecting each token onto a local convex combination of learned hidden representation editors. This design preserves the inference friendliness of representation editing, avoids rank/scale knobs from low-rank adapters, and provides a simple, robust training recipe with optional load-balancing regularization.

- The proposed TARE method is evaluated on a decoder model (LLaMA-3-8B) across eight knowledge reasoning tasks and seven mathematical reasoning tasks, and on encoder models (RoBERTa-base/large) on GLUE benchmark. It achieves 86.7% average over eight knowledge reasoning tasks (slightly above LoReFT and notably higher than LoRA/RED), 76.7% average over seven mathematical reasoning tasks and 88.3% on GLUE benchmark, while tuning only 0.0392% of parameters with ∼20 GiB memory. TARE consistently matches or surpasses PEFT SOTAs (LoRA, DoRA, MiLoRA, RED, and LoReFT) under tight parameter and memory budgets.

Table 1: A simplified toy running example of token-wise routing in TARE. For each token, the weights are normalized over the selected Top-3 editors and sum to 1.

Token	Illustrative token type	Top-3 editors (mixture weight)
Edgar	entity/name	E_4 (0.52), E_1 (0.28), E_7 (0.20)
eats	verb/action	E_2 (0.49), E_5 (0.33), E_1 (0.18)
18	number	E_6 (0.61), E_3 (0.25), E_8 (0.14)
day	time unit	E_3 (0.48), E_6 (0.34), E_1 (0.18)
1/2	fraction	E_6 (0.64), E_8 (0.21), E_3 (0.15)
how	question word	E_7 (0.53), E_2 (0.26), E_1 (0.21)
week	time unit	E_3 (0.52), E_6 (0.30), E_1 (0.18)
?	punctuation	E_1 (0.68), E_7 (0.20), E_2 (0.12)

2 Related Work (in Appendix A.2)

3 Token-Aware Representation Editing

This section introduces the proposed TARE method. Rather than using dense low-rank adapters, TARE employs a lightweight, token-wise selector. For each token, it activates a small set of hidden representation editors (per-feature scaling and bias) and mixes their edits with normalized weights. This token-aware, k-sparse, diagonal adjustment increases expressiveness and captures fine-grained context. Although it uses token-wise Top-k routing, it is not an MoE-based adapter: the routed units are lightweight diagonal post-FFN editors rather than expert FFN sub-networks. It adds a small additional inference overhead and avoids rank/scale hyperparameters. As a result, TARE transfers well across diverse tasks while alleviating the extra computation and overfitting issues of conventional fine-tuning.

3.1 Overall Design

The proposed TARE method augments the hidden representation editor with a lightweight token-aware selector, as shown in Figure 2. At each token position, the selector activates a small subset of hidden representation editors, each providing its own per-feature scaling and bias; they are then linearly combined with normalized weights. This multi-path yet k-sparse design enables flexible and efficient token-wise adjustment, enhancing adaptability across heterogeneous tasks while keeping the added inference overhead small in practice. Please refer to Appendix A.3 for the detailed design principles.

Table 2: Comparison of Accuracy (%) on the Knowledge Reasoning Task Under the LLaMA-3-8B Model. The Knowledge Reasoning task contains eight test datasets: BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA. We reuse the results for LoRA, DoRA, and LoReFT reported in (Wu et al., 2024b), and the results for MiLoRA reported in (Wang et al., 2024a).

PEFT	Source	Params.(%)	VRAM(MiB)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
LoRA	ICLR' 21	0.7002	21 828	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
DoRA	ICML' 24	0.7098	41 780	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
MiLoRA	NAACL' 25	0.7002	21 580	68.8	86.7	77.2	92.9	85.6	86.8	75.5	81.8	81.9
LoReFT	NeurIPS' 24	0.0260	21 050	75.1	**90.2**	82.0	**96.3**	87.4	**92.4**	81.6	87.5	86.6
RED	ACL' 24	0.0033	20 132	68.0	83.7	79.7	90.0	83.2	85.2	72.8	79.4	80.2
PiSSA	NeurIPS' 24	0.7002	21 004	72.1	89.2	82.7	94.6	89.6	86.8	**84.5**	85.2	85.6
Spectral Adapter	arXiv' 24	0.7002	21 746	72.1	88.3	**83.1**	94.6	89.3	85.4	82.2	85.2	85.0
LoRA-GA	NeurIPS' 24	0.7002	21 708	72.5	88.8	82.7	94.4	89.6	91.3	80.4	85.6	85.7
LoRA-One	arXiv' 25	0.7002	21 206	72.0	88.9	82.9	94.4	**89.8**	85.1	82.6	87.6	85.4
TARE (ours)	This paper	0.0392	21 724	**75.2**	**90.2**	82.5	94.1	88.6	91.3	82.3	**88.4**	**86.7**

Table 3: Comparison of Accuracy (%) on the Mathematical Reasoning Task Under the LLaMA-3-8B and Qwen-2.5-7B-Instruct Models. The Mathematical Reasoning task contains seven test datasets: MultiArith, GSM8K, SVAMP, MAWPS, AddSub, AQuA, and SingleEq.

PEFT	Source	Model	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
LoRA	ICLR' 21		0.2345	21 070	92.0	61.4	69.8	84.2	85.4	**44.7**	90.3	75.4
DoRA	ICML' 24		0.2361	29 284	91.7	59.0	72.3	82.1	86.1	39.9	89.5	74.4
MiLoRA	NAACL' 25		0.2345	21 520	91.7	59.0	70.5	88.3	86.1	43.4	90.5	75.6
LoReFT	NeurIPS' 24		0.0260	21 940	89.2	56.2	68.7	80.3	90.1	33.1	90.0	72.5
RED	ACL' 24	LLaMA-3-8B	0.0033	19 852	91.0	54.2	66.8	81.1	87.3	34.1	90.9	72.2
PiSSA	NeurIPS' 24		0.2345	21 417	91.5	**61.8**	69.3	**89.5**	86.9	42.2	90.7	76.0
Spectral Adapter	arXiv' 24		0.2345	21 892	91.0	59.7	71.0	88.7	86.9	43.3	90.9	75.9
LoRA-GA	NeurIPS' 24		0.2345	21 156	92.2	60.0	72.0	89.1	86.7	42.4	90.1	76.1
LoRA-One	arXiv' 25		0.2345	21 739	91.8	59.7	71.9	88.7	86.9	42.6	90.1	76.0
TARE (ours)	This paper		0.0392	20 900	**95.8**	57.3	**72.9**	86.1	**90.9**	41.4	**92.1**	**76.7**
LoRA	ICLR' 21		0.2643	21 244	94.7	72.8	81.1	89.4	88.4	66.5	91.7	83.5
DoRA	ICML' 24		0.2657	29 604	93.2	72.1	79.8	88.2	89.7	63.7	92.7	82.8
MiLoRA	NAACL' 25		0.2643	21 518	93.3	72.2	80.8	88.3	89.6	69.6	92.7	83.8
LoReFT	NeurIPS' 24		0.0218	21 832	92.1	71.7	78.5	86.2	90.0	67.3	90.5	82.3
RED	ACL' 24	Qwen-2.5-7B-Instruct	0.0026	20 100	91.3	71.3	77.4	84.1	**90.4**	**70.9**	88.2	81.9
PiSSA	NeurIPS' 24		0.2643	21 568	94.3	72.4	**83.1**	88.8	87.6	63.6	91.1	83.0
Spectral Adapter	arXiv' 24		0.2643	21 957	93.8	70.2	82.0	89.4	88.9	67.2	93.1	83.5
LoRA-GA	NeurIPS' 24		0.2643	21 244	93.5	70.2	80.4	89.3	89.7	67.5	92.7	83.3
LoRA-One	arXiv' 25		0.2643	21 819	94.5	71.5	81.5	89.4	88.4	66.6	**93.5**	83.6
TARE (ours)	This paper		0.0316	20 624	**96.0**	**75.1**	80.3	**92.4**	**90.4**	63.6	91.3	**84.2**

For every layer, TARE attaches n hidden representation editors, each with an independent parameter set for editing operations (element-wise scaling and bias by default, extensible to other simple transforms). During the forward pass, a Top-k mechanism selects the k most relevant hidden representation editors conditioned on the current activation, and the final representation is obtained by a weighted combination of their edits.

To make the token-wise routing and editing mechanism more concrete, we provide a small illustrative example before introducing the full notation. Consider the math word problem "Edgar eats 18 pretzels a day. If his brother eats 1/2 as many, how many does his brother eat in a week?" Suppose TARE uses $n = 8$ hidden representation editors and keeps the Top-$k = 3$ editors for each token. For each token, the selector produces logits over the editor pool, the Top-3 logits are normalized into mixture weights, and the corresponding diagonal editors are then combined. Table 1 shows a simplified routing example. The editor indices are arbitrary and are used only to illustrate the mechanism rather than to assign semantic labels to particular editors. For readers familiar with sparse routing, note that TARE routes over lightweight diagonal post-FFN editors rather than expert FFN blocks. This example highlights the intuition behind TARE. Tokens such as numbers, fractions, and time units (e.g., "18", "1/2", and "day/week") may concentrate more on a subset of editors (here, E_6/E_3), while punctuation and more generic functional tokens may place more mass on shared editors (here, E_1). The key point is not the particular editor identity, but that different token roles can favor different convex mixtures of lightweight editors, which is precisely the behavior that TARE is designed to capture.

The proposed TARE method consists of three main steps: Token-Aware Selection, Top-k Activation, and Hidden Representation Editing and Aggregation, which are articulated in the following sections.

3.2 Token-Aware Selection

Let the hidden representation of a given token t at layer ℓ be $h_{\ell,t} \in \mathbb{R}^{1\times1\times D_\ell}$. TARE first applies a

Table 4: Performance Comparison Under the LLaMA-3-8B Model. The BLEU, NIST, METEOR, ROUGE–L, and CIDEr metrics are used for dataset E2E-Challenge.

PEFT	Source	Params.(%)	VRAM(MiB)	BLEU↑	NIST↑	METEOR↑	ROUGE–L↑	CIDEr↑
LoRA	ICLR' 21	0.2345	39 166	0.6255	8.2791	0.4404	0.6661	2.1524
DoRA	ICML' 24	0.2361	45 326	0.6201	8.1455	0.4367	0.6617	2.1578
MiLoRA	NAACL' 25	0.2345	39 590	0.6244	8.2652	0.4283	0.6606	2.1845
LoReFT	NeurIPS' 24	0.0260	32 502	0.5719	7.5671	0.4304	0.6431	1.6881
RED	ACL' 24	0.0033	29 492	0.5994	7.9229	0.4401	0.6692	2.1958
TARE (ours)	This paper	0.0392	34 626	**0.6333**	**8.3105**	**0.4456**	**0.6758**	**2.2027**

Table 5: Comparison of Accuracy (%) on the GLUE Task Under the RoBERTa-base and RoBERTa-large Models. The GLUE task contains eight test datasets: MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B. We reuse the results for LoRA, Adapter-FFN, BitFit, IA^3, and RED reported in (Wu et al., 2024a).

PEFT	Source	RoBERTa	Params.(M)	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
LoRA	ICLR' 21	base	0.29	86.6	93.9	88.7	59.7	**92.6**	**90.4**	75.3	90.3	84.7
Adapter-FFN	EMNLP' 20	base	0.30	**87.1**	93.0	88.8	58.5	92.0	90.2	77.7	90.4	84.7
BitFit	ACL' 22	base	0.10	84.7	**94.0**	88.1	54.0	91.0	87.3	69.8	89.5	82.3
IA^3	NeurIPS' 22	base	0.06	85.4	93.4	86.4	57.8	91.1	88.5	73.5	88.5	83.1
RED	ACL' 24	base	0.02	83.9	93.9	89.2	**61.0**	90.7	87.2	**78.0**	90.4	84.3
TARE (ours)	This paper	base	0.22	86.3	93.1	**91.5**	58.6	91.7	88.6	77.8	**90.6**	**84.8**
LoRA	ICLR' 21	large	0.79	90.2	96.0	89.8	65.5	**94.7**	90.7	**86.3**	91.7	88.1
Adapter-FFN	EMNLP' 20	large	0.80	**90.3**	**96.1**	90.5	64.4	94.3	**91.3**	84.8	90.2	87.7
IA^3	NeurIPS' 22	large	0.15	90.1	94.5	87.1	63.2	93.9	89.3	85.3	91.5	86.9
RED	ACL' 24	large	0.05	89.5	96.0	90.3	**68.1**	93.5	88.8	86.2	91.3	87.9
TARE (ours)	This paper	large	0.59	90.0	94.5	**92.3**	67.9	94.6	89.4	85.5	**92.1**	**88.3**

token-wise selector: a small feed-forward network that produces a real-valued score for each of the n candidate hidden representation editors. Formally,

$$h_{\ell,t}^{\text{new}} = \text{selector}(h_{\ell,t}) \quad \in \mathbb{R}^{1\times 1\times n}. \tag{1}$$

The selector uses one linear layer and is kept narrow so its parameter footprint remains negligible. Intuitively, it scores token–editor compatibility, playing a role analogous to sparse routing while keeping the backbone frozen; unlike in MoE, however, it does not replace the FFN with expert subnetworks.

3.3 Top-k Activation

To avoid activating all n hidden representation editors and increasing compute, the proposed TARE method keeps only the k highest-scoring hidden representation editors per token ($k \ll n$, e.g., $k = 3$):

$$\big(\text{topk\_values},\ \text{topk\_indices}\big) = \text{Top-k}\big(h_{\ell,t}^{\text{new}},\ k\big). \tag{2}$$

The selected logits are then normalized with a softmax (along the last dimension) to obtain a probabilistic selection mask:

$$w = \text{softmax}\big(\text{topk\_values},\ -1\big), \tag{3}$$

so that $\sum_{i=1}^{k} w_{\ell,t,i} = 1$ for every token. This sparse selection keeps the additional latency modest relative to the original model, because the cost of processing k lightweight hidden representation editors is small compared with the backbone's already-computed attention and feed-forward layers.

The selector's Top-k routing can collapse (most tokens routed to a few editors), which hurts both stability and capacity usage. We add a lightweight auxiliary term on the selector probabilities, which encourages balanced utilization across editors, stabilizes training, and yields consistent accuracy gains. A fuller discussion is provided in Appendix A.5.

3.4 Hidden Representation Editing and Aggregation

Each hidden representation editor i maintains its own pair of element-wise scaling and bias vectors $\gamma_{\ell,i}, b_{\ell,i} \in \mathbb{R}^{1\times 1\times D_\ell}$, trained from scratch while the backbone remains frozen. For each selected hidden representation editor, the proposed TARE method computes a candidate edit

$$h_{\ell,t,i} = h_{\ell,t} \odot \gamma_{\ell,i} + b_{\ell,i}, \tag{4}$$

where $\odot$ denotes the Hadamard (element-wise) product. Because these operations are diagonal in feature space, they introduce no additional matrix multiplications and can be fused into a single CUDA kernel in practical implementations. Finally, the k token-specific hidden representation editors

Table 6: Performance Comparison Under the LLaMA-3-8B Model. The Pass@1 rate (%) is used for datasets HumanEval and MBPP. The accuracy (%) metric is used for datasets ScienceQA and CoinFlip.

PEFT	Source	Params.(%)	VRAM(MiB)	HumanEval	MBPP	ScienceQA	CoinFlip
LoRA	ICLR' 21	0.7002	23 038	31.3	46.0	92.6	50.8
DoRA	ICML' 24	0.7098	44 382	25.0	42.0	92.0	55.8
MiLoRA	NAACL' 25	0.7002	23 478	37.5	40.0	92.9	**57.1**
LoReFT	NeurIPS' 24	0.0260	20 116	43.8	42.0	92.4	53.5
RED	ACL' 24	0.0033	18 762	25.0	46.0	93.4	50.5
TARE (ours)	This paper	0.0392	20 008	**56.3**	**48.0**	**94.5**	**57.1**

Table 7: Comparison of Accuracy (%) on WikiFact Dataset Under the LLaMA-3-8B Model. The WikiFact dataset contains five relation domains: jurisdiction, country, capital, capital_of, and continent.

PEFT	Source	Params.(%)	VRAM(MiB)	jurisdiction	country	capital	capital_of	continent	Avg.
LoRA	ICLR' 21	0.7002	17 676	77.0	58.0	40.0	39.0	82.0	59.0
DoRA	ICML' 24	0.7098	36 712	75.0	56.0	39.0	41.0	83.0	59.0
MiLoRA	NAACL' 25	0.7002	18 710	75.0	66.0	39.0	39.0	80.0	60.0
LoReFT	NeurIPS' 24	0.0260	18 492	83.0	65.0	51.0	39.0	85.0	65.0
RED	ACL' 24	0.0033	16 352	82.0	62.0	46.0	40.0	83.0	63.0
TARE (ours)	This paper	0.0392	16 512	**86.0**	**69.0**	**55.0**	**41.0**	**86.0**	**67.0**

Table 8: Comparison of First Turn Score on MT-Bench Dataset Under the LLaMA-2-7B Model. We reuse the results for LoRA, PiSSA, rsLoRA, and LoRA+ reported in (Wang et al., 2024b).

PEFT	Source	Params.(%)	First Turn Score
LoRA	ICLR' 21	0.2970	5.61 ± 0.10
PiSSA	NeurIPS' 24	0.2970	5.30 ± 0.02
rsLoRA	arXiv' 23	0.2970	5.25 ± 0.03
LoRA+	ICML' 24	0.2970	5.71 ± 0.08
TARE (ours)	This paper	0.0467	$\mathbf{5.73 \pm 0.05}$

are linearly combined according to their selection weights to yield the updated representation

$$h_{\ell,t}^{\text{update}} = \sum_{i=1}^{k} h_{\ell,t,i}\, w_{\ell,t,i}. \tag{5}$$

This convex combination acts as a soft winner-take-all mechanism: hidden representation editors that the selector deems most relevant contribute the most, while others are softly suppressed.

In summary, the proposed TARE method adds a lightweight, token-aware, k-sparse hidden representation editor that lifts the representational ceiling of simple scaling/bias edits while keeping the backbone frozen. By conditionally routing and mixing a few per-feature edits per token, it attains high expressiveness and contextual adaptivity with a small additional inference overhead.

4 Experiment

We conduct a comprehensive study on the decoder-only model LLaMA-3-8B and the encoder models RoBERTa-base and RoBERTa-large. Our evaluation covers nine task families, including knowledge reasoning, mathematical reasoning, GLUE, conditional text generation, code synthesis, knowledge completion, closed-book question answering, symbolic reasoning, and instruction following. We compare against representative PEFT baselines, including LoRA, DoRA, MiLoRA, LoReFT, and RED; for GLUE, we additionally include Adapter-FFN, IA^3, and BitFit. Our ablation study isolates the effects of scaling-only versus bias-only editing. Our sensitivity analysis varies the number of hidden representation editors n and the number of selected editors k, quantifying the expressiveness–efficiency trade-off. In addition, a visual analysis examines layer-wise load-balancing behavior, showing how the auxiliary loss equalizes editor utilization and often coincides with accuracy gains. Additional details on datasets, baselines, implementation details, efficiency analysis, and interpretability analysis are provided in Appendix A.6, A.7, A.8, A.11, and A.12, respectively.

4.1 Overall Performance

TARE achieves strong performance across diverse evaluation settings, including conditional text generation, knowledge reasoning, mathematical reasoning, instruction following, GLUE, code synthesis, knowledge completion, closed-book QA and symbolic reasoning. Meanwhile, it tunes only 0.0392% of the trainable parameters and maintains a low GPU-memory footprint, with only a small additional inference overhead; on MultiArith, the

mean per-example latency is 3.20 s for TARE versus 2.68 s for merged LoRA. Representative results include: best performance on E2E across all reported metrics; the highest Pass@1 rate on HumanEval/MBPP; 86.7% average on Commonsense; 76.7% average on Math-10K; and 88.3% on GLUE. Overall, TARE matches or surpasses widely used PEFT baselines, including LoRA, DoRA, MiLoRA, LoReFT, and RED, on most benchmarks.

4.1.1 Knowledge Reasoning

Table 2 shows that TARE achieves the best average accuracy (86.7) across eight knowledge reasoning benchmarks on LLaMA-3-8B, slightly exceeding LoReFT (86.6). Rather than relying on a large low-rank update, TARE attains this performance with only 0.0392% trainable parameters (about 17.9× fewer than LoRA-style baselines at 0.7002%), while keeping training memory practical (21,724 MiB; far below DoRA's 41,780 MiB). This result supports our design intuition that commonsense reasoning benefits from token-aware, lightweight representation calibration: by routing tokens to a small set of hidden representation editors, TARE can apply targeted per-token adjustments without globally reshaping the backbone, yielding strong generalization under a tight adaptation budget. Consistent with this interpretation, TARE shows clear advantages over most PEFT baselines (e.g., +5.9 over LoRA and +6.5 over RED in average accuracy), while remaining competitive with the strongest representation-editing baseline.

4.1.2 Mathematical Reasoning

Table 3 reports results on seven mathematical reasoning benchmarks for LLaMA-3-8B and Qwen-2.5-7B-Instruct. TARE achieves the best macro-average accuracy on both backbones while tuning only a tiny fraction of parameters (0.0392% on LLaMA-3-8B and 0.0316% on Qwen-2.5-7B-Instruct) with about 20–21 GiB peak GPU memory during training. These gains are consistent with TARE's design: mathematical reasoning requires different tokens to play distinct roles (e.g., numbers, operators, intermediate variables, and reasoning cues), and a token-aware editor can selectively calibrate such heterogeneous representations rather than applying a single global update. As a result, TARE improves the overall reasoning behavior without heavily reshaping the backbone, which helps explain why it remains strong under an extremely small update space. To control for insertion location, all PEFT methods are applied only to the FFN projection matrices in each MLP block of the backbone LM.

4.1.3 Conditional Text Generation

On the E2E NLG benchmark, TARE achieves the best conditional generation performance with LLaMA-3-8B (Table 4). We attribute the consistent gains to TARE's token-aware editing: data-to-text generation involves heterogeneous token roles (e.g., slot values vs. function words), and selectively calibrating token representations via a small set of hidden representation editors improves both fidelity and fluency. Accordingly, TARE outperforms strong PEFT baselines on all automatic metrics, e.g., BLEU 0.6333 and NIST 8.3105. These improvements are obtained with a lightweight adaptation budget, tuning only 0.0392% parameters and using 34,626 MiB peak VRAM. To control for insertion location, all PEFT methods are applied only to the FFN projection matrices in each MLP block of the backbone LM.

4.1.4 GLUE

Table 5 shows that TARE achieves the best average GLUE score on both RoBERTa-base (84.8) and RoBERTa-large (88.3). It performs particularly well on MRPC and STS-B, and remains strong on MNLI and QNLI, which is consistent with TARE's token-aware, per-dimension editing that refines local semantic alignment while preserving pretrained hidden representations for multi-task transfer. Despite weaker results on a few tasks (e.g., CoLA and QQP), TARE yields the highest overall average. Moreover, it delivers these gains with fewer trainable parameters (0.22M/0.59M) than LoRA (0.29M/0.79M) and Adapter-FFN (0.30M/0.80M) under the same settings.

4.1.5 Code Synthesis

As shown in Table 6, TARE achieves the best Pass@1 rate on both HumanEval (56.3) and MBPP (48.0), with a particularly large gain on HumanEval (+12.5 percentage points over LoReFT). We attribute this to TARE's token-aware editing: it routes each token to a small set of hidden representation editors, enabling specialized adjustments for heterogeneous code tokens and long-range reasoning traces. Meanwhile, the per-dimension scaling and bias provide fine-grained calibration that is critical for code generation, where small representation shifts can affect syntactic and semantic correctness.

4.1.6 Knowledge Completion

Table 7 shows that TARE achieves the best average accuracy on WikiFact (67.0). We attribute the gains to TARE's token-aware editing, which selectively calibrates salient entity and relation tokens via a small set of hidden representation editors without globally perturbing the backbone. TARE performs best on four relations and ties for the best on capital_of, indicating robust improvements across relation types. Overall, it yields absolute gains of 8.0 percentage points over LoRA/DoRA, 7.0 over MiLoRA, 4.0 over RED, and 2.0 over LoReFT, while tuning only 0.0392% parameters and using 16,512 MiB peak GPU memory during training.

4.1.7 Closed-Book QA and Symbolic Reasoning

Table 6 shows that TARE performs strongly on both closed-book QA and symbolic reasoning, reaching 94.5% on ScienceQA and 57.1% on CoinFlip while tuning only 0.0392% parameters with ∼20 GiB peak training VRAM. We attribute these gains to TARE's token-aware hidden representation editing: closed-book QA benefits from selectively calibrating salient knowledge-bearing tokens for more reliable recall, whereas symbolic tasks such as CoinFlip require stable, discrete decision patterns that are sensitive to small representation shifts. Consistent with this interpretation, TARE improves over strong baselines (e.g., +1.1 percentage points over RED on ScienceQA and +6.3 over LoRA on CoinFlip) and ties the best CoinFlip result, indicating that lightweight per-token, per-dimension adjustments can enhance both factual retrieval and rule-following behavior.

4.1.8 Instruction Following

Table 8 reports MT-Bench results for LLaMA-2-7B instruction-tuned on WizardLM and evaluated by GPT-4. TARE achieves the best First Turn Score with only 0.0467% trainable parameters, it reaches 5.73 ± 0.05, surpassing LoRA+ (5.71 ± 0.08) and other baselines that all tune 0.297%. We attribute this gain to TARE's token-aware editing, which selectively calibrates instruction-critical tokens without globally perturbing the model, improving first-turn reliability in multi-turn dialogue while keeping adaptation cost low.

4.2 Ablation Study

We conduct ablations on TARE's key design choices: the scaling/bias editor, the load-balancing auxiliary loss, and the insertion position, to validate their respective contributions to performance and efficiency.

4.2.1 Component ablation

Table 10 shows that TARE achieves the best overall performance (76.7) while tuning only 0.0392% parameters with about 20,900 MiB peak VRAM, and it clearly outperforms the ablated variants without scaling (50.5) or without bias (56.4). This large drop indicates that both components are essential for effective representation editing: per-dimension scaling controls feature magnitudes and stabilizes activation dynamics, whereas per-dimension bias corrects systematic offsets and improves decision boundaries.

4.2.2 Load-balancing ablation

Adding the load-balancing auxiliary loss (Appendix A.5) improves the average accuracy from 75.8 to 76.7, while keeping the trainable ratio fixed at 0.0392% and VRAM nearly unchanged (Table 11). This gain supports our motivation for the auxiliary loss: without it, the top-k selector tends to collapse onto a few editors, leaving others under-utilized and reducing effective capacity. As shown in Figure 3, in the 16th block the selection distribution shifts from highly skewed (one editor rarely chosen; others $\sim 7.3\times10^5$–1.16×10^6) to near-uniform usage across all eight editors ($\sim 7.6\times10^5$–8.2×10^5 each). By encouraging balanced routing, the model can apply specialized hidden representation edits to a broader range of tokens, yielding more consistent improvements.

4.2.3 Position ablation

Table 12 compares inserting TARE at seven locations (q/k/v/o and up/gate/down) in LLaMA-3-8B, revealing where lightweight token-wise edits are most effective. Editors on q or k are the most parameter- and memory-efficient, yet they consistently underperform, suggesting that merely perturbing query/key projections does not sufficiently propagate to downstream behavior. In contrast, moving the insertion to v and then o yields steadily better results, indicating that editing value aggregation and attention outputs more directly shapes the information written into the residual stream, albeit with higher VRAM. For the FFN, inserting at up can be competitive but is substantially more expensive (0.1369% trainable parameters and ∼29.6 GiB), while gate is weaker overall, likely

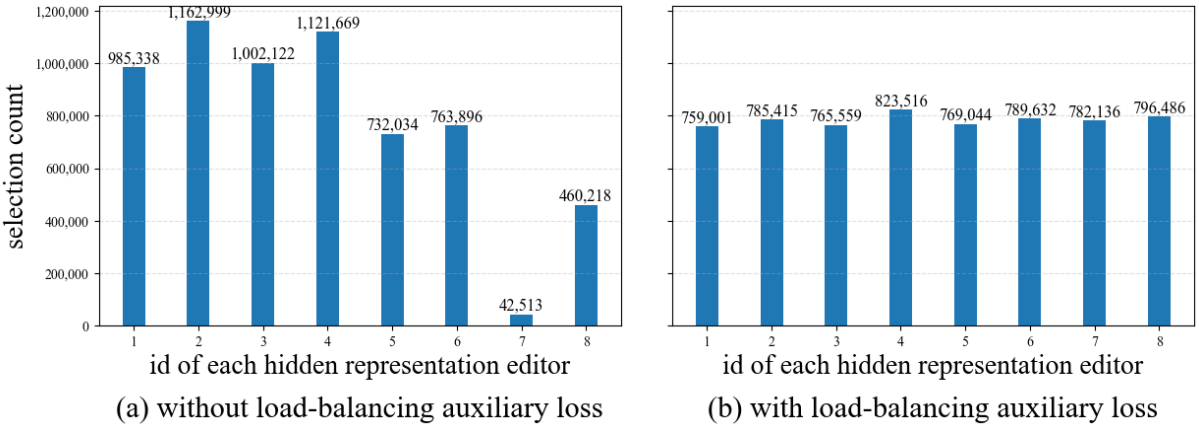


Figure 3: Effect of load balancing on editor utilization. Each bar shows how many times an editor in the 16th Transformer block of LLaMA-3-8B is selected (eight editors, Top-k=3).

because it only indirectly modulates activated channels. Overall, the default down insertion provides the best accuracy–efficiency trade-off, achieving the highest average accuracy with only 0.0392% trainable parameters and ∼20.9 GiB VRAM.

4.3 Sensitivity Analysis

We examine the robustness of TARE, showing that its performance remains stable under reasonable choices of k, n, and training-set size.

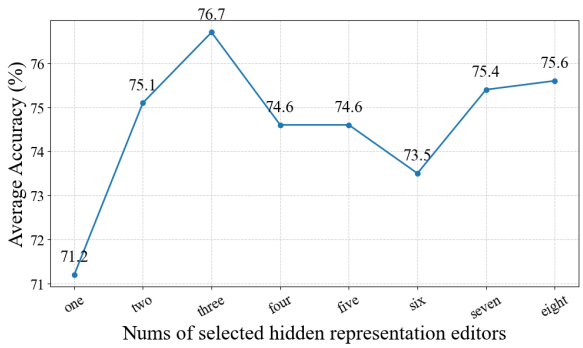


Figure 4: Sensitivity to Number of Selected Editors. Average accuracy on seven math-reasoning datasets (trained on Math-10K) as we vary the count of token-wise selected hidden representation editors from one to eight. Accuracy rises sharply from one to three and peaks at three (76.7%). Larger selections show diminishing returns and fluctuate within 73.5–75.6%.

4.3.1 Hyperparameter sensitivity

TARE achieves the best average accuracy with k=3 selected editors (76.7; Figure 4, Table 13), improving sharply from k=1 to k=3 (71.2 → 76.7). This supports sparse token-wise editing: small k underfits diverse token patterns, whereas larger k brings diminishing returns due to redundant corrections. With the trainable ratio fixed at 0.0392%, VRAM increases only modestly as k grows (∼20.2 GiB→ ∼22.8 GiB), so we use k=3 by default. Varying the editor pool size $n \in \{6, 8, 10\}$ yields stable results with a mild peak at n=8 (Table 14), hence we set n=8 as default.

4.3.2 Sample sensitivity

Table 15 shows that TARE is robust to the amount of Math-10K supervision under a fixed configuration. Performance improves smoothly as the training set grows (69.8 average with 500 examples, 74.3 at 2k, 76.2 at 5k, and 76.7 with all 9,919), without large fluctuations. This trend supports our design: the token-wise selector and hidden representation editors can capture reusable reasoning patterns even from limited data, and additional supervision mainly refines these edits rather than changing the adaptation behavior.

5 Conclusion

We presented TARE, a lightweight PEFT approach that replaces one-size-fits-all edits with per-token, per-dimension adjustments. A lightweight selector activates a few hidden representation editors per token, enabling fine-grained adaptation with a small additional inference overhead and no rank or scale hyperparameters. Extensive experiments validate TARE's benefits on both decoder and encoder families, while tuning only 0.0392% of parameters and using about 20 GiB of GPU memory, matching or surpassing LoRA, DoRA, MiLoRA, LoReFT, and RED across many settings.

6 Limitations

We have shown that the proposed TARE method can be an efficient and effective PEFT approach by enabling token-aware representation editing with sparse selection over a small set of hidden representation editors across diverse tasks and backbones. Nevertheless, TARE currently relies on a top-k routing mechanism whose stability and efficiency may vary across layers, tasks, and sequence lengths; in particular, selector collapse or uneven editor utilization can degrade robustness and may require auxiliary balancing losses and careful hyperparameter tuning of (n, k). Moreover, our editors are restricted to per-dimension scaling and bias, which provides strong efficiency but may limit expressivity for tasks requiring structured, higher-order transformations of hidden states. Finally, while TARE introduces a small but nonzero inference overhead in standard settings, its token-wise selection can become less favorable under extremely long-context inference or strict latency constraints. Exploring more expressive yet lightweight editor parameterizations, more stable routing strategies, and extensions to other modalities (e.g., vision, speech, and video) remains an important direction for future work.

7 Ethical Considerations

This paper proposes TARE, a parameter-efficient fine-tuning method for large language models. The method itself does not introduce new data collection or deployment practices. However, like other techniques that improve adaptation efficiency, TARE may lower the barrier to customizing models, which can be used for both beneficial applications and harmful purposes (e.g., generating misleading or abusive content). We therefore emphasize that TARE should be used in compliance with applicable policies and safety guidelines, and we encourage practitioners to pair fine-tuned models with standard safeguards such as data filtering, content moderation, and responsible release practices. We do not use personal data in our experiments beyond what is contained in the benchmark datasets, and we follow their original licenses and usage conditions.

8 Acknowledgments

This work was supported in part by the Beijing Municipal Science and Technology Project (No. Z231100010323002), National Natural Science Foundation of China (Grant No. 62472042), and Fundamental Research Funds for the Beijing University of Posts and Telecommunications (No. 2025TSQY01).

References

Jacob Austin, Augustus Odena, and Maxwell Nye. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Yonatan Bisk, Rowan Zellers, and Ronan Le Bras. 2019. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.

Mark Chen, Jerry Tworek, and Heewoo Jun. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Christopher Clark, Kenton Lee, and Ming-Wei Chang. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044.

Peter Clark, Isaac Cowhey, and Oren Etzioni. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, and Mohammad Bavarian. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*. Introduces the GSM8K dataset.

Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

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

Ben Goodrich, Vinay Rao, and Peter J Liu. 2019. Assessing the factual accuracy of generated text. In *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 166–175.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, and Oren Etzioni. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Introduces the AddSub dataset.

J. Edward Hu, Yelong Shen, and Phillip Wallis. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Zhiqiang Hu, Yihuai Lan, and Lei Wang. 2023a. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *ArXiv*, abs/2304.01933.

Zhiqiang Hu, Lei Wang, and Yihuai Lan. 2023b. 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, Singapore. Introduces the Math10K dataset.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, and Ashish Sabharwal. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597. Introduces the SingleEq dataset.

Rik Koncel-Kedziorski, Subhro Roy, and Aida Amini. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California.

Wang Ling, Dani Yogatama, and Chris Dyer. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Introduces the AQuA-RAT dataset.

Haokun Liu, Derek Tam, and Mohammed Muqeeth. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Shih-Yang Liu, Chien-Yi Wang, and Hongxu Yin. 2024. Dora: Weight-decomposed low-rank adaptation. *ArXiv*, abs/2402.09353.

Yinhan Liu, Myle Ott, and Naman Goyal. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Todor Mihaylov, Peter Clark, and Tushar Khot. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *ArXiv*, abs/1706.09254.

OpenAI, Josh Achiam, and Steven Adler. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Introduces the SVAMP dataset.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal.

Tanik Saikh, Tirthankar Ghosal, and Amish Mittal. 2022. Scienceqa: a novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23:289 – 301.

Keisuke Sakaguchi, Ronan Le Bras, and Chandra Bhagavatula. 2019. An adversarial winograd schema challenge at scale.

Maarten Sap, Hannah Rashkin, and Derek Chen. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Alex Wang, Amanpreet Singh, and Julian Michael. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hanqing Wang, Zeguan Xiao, and Yixia Li. 2024a. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *North American Chapter of the Association for Computational Linguistics*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024b. Lora-ga: Low-rank adaptation with gradient approximation. *ArXiv*, abs/2407.05000.

Shaowen Wang, Linxi Yu, and Jian Li. 2024c. Lora-ga: Low-rank adaptation with gradient approximation. In *NeurIPS*.

Jason Wei, Xuezhi Wang, and Dale Schuurmans. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Muling Wu, Wenhao Liu, and Xiaohua Wang. 2024a. Advancing parameter efficiency in fine-tuning via representation editing. *ArXiv*, abs/2402.15179.

Zhengxuan Wu, Aryaman Arora, and Zheng Wang. 2024b. Reft: Representation finetuning for language models. *ArXiv*, abs/2404.03592.

Can Xu, Qingfeng Sun, and Kai Zheng. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *ICLR*.

Rowan Zellers, Ari Holtzman, and Yonatan Bisk. 2019. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*.

Fangzhao Zhang and Mert Pilanci. 2024. Spectral adapter: Fine-tuning in spectral space. *ArXiv*, abs/2405.13952.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. 2025. Lora-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. *arXiv*, abs/2502.01235.

Lianmin Zheng, Wei-Lin Chiang, and Ying Sheng. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Appendix

A.1 Use of LLMs

We used large language models strictly for editorial assistance, including spell checking, grammar polishing, and minor wording suggestions for the paper text. No model outputs were used to create, modify, or label datasets; implement algorithms; tune hyperparameters; or select results. All technical content (methods, proofs, experiments, and numbers) was written and verified by the authors, and every LLM-suggested edit was reviewed manually for accuracy and clarity.

A.2 Related Work

Parameter-Efficient Fine-Tuning and Representation Editing. PEFT aims to adapt large Transformers by training only a tiny fraction of parameters while freezing the backbone. Low-rank adapters such as LoRA (Hu et al., 2021) inject rank-r updates into weight matrices; in standard single-adapter deployments these increments are merged into the base weights, so there is effectively no additional inference overhead. They still require nontrivial choices of rank and scaling, which can complicate tuning across layers and tasks. When merging is not desirable (e.g., multi-adapter hot-swap, online mixture/selection, or coexistence with certain quantization pipelines), one may resort to on-the-fly composition that re-introduces extra operators, but this is an engineering choice rather than an inherent property of LoRA-style methods. DoRA (Liu et al., 2024) decouples direction and magnitude to stabilize optimization while remaining low-rank; MiLoRA (Wang et al., 2024a) modifies singular subspaces to reduce redundancy in LoRA updates. A complementary line edits hidden representations directly: RED (Wu et al., 2024a) learns a single global diagonal scaling/bias with near-zero inference cost but limited contextual adaptivity; LoReFT (Wu et al., 2024b) performs low-rank projections in representation space but applies the same projection to every token and inherits rank selection. We follow representation editing but replace one-size-fits-all edits with token-aware diagonal modulation, retaining the efficiency of feature-wise operations while addressing the lack of per-token expressiveness observed in global edits and uniform low-rank mappings.

Token-Aware Conditional Modulation and Dynamic Editing. For encoder models, widely used PEFT baselines include LoRA and RED as above, together with IA^3 (Liu et al., 2022) and BitFit (Ben Zaken et al., 2021). IA^3 gates attention/FFN channels via learned per-feature multipliers, and BitFit updates only biases; both are extremely lightweight but share a uniform modulation across tokens, limiting fine-grained adaptivity. RED is inference-friendly but globally shared. In contrast, the proposed TARE method performs token-aware, diagonal representation editing: for each token it mixes a few learned diagonal templates to yield per-token, per-dimension adjustments while incurring only a small additional inference overhead. This design directly targets the expressiveness–efficiency tension highlighted by these baselines.

Relation to Mixture-of-Experts (MoE). Similarities. TARE borrows two well-established ideas from the MoE literature (Fedus et al., 2022): (i) token-wise sparse routing, where each token is routed to a small subset (Top-k) of candidates; and (ii) an auxiliary load-balancing loss that encourages the average routing distribution to be close to uniform, preventing collapse of routing to only a few choices. In our implementation the selector produces token-level scores over n candidates and activates k of them, and we use a KL-to-uniform load-balancing term (weight λ=0.02) to distribute traffic across candidates. Key differences. Despite these conceptual overlaps, TARE is not an MoE-based adapter or an FFN replacement. In classic MoE, each expert is a full feed-forward subnetwork that replaces the FFN block for routed tokens, incurring additional matrix multiplications, parameters, capacity management, and dispatch overhead. By contrast, TARE's routed units are lightweight diagonal hidden-representation editors applied after the FFN within a PEFT regime. The backbone remains frozen, no FFN is duplicated or replaced, and the added inference overhead is small rather than zero. Practically, TARE performs a convex mixture of a few diagonal edits for each

Table 9: Overview of the tasks, datasets (training/test splits), and evaluation metrics used in our experiments.

task	training set	test set	metrics
Knowledge Reasoning	Commonsense-170K	BoolQ, PIQA, SIQA HellaSwag, WinoGrande ARC-e/c, OBQA	accuracy
Mathematical Reasoning	Math-10K	MultiArith, GSM8K, SVAMP, MAWPS, AddSub, AQuA, SingleEq	accuracy
GLUE	MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, STS-B train	MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, STS-B test	Matthews Correlation F1, accuracy Pearson, Spearman
Conditional Text Generation	E2E-Challenge train	E2E-Challenge test	BLEU, NIST, METEOR, ROUGE–L, CIDEr
Code Synthesis	HumanEval, MBPP test (90%)	HumanEval, MBPP test (10%)	Pass@1 rate
Knowledge Completion	WikiFact train	WikiFact test	accuracy
Closed-Book QA	ScienceQA train	ScienceQA test	accuracy
Symbolic Reasoning	CoinFlip train	CoinFlip test	accuracy
Instruction Following	WizardLM	MT-Bench	First Turn Score

Table 10: Component ablation of TARE Under the LLaMA-3-8B Model.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE (ours)	0.0392	20 900	**95.8**	**57.3**	**72.9**	**86.1**	**90.9**	**41.4**	**92.1**	**76.7**
TARE (w/o scaling)	0.0261	19 348	43.7	28.4	56.1	52.9	71.4	31.6	69.5	50.5
TARE (w/o bias)	0.0261	19 112	60.5	33.1	56.0	54.6	81.3	32.1	77.2	56.4

token rather than switching among large FFN experts, so the method reuses sparse routing for efficient representation editing instead of MoE-style expert substitution. Positioning and intent. We intentionally reuse MoE's load-balancing principle to stabilize token-wise routing and improve utilization, while introducing a new application of these principles to efficient representation editing. This framing positions TARE as a creative specialization of MoE-style routing for PEFT: it preserves the benefits of token-level adaptivity, but delivers them through tiny diagonal hidden representation editors that are computationally frugal and architecturally compatible with frozen backbones and low-overhead fine-tuning.

A.3 Design Principles

Notation and setup. Fix a Transformer layer index ℓ. Let $h_{\ell,t} \in \mathbb{R}^{1\times1\times D_\ell}$ denote the hidden representation of a given token t at layer ℓ. A diagonal hidden representation editor applies a feature-wise affine transformation

$$E_{\theta,\ell,t}(h_{\ell,t}) = h_{\ell,t} \odot \gamma_\ell + \beta_\ell, \quad \theta = (\gamma_\ell, \beta_\ell) \in \mathbb{R}^{1\times1\times D_\ell} \times \mathbb{R}^{1\times1\times D_\ell}. \tag{6}$$

where $\odot$ is the Hadamard product. Let $f_\ell(\cdot)$ denote the remainder network from layer ℓ to the task head, and let $\mathcal{L}(\cdot)$ be the task loss. We consider diagonal edits constrained to a feasible set $\mathcal{B}$ (e.g., $\|(\gamma_\ell - \mathbf{1}, \beta_\ell)\|_2 \le \rho$ or box constraints on γ_ℓ), which makes the optimization and approximation well-defined. For a codebook of n editors $\Theta = \{\theta_i\}_{i=1}^n$, the token-wise selector returns a Top-k index set $\mathcal{T} \subseteq \{1,\dots,n\}$, $|\mathcal{T}| = k$, and nonnegative mixing weights $\{w_i\}_{i\in\mathcal{T}}$ with $\sum_{i\in\mathcal{T}} w_i = 1$. We write $\Theta_k = \mathrm{conv}\{\theta_i : i \in \mathcal{T}\}$ for the corresponding convex hull. Unless stated otherwise, $\|\cdot\|$ denotes the Euclidean norm.

Token-optimal diagonal edit. For a fixed token representation $h_{\ell,t}$, we define the token-optimal

Table 11: Load-balancing ablation of TARE Under the LLaMA-3-8B Model.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE (ours)	0.0392	20 900	**95.8**	**57.3**	**72.9**	**86.1**	**90.9**	**41.4**	**92.1**	**76.7**
TARE (w/o lb loss)	0.0392	20 842	93.8	56.3	72.4	86.6	88.9	39.8	92.7	75.8

Table 12: Position ablation of TARE Under the LLaMA-3-8B Model.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE (q)	0.0392	20 430	86.0	50.4	63.3	74.8	77.5	36.7	86.8	67.9
TARE (k)	0.0098	18 492	78.3	44.7	60.4	73.9	78.2	**43.1**	85.6	66.3
TARE (v)	0.0098	18 486	91.0	56.0	68.1	79.4	86.1	38.4	90.4	72.8
TARE (o)	0.0392	22 438	92.0	57.9	72.2	85.3	88.6	39.4	92.1	75.4
TARE (up)	0.1369	29 556	91.7	**62.2**	69.6	**88.2**	87.3	38.4	**92.3**	75.7
TARE (gate)	0.1369	29 536	87.0	54.7	67.5	82.8	85.1	32.6	91.7	71.6
TARE (down)	0.0392	20 900	**95.8**	57.3	**72.9**	86.1	**90.9**	41.4	92.1	**76.7**

diagonal parameters as

$$\theta^\star(h_{\ell,t}) \in \arg\min_{\theta\in\mathcal{B}} \mathcal{L}\big(f_\ell(E_{\theta,\ell,t}(h_{\ell,t}))\big). \quad (7)$$

This object serves as the ground-truth reference for our approximation analysis; it is the best diagonal edit (within $\mathcal{B}$) for the current token at layer ℓ.

Why token-aware edits are necessary. Consider a first–order Taylor expansion of $\mathcal{L}(f_\ell(E_{\theta,\ell,t}(h_{\ell,t})))$ around the identity edit $(\gamma_\ell, \beta_\ell) = (\mathbf{1}, \mathbf{0})$:

$$\begin{aligned}&\mathcal{L}\big(f_\ell(E_{\theta,\ell,t}(h_{\ell,t}))\big)\\ &\approx \mathcal{L}\big(f_\ell(h_{\ell,t})\big)\\ &\quad + \underbrace{g_\ell(h_{\ell,t})^\top\big((\gamma_\ell - \mathbf{1}) \odot h_{\ell,t} + \beta_\ell\big)}_{\text{first-order term}} \quad (8)\\ &\quad + R_2(\theta; h_{\ell,t}).\end{aligned}$$

where $g_\ell(h_{\ell,t}) = \nabla_{h_{\ell,t}}\mathcal{L}(f_\ell(h_{\ell,t}))$ and R_2 collects second-order terms (bounded under standard smoothness assumptions). Under a norm constraint on $(\gamma_\ell - \mathbf{1}, \beta_\ell)$, the first-order decrease aligns coordinate-wise with $-g_\ell(h_{\ell,t})$, which is token-dependent. Hence a single global edit is generally suboptimal; edits must be token-aware.

Why a small set of prototypes suffices. Empirically (Figure 1), $\theta^\star(h_{\ell,t})$ across tokens clusters into a handful of smooth modes. This invites a codebook view: treat each editor parameter $\theta_i = (\gamma_i, \beta_i)$ as a codeword and the set Θ as a codebook. Classical vector quantization (e.g., Lloyd–Max, k-means) relates hard assignment (Top-1) error to within-cluster radius/variance; learning Θ reduces these radii, improving approximation of $\theta^\star(h_{\ell,t})$ by nearby codewords. We operationalize this with a token-wise selector.

Why Top-k convex mixing is principled. Given the Top-k set $\mathcal{T}$ and weights $\{w_i\}_{i\in\mathcal{T}}$, the mixed parameter is

$$\widehat{\theta} = \sum_{i\in\mathcal{T}} w_i\,\theta_i \in \Theta_k. \quad (9)$$

Let $\mathrm{dist}(\theta^\star, \Theta_k) = \min_{\vartheta\in\Theta_k}\|\theta^\star - \vartheta\|$ be the distance from the token-optimal parameter to the convex set Θ_k. Then, by convexity,

$$\mathrm{dist}(\theta^\star, \Theta_k) \le \min_{i\in\mathcal{T}}\|\theta^\star - \theta_i\|, \quad (10)$$

so allowing convex combinations (Top-k) is never worse than nearest-neighbor/Top-1 in parameter space.

From parameter error to output error. Fix h_ℓ and two parameter vectors θ, θ'. Since $E_{\theta,\ell,t}(h_{\ell,t}) = h_{\ell,t} \odot \gamma_\ell + \beta_\ell$ is affine in θ, one has

$$\begin{aligned}&\|E_{\theta,\ell,t}(h_{\ell,t}) - E_{\theta',\ell,t}(h_{\ell,t})\|_2\\ &= \|\, h_{\ell,t} \odot (\gamma_\ell - \gamma'_\ell) + (\beta_\ell - \beta'_\ell)\,\|_2 \quad (11)\\ &\le L(h_{\ell,t})\,\|\theta - \theta'\|_2.\end{aligned}$$

with $L(h_{\ell,t}) = \sqrt{\|h_{\ell,t}\|_\infty^2 + 1}$ (a token-dependent Lipschitz constant; proof in A.4). Combining (10) and (11) yields an end-to-end token-level bound:

$$\begin{aligned}&\|E_{\widehat{\theta},\ell,t}(h_{\ell,t}) - E_{\theta^\star,\ell,t}(h_{\ell,t})\|_2\\ &\le L(h_{\ell,t})\,\mathrm{dist}(\theta^\star, \Theta_k) \quad (12)\\ &\le L(h_{\ell,t})\,\min_{i\in\mathcal{T}}\|\theta^\star - \theta_i\|_2.\end{aligned}$$

Thus, learning a small set of diagonal hidden representation editors (a codebook) and performing token-wise Top-k convex mixing provides a principled approximation of the unknown token-optimal edit, with guarantees that are never worse than Top-1 and improve as the learned codewords shrink the cluster radii.

Table 13: Hyperparameter Sensitivity Analysis of k (number of selected editors) on TARE.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE ($k=1$)	0.0392	20 196	92.3	47.9	66.0	84.9	88.4	28.0	91.1	71.2
TARE ($k=2$)	0.0392	20 548	93.2	56.9	71.7	83.6	89.1	40.2	90.9	75.1
TARE ($k=3$)	0.0392	20 900	**95.8**	57.3	72.9	86.1	**90.9**	41.4	92.1	**76.7**
TARE ($k=4$)	0.0392	21 274	92.5	56.4	71.5	82.4	87.6	39.4	92.7	74.6
TARE ($k=5$)	0.0392	21 652	93.3	57.2	71.3	85.7	90.6	31.1	92.7	74.6
TARE ($k=6$)	0.0392	22 074	93.5	55.2	**73.6**	85.7	88.1	36.2	**93.5**	75.1
TARE ($k=7$)	0.0392	22 412	95.5	**62.2**	70.7	87.0	89.6	29.6	93.3	75.4
TARE ($k=8$)	0.0392	22 784	91.7	56.9	70.1	**87.4**	88.6	**43.0**	91.9	75.6

Table 14: Hyperparameter Sensitivity Analysis of n (total number of editors) on TARE.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE ($n=6$)	0.0294	20 892	92.8	60.4	73.0	**88.1**	89.1	38.2	**93.7**	76.5
TARE ($n=8$)	0.0392	20 900	**95.8**	57.3	72.9	86.1	**90.9**	**41.4**	92.1	**76.7**
TARE ($n=10$)	0.0489	20 904	93.7	**62.1**	**74.1**	87.7	87.1	35.8	91.9	76.1

Summary. (1) The first-order analysis (8) motivates token-aware diagonal edits. (2) The clustering of $\theta^\star(h_{\ell,t})$ across tokens justifies a finite codebook of hidden representation editors. (3) Top-k convex mixing is a principled realization, with the projection bound (10) and the Lipschitz link (12) connecting parameter-space approximation to output-space error. These results explain why TARE attains fine-grained adaptivity with near-zero inference overhead: hidden representation editors are diagonal (cheap) and selection is sparse (Top-k).

A.4 Lipschitz continuity of the editor's parameters

Fix a layer ℓ and a token t's hidden representation $h_{\ell,t} \in \mathbb{R}^{1\times1\times D_\ell}$. For diagonal hidden representation editors $E_\theta(h_{\ell,t}) = h_{\ell,t} \odot \gamma_\ell + \beta_\ell$ with $\theta = (\gamma_\ell, \beta_\ell) \in \mathbb{R}^{1\times1\times D_\ell} \times \mathbb{R}^{1\times1\times D_\ell}$, we have for any θ, θ':

$$\big\|E_{\theta,\ell,t}(h_{\ell,t}) - E_{\theta',\ell,t}(h_{\ell,t})\big\|_2 \le L(h_{\ell,t})\,\|\theta-\theta'\|_2, \qquad L(h_{\ell,t}) := \sqrt{\|h_{\ell,t}\|_\infty^2 + 1}. \tag{13}$$

Proof. Let $\Delta\gamma_\ell := \gamma_\ell - \gamma'_\ell$ and $\Delta\beta_\ell := \beta_\ell - \beta'_\ell$, and write $\Delta\theta := (\Delta\gamma_\ell, \Delta\beta_\ell)$. By definition,

$$E_{\theta,\ell,t}(h_{\ell,t}) - E_{\theta',\ell,t}(h_{\ell,t}) = h_{\ell,t} \odot \Delta\gamma_\ell + \Delta\beta_\ell. \tag{14}$$

Using the triangle inequality and Hölder/Cauchy–Schwarz,

$$\begin{aligned} \big\|h_{\ell,t} \odot \Delta\gamma_\ell + \Delta\beta_\ell\big\|_2 \;&\le\; \\ \|h_{\ell,t} \odot \Delta\gamma_\ell\|_2 + \|\Delta\beta_\ell\|_2 \;&\le\; \\ \|h_{\ell,t}\|_\infty\,\|\Delta\gamma_\ell\|_2 + \|\Delta\beta_\ell\|_2. & \end{aligned} \tag{15}$$

Define $u := (\|h_{\ell,t}\|_\infty, 1) \in \mathbb{R}^2$ and $v := (\|\Delta\gamma_\ell\|_2, \|\Delta\beta_\ell\|_2) \in \mathbb{R}^2$. Then the previous line is $u^\top v$ and, by Cauchy–Schwarz,

$$\begin{aligned} u^\top v &\le \|u\|_2\,\|v\|_2 \\ &= \sqrt{\|h_{\ell,t}\|_\infty^2 + 1}\,\sqrt{\|\Delta\gamma_\ell\|_2^2 + \|\Delta\beta_\ell\|_2^2} \\ &= L(h_{\ell,t})\,\|\Delta\theta\|_2. \end{aligned} \tag{16}$$

This proves the claim. □

A.5 Load-balancing auxiliary loss

Let $N{=}B\times L$ be the number of tokens in a batch, and let $p_t \in \Delta^{n-1}$ denote the token-wise selection distribution over the n hidden representation editors (e.g., the softmax over the last dimension of h_1^{new}; it may be computed on the Top-k subset or on all n hidden representation editors). We define the average selection distribution across tokens

$$\bar{p} \;=\; \frac{1}{N}\sum_{t=1}^{N} p_t \;\in\; \Delta^{n-1}, \tag{17}$$

and the uniform distribution $U = (\frac{1}{n}, \ldots, \frac{1}{n})$. The load-balancing regularizer encourages aggregate editor usage to be uniform by minimizing the KL divergence

$$\begin{aligned} \mathcal{L}_{\text{LB}} \;&=\; \lambda\,\mathrm{KL}\big(\bar{p}\,\|\,U\big) \\ &=\; \lambda\sum_{i=1}^{n}\bar{p}_i\log\frac{\bar{p}_i}{1/n} \\ &=\; \lambda\Big(\sum_{i=1}^{n}\bar{p}_i\log\bar{p}_i - \log 1/n\Big). \end{aligned} \tag{18}$$

where $\lambda{>}0$ is a weighting coefficient. This term balances overall hidden representation editor utilization without forcing each token's distribution

Table 15: Sample Sensitivity Analysis of TARE Under the LLaMA-3-8B Model.

PEFT	Params.(%)	VRAM(MiB)	MultiArith	GSM8K	SVAMP	MAWPS	AddSub	AQuA	SingleEq	Avg.
TARE ($sample = 500$)	0.0392	20 398	86.3	52.5	64.1	81.1	81.0	36.7	87.0	69.8
TARE ($sample = 1000$)	0.0392	20 406	85.3	51.3	68.6	81.1	85.3	32.8	89.4	70.5
TARE ($sample = 2000$)	0.0392	20 606	89.0	56.6	67.5	84.9	85.1	**46.5**	90.6	74.3
TARE ($sample = 5000$)	0.0392	20 274	91.8	**60.8**	68.1	**87.4**	88.9	41.9	**94.3**	76.2
TARE ($sample = 9919$)	0.0392	20 900	**95.8**	57.3	**72.9**	86.1	**90.9**	41.4	92.1	**76.7**

Table 16: Comparison between LoRA and TARE in terms of parameter ratio, training time, inference time statistics Under the LLaMA-3-8B Model.

PEFT	Source	Params. (%)	Training time (s/epoch)	Mean inference time (s)
LoRA	ICLR 21	0.2345	1015.56	$\mathbf{2.68 \pm 0.43}$
TARE (ours)	This paper	0.0392	**837.35**	3.20 ± 0.67

to be uniform. In practice, for numerical stability we evaluate the log on $\max(\bar{p}_i, \varepsilon)$ with a small ε. The total objective becomes

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{main}} + \mathcal{L}_{\text{LB}}. \quad (19)$$

A.6 Dataset

We extensively evaluate the proposed TARE method on a suite covering nine capability categories: conditional text generation, code synthesis, knowledge completion, symbolic reasoning, closed-book QA, commonsense reasoning, mathematical reasoning, instruction following, and the GLUE benchmark. The datasets and metrics for each task are as follows (see Table 9):

- **Knowledge Reasoning** trains on Commonsense-170K (Hu et al., 2023a) and tests on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-e/c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018), reporting accuracy;
- **Mathematical Reasoning** trains on Math-10K (Hu et al., 2023b) and tests on MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MAWPS (Koncel-Kedziorski et al., 2016), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), and SingleEq (Koncel-Kedziorski et al., 2015), reporting accuracy;
- **GLUE** (Wang et al., 2019) uses the official train/test splits, evaluating MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B with the standard metrics (Matthews correlation, F1, accuracy, Pearson, and Spearman).
- **Conditional Text Generation** uses the E2E-Challenge (Novikova et al., 2017) train/test split and reports BLEU, NIST, METEOR, ROUGE-L, and CIDEr;
- **Code Synthesis** uses HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), training on 90% of the datasets (the remaining 10% as test) and evaluating Pass@1 rate on the official HumanEval/MBPP test sets;
- **Knowledge Completion** uses WikiFact (Goodrich et al., 2019) with accuracy;
- **Closed-Book QA** uses ScienceQA (Saikh et al., 2022) with accuracy;
- **Symbolic Reasoning** uses CoinFlip (Wei et al., 2022) with accuracy;
- **Instruction Following** trains on WizardLM (Xu et al., 2024) and tests on MT-Bench (Zheng et al., 2023), reporting First Turn Score judged by GPT-4 (OpenAI et al., 2023).

A.7 Baseline

The following state-of-the-art baselines are used to compare with our proposed TARE method.

• **LoRA** (Hu et al., 2021): injects trainable low-rank matrices $\mathbf{AB}^\top$ into the updates of linear layers while keeping the original weights frozen; we follow the authors' defaults with rank r=32 and scaling α=32.

• **DoRA** (Liu et al., 2024): decouples the adaptation of direction and radius in weight space, improving optimization stability while maintaining a low update rank.

• **MiLoRA** (Wang et al., 2024a): performs SVD on each weight matrix, keeps the principal singular

Table 17: Comparison of Accuracy (%) on the GLUE Task Under the RoBERTa-base and RoBERTa-large Models with standard deviation. The GLUE task contains eight test datasets: MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B. We reuse the results for LoRA, Adapter-FFN, BitFit, and RED reported in (Wu et al., 2024a). We report averaged performance of five runs with distinct random seeds for our method.

PEFT	Source	RoBERTa	Params.(M)	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
LoRA	ICLR' 21	base	0.29	86.6 ± 0.23	93.9 ± 0.49	88.7 ± 0.76	59.7 ± 4.36	$\mathbf{92.6 \pm 0.10}$	$\mathbf{90.4 \pm 0.08}$	75.3 ± 2.79	90.3 ± 0.54	84.7
Adapter-FFN	EMNLP' 20	base	0.30	$\mathbf{87.1 \pm 0.10}$	93.0 ± 0.00	88.8 ± 1.38	58.5 ± 1.69	92.0 ± 0.28	90.2 ± 0.07	77.7 ± 1.93	90.4 ± 0.31	84.7
BitFit	ACL' 22	base	0.10	84.7 ± 0.08	$\mathbf{94.0 \pm 0.87}$	88.1 ± 1.57	54.0 ± 3.07	91.0 ± 0.05	87.3 ± 0.02	69.8 ± 1.51	89.5 ± 0.35	82.3
RED	ACL' 24	base	0.02	83.9 ± 0.14	93.9 ± 0.31	89.2 ± 0.98	$\mathbf{61.0 \pm 2.96}$	90.7 ± 0.35	87.2 ± 0.17	$\mathbf{78.0 \pm 2.06}$	90.4 ± 0.32	84.3
TARE (ours)	This paper	base	0.22	86.3 ± 0.24	93.1 ± 0.68	$\mathbf{91.5 \pm 2.58}$	58.6 ± 2.64	91.7 ± 0.21	88.6 ± 0.24	77.8 ± 2.53	$\mathbf{90.6 \pm 0.31}$	**84.8**
LoRA	ICLR' 21	large	0.79	90.2 ± 0.25	96.0 ± 0.85	89.8 ± 2.09	65.5 ± 2.02	$\mathbf{94.7 \pm 0.21}$	90.7 ± 0.91	$\mathbf{86.3 \pm 2.41}$	91.7 ± 0.44	88.1
Adapter-FFN	EMNLP' 20	large	0.80	$\mathbf{90.3 \pm 0.15}$	$\mathbf{96.1 \pm 0.75}$	90.5 ± 1.26	64.4 ± 1.56	94.3 ± 0.39	$\mathbf{91.3 \pm 0.24}$	84.8 ± 2.01	90.2 ± 0.24	87.7
RED	ACL' 24	large	0.05	89.5 ± 0.38	96.0 ± 0.48	90.3 ± 1.40	$\mathbf{68.1 \pm 1.69}$	93.5 ± 0.33	88.8 ± 0.11	86.2 ± 1.40	91.3 ± 0.21	87.9
TARE (ours)	This paper	large	0.59	90.0 ± 0.29	94.5 ± 0.75	$\mathbf{92.3 \pm 1.21}$	67.9 ± 1.47	94.6 ± 0.36	89.4 ± 0.47	85.5 ± 1.52	$\mathbf{92.1 \pm 0.46}$	**88.3**

subspace frozen, and attaches LoRA-style low-rank adapters to the minor subspace; during fine-tuning only these adapters are trained.

• **LoReFT** (Wu et al., 2024b): applies low-rank reparameterization jointly across layers and transfers features between tasks via a gating mechanism; we use the public configuration with rank 8.

• **RED** (Wu et al., 2024a): edits hidden representations directly by learning per-feature scaling and bias, without introducing inference-time modules.

• **BitFit** (Ben Zaken et al., 2021): tunes only the bias terms in Transformer layers (e.g., attention and feed-forward blocks) while keeping all other weights frozen; introduces virtually no inference-time overhead.

• **IA^3** (Liu et al., 2022): applies learned per-feature multiplicative gates to key/value and feed-forward activations, modulating channels without changing the backbone weights; requires no rank hyperparameters and adds negligible inference cost.

• **PiSSA** (Meng et al., 2024): computes the singular value decomposition of each weight matrix and uses the top-r singular values and singular vectors to initialize a low-rank adapter $\mathbf{AB}^\top$ while freezing the residual matrix.

• **Spectral Adapter** (Zhang and Pilanci, 2024): first performs SVD $W = USV^\top$ for each pretrained weight matrix and then fine-tunes only the top-r singular directions, either by additive updates to the leading singular vectors (Spectral AdapterA) or by orthogonal rotations parameterized via Cayley transforms (Spectral AdapterR).

• **LoRA-GA** (Wang et al., 2024c): proposes a gradient-aware initialization for LoRA, where the low-rank adapters $\mathbf{BA}$ are initialized from the singular vectors of the full gradient matrix so that the first-step update $\Delta(\eta\mathbf{BA})$ closely aligns with the full fine-tuning gradient $\Delta\mathbf{W}$.

• **LoRA-One** (Zhang et al., 2025): computes the one-step full fine-tuning gradient and performs an SVD-based spectral initialization so that the LoRA adapters $\mathbf{AB}^\top$ are aligned with the top-r singular subspaces of this gradient, already providing a close low-rank approximation to the target update.

We confirm that the experimental setup for the baselines, including backbone models, training processes, and data preprocessing, directly matches the conditions used for TARE. Any differences in training conditions between TARE and the baselines will be clearly explained to ensure a fair comparison. For reproducibility, detailed information about the datasets, model configurations, and hyperparameters are provided in A.6 and A.8.

Regarding the use of load-balancing auxiliary loss in TARE with a value of $\lambda = 0.02$, we clarify that none of the baseline methods use an equivalent loss function. This is because the baselines do not employ a routing mechanism in their architectures. This distinction is critical for evaluating the performance gains attributed to TARE's unique load-balancing approach, and it is addressed in the ablation study to provide clearer insights into the impact of this loss term on model performance.

A.8 Implementation Detail

To cover both major Transformer branches, we fine-tune and evaluate **TARE** on a decoder-only backbone (LLaMA-3-8B (Dubey et al., 2024)) and an encoder backbone (RoBERTa-base/large (Liu et al., 2019)). Unless otherwise noted, we set the number of hidden representation editors to $n = 8$ and select $k = 3$ editors per token via the token-aware selector; the load-balancing auxiliary loss is used with coefficient $\lambda = 0.02$. All experiments are implemented in PyTorch 2.4.1 and run on NVIDIA A100 (80 GB) GPUs. We use AdamW with learning rate 9×10^{-4} ,batch size 32, and 6 epochs, and load base language models in `bfloat16` to reduce memory usage. The datasets and task-specific evaluation metrics are summarized in Table 9.

A.9 Ablation Study

Comparison of the full TARE (scaling plus bias) against variants that remove scaling or bias. Entries report accuracy on seven math reasoning datasets and the average. Params.(%) denotes the percentage of trainable parameters. VRAM(MiB) denotes peak GPU memory. The full model attains the highest average score of 76.7 with 0.0392% trainable parameters. Removing either component degrades performance, and the scaling-only variant (56.4) outperforms the bias-only variant (50.5).

Effect of load-balancing auxiliary loss (n=8, k=3). With the loss, TARE attains a higher average accuracy (76.7 vs. 75.8) while keeping the trainable ratio fixed at 0.0392% and VRAM nearly unchanged. Improvements are seen on MultiArith (+2.0), GSM8K (+1.0), SVAMP (+0.5), AddSub (+2.0), and AQuA (+1.6), with small changes on MAWPS (–0.5) and SingleEq (–0.6). This indicates that load balancing provides consistent gains without additional parameter or memory cost.

A.10 Sensitivity Analysis

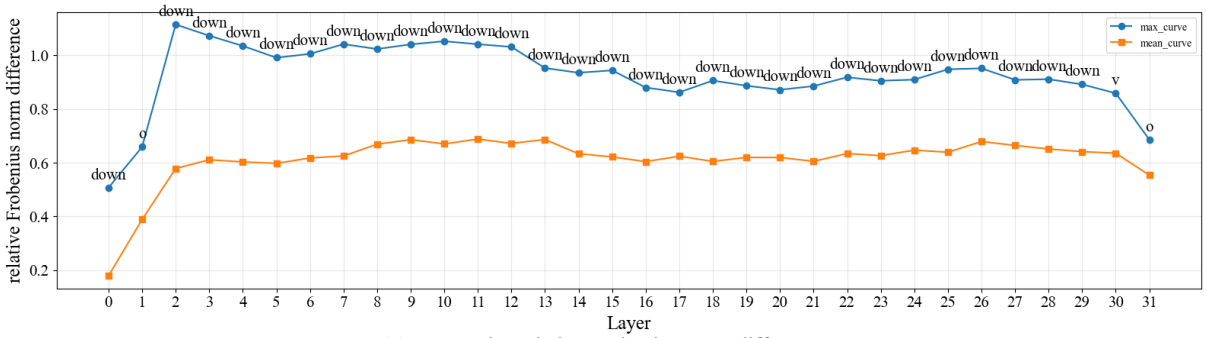
Eight k-variants (from k=1 to k=8) are compared under the same trainable-parameter ratio of 0.0392%. We report accuracy on seven math-reasoning datasets as well as the average, where VRAM(MiB) denotes peak GPU memory. As shown in Table 13, k=3 achieves the best overall average 76.7 with moderate memory usage (20,900 MiB), indicating a favorable balance between capacity and efficiency. Increasing k generally raises VRAM (from 20,196 to 22,784 MiB) and can improve specific tasks, but may not translate to higher overall average: for example, k=7 yields the best GSM8K score 62.2 and strong MultiArith 95.5, while k=8 attains the best AQuA 43.0 and MAWPS 87.4. We further study the sensitivity to the total number of editors n in Table 14. With larger n, the parameter ratio increases (0.0294%→0.0489%) while VRAM remains nearly unchanged (∼20.9 GiB). Overall, n=8 delivers the best average accuracy of 76.7, n=6 is highly competitive 76.5 with fewer trainable parameters, and n=10 brings gains on GSM8K 62.1 and SVAMP 74.1 but slightly lowers the average 76.1. Together, these results suggest that the default setting (k=3, n=8) provides a robust accuracy–efficiency trade-off, while alternative choices can be used to emphasize particular benchmarks.

A.11 Efficiency Analysis

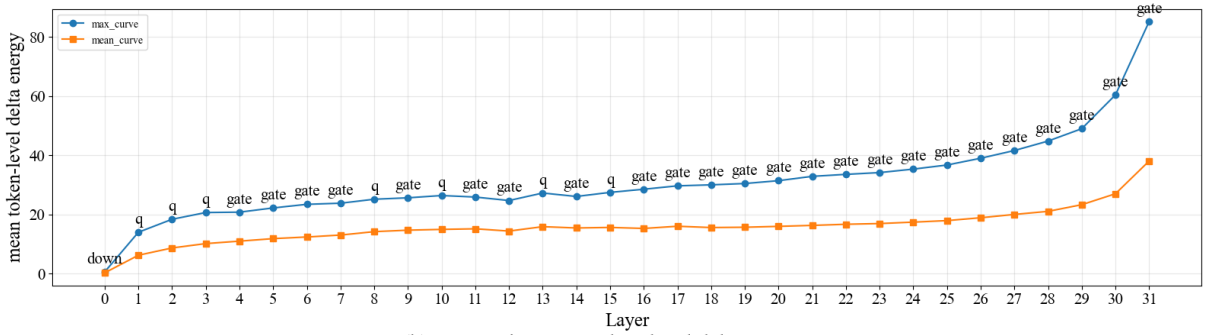
From Table 16, under the same backbone (LLaMA-3-8B) and task setup, TARE improves training efficiency while incurring a small additional inference overhead. In terms of parameter scale, TARE tunes only 0.0392% of parameters, reducing the trainable parameter ratio by roughly $\frac{5}{6}$ ($\approx 83\%$) compared with LoRA (0.2345%), thereby lowering the compute and memory cost of backpropagation and optimization. On Math-10K, this results in a shorter per-epoch training time ($1015.56\,\mathrm{s}$ for LoRA vs. $837.35\,\mathrm{s}$ for TARE, an $\approx 17\%$ reduction), yielding meaningful savings for long-horizon fine-tuning. At inference, after weight merging, LoRA introduces no additional operators and serves as the merged-adapter baseline. On MultiArith, the mean per-example latency is $2.68\,\mathrm{s}$ for LoRA and $3.20\,\mathrm{s}$ for TARE (an increase of $0.52\,\mathrm{s}$/example, $\approx 19\%$). Although TARE keeps its online token-wise editing enabled, the added latency remains sub-second per example. More fine-grained measurements show that the selector and editing operators themselves run at sub-millisecond scale: the mean selection time is $(6.94\times10^{-5}\pm1.46\times10^{-8})\,\mathrm{s}$ and the mean editing time is $(9.78\times10^{-5}\pm1.31\times10^{-10})\,\mathrm{s}$. Overall, TARE provides a smaller parameter footprint, faster training, and a small additional inference overhead, supporting practical use in efficiency-sensitive settings.

A.12 Interpretability Analysis

Based on two representation-difference metrics, we analyze how TARE edits hidden representations across layers. Specifically, $\mathrm{RelFroDiff}$ measures the relative change of a layer activation tensor between the pretrained and fine-tuned models (normalized by the Frobenius norm), while $\mathrm{MeanTokenDeltaEnergy}$ quantifies the average magnitude of token-level modifications, i.e., the mean ℓ_2 norm of per-token differences in the hidden dimension. Taken together, the layer-wise trends of $\mathrm{RelFroDiff}$ and $\mathrm{MeanTokenDeltaEnergy}$ indicate that TARE does not induce a large, uniform drift of layer representations. Instead, it maintains a comparatively stable global change ratio while producing stronger token-dependent perturbations, which is consistent with its token-aware design. These effects are more pronounced in mid-to-late layers and are especially evident along the FFN/MLP pathway where TARE is inserted, suggesting that TARE mainly performs



(a) Layer-wise relative Frobenius norm difference



(b) Layer-wise mean token-level delta energy

Figure 5: Layer-wise representation differences between the original and TARE-tuned models. Top: relative Frobenius norm difference (max_curve and mean_curve) across layers. Bottom: mean token-level delta energy across layers.

fine-grained, token-specific modulation rather than coarse layer-wise shifts.

A.12.1 Definition of relative Frobenius norm difference

Let $a, b \in \mathbb{R}^{B\times L\times H}$, where B, L, and H correspond to batch_size, sequence_length, and hidden_dim, respectively. Flattening the first two dimensions yields

$$\begin{aligned}\tilde{a} &= \mathrm{reshape}(a) \in \mathbb{R}^{(BL)\times H},\\ \tilde{b} &= \mathrm{reshape}(b) \in \mathbb{R}^{(BL)\times H}.\end{aligned} \tag{20}$$

The relative Frobenius norm difference is defined as

$$\mathrm{RelFroDiff}(a,b) = \frac{\|\tilde{b}-\tilde{a}\|_F}{\|\tilde{a}\|_F + \varepsilon}, \tag{21}$$

where $\varepsilon > 0$ is a numerical stabilizer (e.g., 10^{-8}), and $\|\cdot\|_F$ denotes the Frobenius norm. Equivalently, it can be written as the element-wise summation form:

$$\mathrm{RelFroDiff}(a,b) = \frac{\sqrt{\sum_{i=1}^{B}\sum_{t=1}^{L}\sum_{j=1}^{H}(b_{i,t,j}-a_{i,t,j})^2}}{\sqrt{\sum_{i=1}^{B}\sum_{t=1}^{L}\sum_{j=1}^{H}a_{i,t,j}^2}+\varepsilon}. \tag{22}$$

This metric measures the overall magnitude of the change from a to b (characterized by the Frobenius norm) relative to the overall scale of the reference tensor a.

A.12.2 Definition of mean token-level delta energy

Let $a, b \in \mathbb{R}^{B\times L\times H}$, where B, L, and H correspond to batch_size, sequence_length, and hidden_dim, respectively. For each sample $i \in \{1,\dots,B\}$ and token position $t \in \{1,\dots,L\}$, define the token-level delta energy as the ℓ_2 norm along the hidden dimension:

$$\begin{aligned}E_{i,t} &= \|b_{i,t,:} - a_{i,t,:}\|_2\\ &= \sqrt{\sum_{j=1}^{H}(b_{i,t,j}-a_{i,t,j})^2}.\end{aligned} \tag{23}$$

The mean token-level delta energy is then defined as the average over all batches and token positions:

$$\begin{aligned}&\mathrm{MeanTokenDeltaEnergy}(a,b)\\ &= \frac{1}{BL}\sum_{i=1}^{B}\sum_{t=1}^{L}E_{i,t}\\ &= \frac{1}{BL}\sum_{i=1}^{B}\sum_{t=1}^{L}\|b_{i,t,:}-a_{i,t,:}\|_2.\end{aligned} \tag{24}$$

This metric quantifies the per-token magnitude of representation changes from a to b (i.e., the hidden-dimension ℓ_2 distance) and averages it across the entire batch and sequence positions, characterizing the token-level editing intensity in expectation.

A.12.3 Results

From the layer-wise $\mathrm{RelFroDiff}$ results (Figure 5), max_curve is dominated by down (the MLP down_proj output) across most layers, forming a stable plateau around 0.9–1.1. In contrast, mean_curve is smoother and remains around ≈ 0.6. This suggests that, compared with attention projections (q/k/v/o), the overall representation drift after fine-tuning is more concentrated in the FFN output projection (down), which directly writes the FFN-transformed content back into the residual stream. Such a pattern is consistent with TARE's design: the observed changes are less concentrated on attention projections and are more concentrated on the FFN write-back, thereby preserving the global backbone structure.

In contrast, $\mathrm{MeanTokenDeltaEnergy}$ reveals stronger token-level perturbation magnitudes (i.e., the mean ℓ_2 norm of per-token differences). mean_curve increases with depth, reaching ≈ 38 at the final layer, and max_curve rises sharply in the last layers to ≈ 85. Notably, the dominant contributor to max_curve is gate (the MLP gate_proj) for most layers, indicating that the largest per-token ℓ_2 changes are concentrated in the gating branch. Given LLaMA's gated-MLP design, this suggests that TARE tends to adjust token-specific gating/activation patterns. We hypothesize that such token-wise gating modulation may be beneficial for arithmetic and symbol-composition tasks, where selectively amplifying or suppressing key tokens can support multi-step reasoning.

The dominant modules differ across the two metrics (down for $\mathrm{RelFroDiff}$ vs. gate for $\mathrm{MeanTokenDeltaEnergy}$) because they emphasize different notions of change. $\mathrm{RelFroDiff}$ measures relative whole-tensor drift (normalized by the Frobenius norm of a reference activation, e.g., $\|a\|_F$), and thus tends to highlight changes tied to residual write-back. In contrast, $\mathrm{MeanTokenDeltaEnergy}$ captures absolute token-wise magnitudes, where gating can induce sharper changes on a subset of tokens. Finally, the persistent gap between max_curve and mean_curve, which becomes larger in middle-to-late layers, suggests that TARE performs sparse, targeted modulation on key layers/modules/tokens rather than uniform small perturbations, enabling efficient adaptation with very few trainable parameters.

A.13 Results of Standard Deviations

In the main text, we presented the average GLUE results under the RoBERTa-base and RoBERTa-large settings. To provide a quantifiable measure of uncertainty in the reported performance and to help improve the reproducibility of our experiments, we also report the standard deviations on the eight GLUE datasets in Table 17. Specifically, the results of our method are averaged over five runs with different random seeds, while the corresponding baseline statistics are reused from prior work when available.