

StructTune: Structured Sparse Fine-Tuning for Efficient Adaptation of Large Language Models

Yuyang Liu[◇] Lai-Man Po^{◇*} Kwok-Wai Cheung[†] Zeyu Jiang[◇]
Haoxuan Wu[◇] Kun Li[◇] Xuyuan Xu[‡]

[◇]City University of Hong Kong, Hong Kong SAR, China
{yuyangliu5, eelmpo, zeyujiang4}-c@my.cityu.edu.hk
{haoxuanwu2, kunli25}-c@my.cityu.edu.hk

[†]The Hang Seng University of Hong Kong, Hong Kong SAR, China
keithccheung@hsu.edu.hk

[‡]MagicLight Technology Co., Ltd, China
evan@magiclight.ai

Abstract

Parameter-efficient fine-tuning (PEFT) methods like LoRA have become crucial for adapting large language models under computational constraints. While LoRA focuses on dense weight updates, recent sparse methods propose updating only selected portions of pretrained weights. However, these sparse approaches predominantly use element-wise parameter selection, overlooking inherent structured patterns in LLM weight distributions. We introduce StructTune¹, a PEFT method that uses entire rows or columns as basic units for identifying task-relevant sparse components. Our analysis reveals LLaMA models exhibit distinct structured patterns in pretrained weights, with fine-tuning updates showing structured amplification rather than uniform changes. We develop a sensitivity-based framework to efficiently identify task-relevant weight structures without full fine-tuning. Experiments on mathematical reasoning and code generation demonstrate StructTune’s structured sparse updates outperform both dense LoRA and element-wise sparse methods while maintaining computational efficiency. Ablation studies validate our sensitivity-based selection strategy, confirming the efficacy of exploiting structured weight patterns in PEFT.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Touvron et al., 2023; Grattafiori et al., 2024) have revolutionized natural language processing with exceptional generalization across diverse tasks (Taori et al., 2023). However, optimal domain-specific performance typically requires fine-tuning on specialized datasets (Zhang et al., 2025a). While full fine-tuning (FFT) updates all model parameters effectively, its computational and memory requirements are prohibitive for most users given LLMs’ immense scale.

To address these challenges, parameter-efficient fine-tuning (PEFT) methods have emerged as compelling alternatives (Han et al., 2024; Ding et al., 2023; Houlsby et al., 2019; Agarwal et al., 2025), updating only small parameter subsets while achieving comparable performance to FFT with substantially lower computational requirements. Among PEFT methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) decomposes weight updates into low-rank matrices but applies dense updates to all parameters in W_0 , often failing to fully replicate FFT performance due to redundant information (Zhang et al., 2025b; Ao et al., 2025).

Recent sparse weight update techniques address these limitations by updating only a small subset of pretrained weights (Nikdan et al., 2024; Wang et al., 2024), yielding much fewer parameters in ΔW than in W_0 . Yet identifying task-relevant parameters remains difficult. Prior work typically selects parameters using gradient-based scores such

* The corresponding author.

¹code will be released at https://github.com/LiuYYGitH/StructTune_TACL

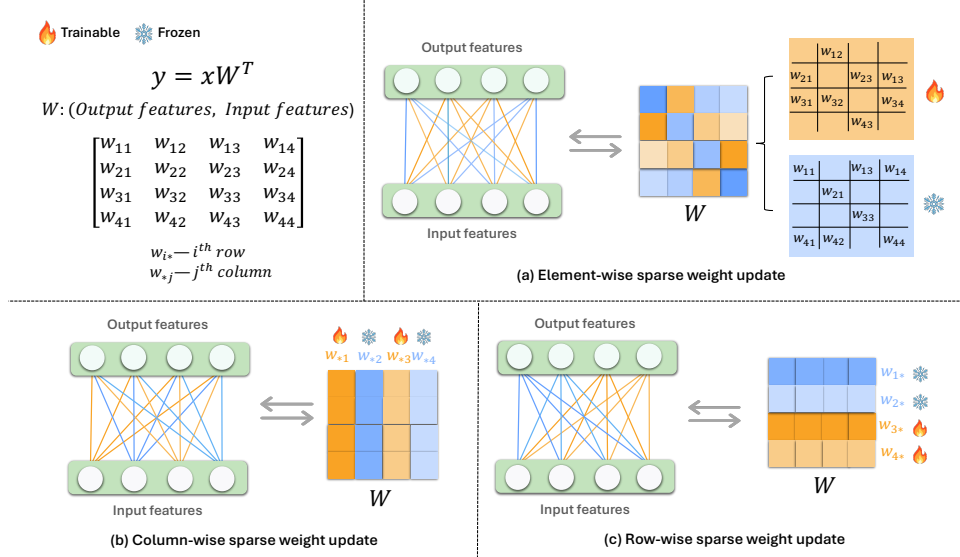


Figure 1: Sparsity determination units for parameter selection: (a) element-wise sparsity treats each matrix element w_{ij} as an independent unit; (b) column-wise sparsity treats entire columns w_j as units; and (c) row-wise sparsity treats entire rows w_i as units.

as $|\delta w_{ij}|$ (Zhang et al., 2023; Wang et al., 2024), leaving the structural design of sparse weight updates largely unexplored.

Current approaches typically select individual weight elements (w_{ij}) as basic units (Nikdan et al., 2024; Wang et al., 2024), but recent research increasingly focuses on entire rows (w_i) or columns (w_j) as cohesive units capturing structured knowledge representations. Studies on FFN neurons (Yu and Ananiadou, 2024) and pruning techniques (Ashkboos et al., 2024; Liu et al., 2025) demonstrate that LLMs organize knowledge in dimension-aligned patterns. Based on this insight, we performed FFT analysis of LLaMA3.2-1B on Math-10k (Hu et al., 2023). Figure 2 shows pre-trained weights display clear row/column stripe patterns, where certain rows or columns experience substantially larger fine-tuning updates, indicating their task-specific importance. These findings suggest structured updates targeting relevant regions could improve performance. Yet current methods use random selection (Po et al., 2024), failing to locate task-relevant areas and resulting in poor parameter choices that overlook critical components while adding unnecessary updates.

In light of the above mentioned observations, we present StructTune, a novel approach to systematically discover task-relevant sparse weight components by treating complete rows or columns as basic units. Our key insight is that *structured weight regions exhibit varying degrees of task rel-*

evance, enabling principled selection over random sampling. Additionally, our row-wise and column-wise updates align naturally with standard matrix operations (see §3.2), providing computational advantages over element-wise sparse methods (Gale et al., 2020). This framework creates a new paradigm for structured sparse weight updates that combines efficiency with accuracy.

Our contributions are summarized as follows:

1. We identify structured row- and column-wise weight update patterns within LLMs. Our analysis reveals that fine-tuning updates are not uniformly distributed but are instead concentrated within specific rows or columns, exhibiting a non-uniform, stripe-like patterns. We define this phenomenon as **structured patterns**, which signify dimension-aligned knowledge representations in model weights and motivate the thought of treating an entire row or column as an atomic functional unit (§A.3).
2. This work proposes a sensitivity-based analysis framework that treats rows and columns as fundamental units for identifying task-relevant weight components in LLMs. It extends prior work on inference analysis and weight pruning to fine-tuning while reducing the computational burden of FFT through structured analysis.

3. StructTune is presented as a novel PEFT approach employing structured sparse weight updates with two variants: StructTune-FA (column-wise selection using importance-based criteria) and StructTune-FAB (combined row-and-column selection based on findings that both dimensions can simultaneously show high task relevance).

We assessed StructTune on mathematical reasoning (Cobbe et al., 2021) and code generation (Wei et al., 2023) across various LLM architectures (Grattafiori et al., 2024; Touvron et al., 2023; Yang et al., 2025), showing superior results versus existing PEFT approaches (Hu et al., 2022; Nikdan et al., 2024; Po et al., 2024). Our evaluation confirms the sensitivity-based scoring’s effectiveness and demonstrates clear benefits over random selection. StructTune’s structured sparsity delivers substantial computational gains compared to element-wise methods, establishing an effective framework that improves both task performance and efficiency.

2 Related work

2.1 PEFT with Sparse Weight Update

FFT’s computational costs have rendered it impractical for many applications, prompting PEFT methods that update parameter subsets while preserving performance. LoRA (Hu et al., 2022) exploits low-rank structure by decomposing $\Delta W \in \mathbb{R}^{d \times k}$ into matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ where $r \ll d, k$:

$$y = x(W_0)^T + x(\Delta W)^T = x(W_0)^T + x(BA)^T, \quad (1)$$

where $W_0 \in \mathbb{R}^{d \times k}$ denotes pretrained weights and the weight update ΔW is approximated by BA .

Beyond LoRA’s uniform updates, recent methods pursue strategic parameter allocation. An early foundational approach, SparseAdapter (He et al., 2022), re-examines adapter efficiency through network pruning at initialization, demonstrating that adapters can maintain performance even at 80% sparsity. While it primarily relies on element-wise pruning, which may overlook the inherent structured knowledge representations in LLMs. Moreover, coarse-grained approaches like AdaLoRA (Zhang et al., 2023) assign layer-specific ranks based on importance, while fine-grained methods target specific components through sparse updating. RoseLoRA (Wang

et al., 2024) applies sparsity to matrices B and A via progressive pruning, retaining one trainable element per column in B and per row in A , though this introduces sparsity indirectly through decomposition constraints rather than inheriting from weight properties. RoSA (Nikdan et al., 2024) combines LoRA with sparse components using two-stage training: gradient collection followed by magnitude-based sparse selection. SBoRA (Po et al., 2024) uses standard basis vectors for low-rank matrix construction, enabling sparse updates with r columns or rows. While SBoRA shows promise for row/column-wise sparse updates, its random selection fails to capture task-critical components, limiting effectiveness.

2.2 Weight Importance Analysis

Effective sparse weight updating requires quantitative metrics to identify task-relevant parameters and assess layer importance. In this section, we will introduce three commonly applied metrics. For clarity, we denote all linear weights as $W^{all} = \{W^1, W^2, \dots, W^M\}$ and loss function as $\mathcal{L}(W^{all})$, where M is the number of linear layers. For each linear weights $W^m \in \mathbb{R}^{d_m \times k_m}$:

$$W^m = \begin{bmatrix} w_{11}^m & w_{12}^m & \cdots & w_{1k_m}^m \\ w_{21}^m & w_{22}^m & \cdots & w_{2k_m}^m \\ \vdots & \vdots & \ddots & \vdots \\ w_{d_m 1}^m & w_{d_m 2}^m & \cdots & w_{d_m k_m}^m \end{bmatrix} \quad (2)$$

Parameter magnitude. Parameter magnitude analysis uses $|w_{ij}^m|$ as an importance metric for model pruning (Renda et al., 2020; Zafrir et al., 2021), treating lower-magnitude weights as less significant. However, this metric is suboptimal for fine-tuning as small-magnitude parameters may be crucial for task-specific outputs, while large magnitude doesn’t ensure greater impact.

Parameter gradient magnitude. Gradient-based importance assessment measures how much each weight w_{ij}^m should be adjusted by computing gradients of loss function $\mathcal{L}$ with respect to w_{ij}^m (Nikdan et al., 2024). While more suitable for fine-tuning than parameter magnitude, this metric ignores parameter values themselves and exhibits high sensitivity to data, resulting in significant variance. This limitation motivates the need for a comprehensive importance metric that effectively unifies both parameter values and their corresponding gradients.

Parameter sensitivity. Sensitivity score $I(w_{ij}^m)$ is a widely adopted weight importance

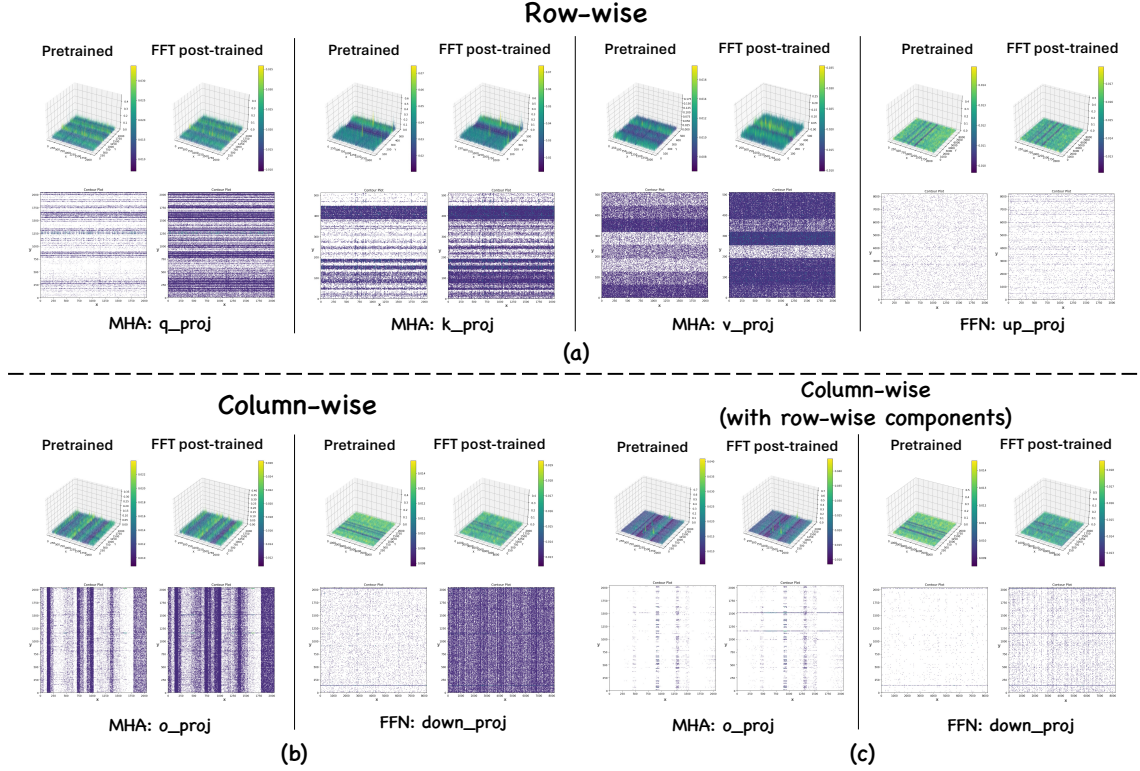


Figure 2: Weight magnitude distributions for LLaMA3.2-1B before and after fine-tuning on Math-10k. Subfigure (a) illustrates layers showing row-wise structured amplification, while subfigures (b) and (c) illustrate layers showing column-wise structured amplification. Notably, subfigure (c) also exhibits a visible row-wise structure coexisting with the dominant column-wise pattern. Each subfigure arranges pre-finetuning weights on the left and post-finetuning weights on the right. The top row shows 3D surface visualizations and the bottom row shows corresponding 2D heatmaps.

metric (Zhang et al., 2023; Wang et al., 2024), defined as:

$$I(w_{ij}^m) = |\delta w_{ij}^m \cdot w_{ij}^m|, \quad (3)$$

where w_{ij}^m denotes the element at the i -th row and j -th column of linear layer m , and δw_{ij}^m represents the gradient of loss function $\mathcal{L}(\cdot)$ at weight w_{ij}^m .

Using first-order Taylor expansion, $I(w_{ij}^m)$ quantifies the loss magnitude change when setting a weight to zero (Molchanov et al., 2019):

$$\mathcal{L}(w_{ij}^m) \approx \mathcal{L}(w_{ij}^{m(\tau)}) + \frac{d\mathcal{L}(w_{ij}^{m(\tau)})}{dw_{ij}^m} (w_{ij}^m - w_{ij}^{m(\tau)}), \quad (4)$$

setting w_{ij}^m to zero give:

$$\begin{aligned} |\mathcal{L}(0) - \mathcal{L}(w_{ij}^{m(\tau)})| &\approx \left| \frac{d\mathcal{L}(w_{ij}^{m(\tau)})}{dw_{ij}^m} (0 - w_{ij}^{m(\tau)}) \right| \\ &= |\delta w_{ij}^{m(\tau)} w_{ij}^{m(\tau)}|. \end{aligned} \quad (5)$$

This shows that the sensitivity score can also be expressed as

$$I(w_{ij}^m) = |\mathcal{L}(0) - \mathcal{L}(w_{ij}^m)|, \quad (6)$$

with the step τ subscript omitted for simplicity.

Higher sensitivity scores indicate greater model dependence on specific parameters, suggesting higher retention probability. This combines parameter magnitude and gradient information effectively. However, computing sensitivity across all weights requires FFT, undermining PEFT’s efficiency. We address this by leveraging LoRA’s approximation $\Delta W \approx BA$ to efficiently identify task-relevant weights without FFT.

While these metrics provide a point-wise estimation of parameter importance, they inherently treat parameters as independent units, potentially overlooking the high-level structural dependencies within Transformer architectures. For instance, in Feed-Forward Networks (FFNs), specific rows of one layer and columns of the next collectively define an intermediate memory cell (Geva et al., 2021). To formally represent this interaction, we express the transformation of an FFN module as a

summation of d_{ff} functional units:

$$y = \sum_{j=1}^{d_{ff}} \sigma(x \cdot (w_{j,*}^{up})^T) (w_{*,j}^{down})^T \quad (7)$$

where $w_{j,*}^{up}$ (the j -th row of the up-projection) and $w_{*,j}^{down}$ (the j -th column of the down-projection) are functionally coupled through the same intermediate dimension j . Similarly, in Multi-Head Attention (MHA), the query, key, and value matrices (W^Q, W^K, W^V) primarily interact with input features through their rows, whereas the output projection W^O interacts via its columns to reconstruct the hidden state. This matrix-role asymmetry is also shown in Figure 2: row-wise masks are better aligned with matrices such as $W^Q/W^K/W^V/W^{up}$ (e.g., subfigure (a)), whereas column-wise masks are better aligned with matrices such as W^O/W^{down} (e.g., subfigures (b) and (c)). Notably, patterns can coexist within the same layer; as illustrated in subfigure (c), a row-wise structure remains visible alongside a dominant column-wise pattern. These observations on layer-level functional heterogeneity and dual-dimensional coexistence motivate StructTune-FAB, which combines row-wise and column-wise sparsity so each layer can adopt its most suitable sparsity pattern.

2.3 Row-Wise and Column-Wise Analysis of LLM

Recent research exploring LLM internal mechanisms has shown increased attention to row-wise and column-wise analysis of learned parameters. Studies emphasize the importance of treating feed-forward network (FFN) neurons as fundamental units (Yu and Ananiadou, 2023; Gurnee et al., 2023; Nanda et al., 2023), with Comparative Neuron Analysis revealing distinct contributions of FFN neurons, corresponding to specific rows and columns in weight matrices, to mathematical reasoning tasks (Yu and Ananiadou, 2024). Neuron-level activation further provides a unified lens for how chain-of-thought elicits arithmetic reasoning in LLMs, with such behaviors aligning with localized activation patterns rather than uniform activity across neurons and dimensions (Rai and Yao, 2024). Complementary analyses ask which internal components are responsible for reasoning and likewise find it concentrated in specific neuron subsets and dimensions rather than uniformly distributed (Shao and Wu, 2025). Model

pruning studies have demonstrated the efficacy of row-wise and column-wise operations (Ashkboos et al., 2024; Liu et al., 2025), suggesting that LLMs organize knowledge in dimension-aligned patterns. These findings motivate exploring row-wise and column-wise weight updates in fine-tuning, as selective updates to critical rows or columns may provide more precise and efficient modification than updating entire weight matrices.

3 Methodology

This section presents the overall pipeline of our proposed method, StructTune, designed to efficiently fine-tune LLMs under structured sparsity constraints. Our approach eliminates the need for full model fine-tuning by introducing a refined, sensitivity-based quantitative analysis of individual weight elements. This analysis enables us to identify the most influential weight rows and columns—those that contribute most significantly to model performance—which are then selectively updated during the fine-tuning process.

3.1 Weight Importance Quantitative Analysis

To enable effective sparsity component selection, we adopt a two-stage strategy inspired by RoSA (Nikdan et al., 2024). In the first stage, we utilize a subset $\mathcal{D}_s$ of the training dataset $\mathcal{D}$ to perform quantitative weight importance analysis. In the second stage, the identified sparse components are employed for structured weight updates during the full training process.

We denote $\mathcal{L}$ as the loss function for the training task and W^{all} as the collection of all linear weight elements, with $\tilde{b}$ representing auxiliary parameters such as biases and normalization terms. The optimization objective is formulated as:

$$\min_{W^{all}, \tilde{b}} \mathcal{L}(\mathcal{D}; W^{all}, \tilde{b}). \quad (8)$$

For a linear layer W^m , the computation is expressed as:

$$y^m = x^m (W^m)^T + \tilde{b}^m, \quad (9)$$

where $x^m \in \mathbb{R}^{1 \times k_m}$ and $y^m \in \mathbb{R}^{1 \times d_m}$ are the input and output of the layer, respectively. Following RoSA, gradients are computed using the chain rule as:

$$\frac{d\mathcal{L}}{dW^m} = \frac{d\mathcal{L}}{dy^m} \frac{dy^m}{dW^m} = \frac{d\mathcal{L}}{dy^m} x^m. \quad (10)$$

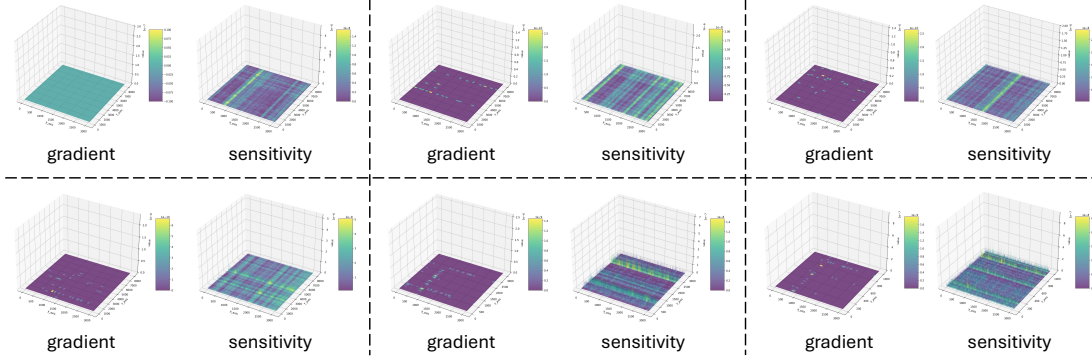


Figure 3: Comparison of importance analysis methods: gradient (RoSA(Nikdan et al., 2024)) vs. sensitivity (ours). Sensitivity values show better row/column distribution patterns than gradients, which exhibit numerous zeros creating a flat surface, while sensitivity reveals more informative patterns.

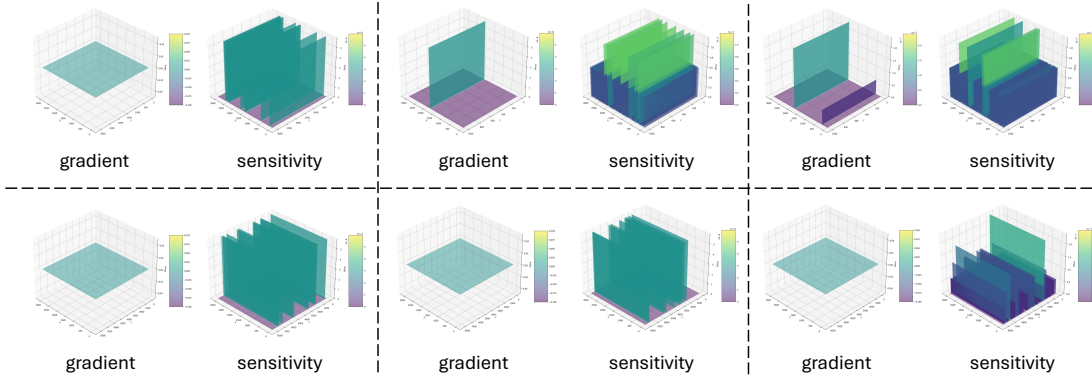


Figure 4: Comparison of importance analysis methods' normalized row/column value distributions: gradient vs. sensitivity (ours). Sensitivity-derived scores provide more distinct patterns than gradients, enabling more effective identification and selection of optimal rows/columns for updating.

However, this gradient-based approach is limited to single-step analysis and exhibits high variability with limited differentiation between gradient magnitudes. Multi-step gradient accumulation, which would require FFT, is computationally prohibitive and undermines PEFT's efficiency. To address this, we propose a novel method leveraging LoRA's low-rank approximation, where $\Delta W^m \approx B^m A^m$, to perform multi-step weight importance analysis without FFT.

We distinguish between the instantaneous gradient update $\delta W^m \in \mathbb{R}^{d_m \times k_m}$ and the cumulative adapted weight $\Delta W^m \in \mathbb{R}^{d_m \times k_m}$. At training step τ , the adapted weight is the sum of all prior gradient updates:

$$\Delta W_\tau^m = \sum_{t=1}^{\tau-1} \delta W_t^m. \quad (11)$$

Within the LoRA framework, the low-rank decomposition for layer m involves matrices $B^m \in \mathbb{R}^{d_m \times r}$ and $A^m \in \mathbb{R}^{r \times k_m}$, with their gradient up-

dates at step τ denoted as δB_τ^m and δA_τ^m . The updated weight matrix at step τ is:

$$W_\tau^{m'} = W_0^m + \Delta W_\tau^m \approx W_0^m + B_\tau^m A_\tau^m, \quad (12)$$

where W_0^m represents the pretrained weights. The forward transformation incorporates both the adapted state and instantaneous updates:

$$y^m = x^m (W_\tau^{m'} + \delta W_\tau^m)^T + \tilde{b}^m = x^m [W_0^m + (B_\tau^m + \delta B_\tau^m)(A_\tau^m + \delta A_\tau^m)]^T + \tilde{b}^m. \quad (13)$$

The instantaneous gradient update at step τ is derived as:

$$\delta W_\tau^m = [W_0^m + (B_\tau^m + \delta B_\tau^m)(A_\tau^m + \delta A_\tau^m)] - W_\tau^{m'} = B_\tau^m (\delta A_\tau^m) + (\delta B_\tau^m) A_\tau^m + (\delta B_\tau^m) (\delta A_\tau^m). \quad (14)$$

We provide a more detailed walk-through of the forward pass and the algebraic expansion leading to δW_τ^m in §A.1. This decomposition yields three

terms: the gradient of A^m scaled by B_τ^m , the gradient of B^m scaled by A_τ^m , and their second-order interaction. Parameter sensitivity at step τ is then defined as:

$$I_\tau(w_{ij}^m) = |\delta W_\tau^m \cdot W_\tau^{m'}|_{ij}. \quad (15)$$

The physical meaning of this sensitivity score is further explained in §A.2. To mitigate noise in instantaneous sensitivity measurements, we apply an exponential moving average (Zhang et al., 2022) to balance historical and current observations:

$$\bar{I}_\tau(w_{ij}^m) = \beta \times I_{\tau-1}(w_{ij}^m) + (1 - \beta) \times I_\tau(w_{ij}^m), \quad (16)$$

where $\beta = 0.8$ provides an optimal balance between stability and responsiveness. After the first stage, row-wise and column-wise sparsity selection is performed using normalized sensitivity scores, calculated as:

$$\bar{I}(w_{i*}^m) = \frac{1}{k_m} \sum_{j=1}^{k_m} \bar{I}(w_{ij}^m), \quad \bar{I}(w_{*j}^m) = \frac{1}{d_m} \sum_{i=1}^{d_m} \bar{I}(w_{ij}^m). \quad (17)$$

As shown in Figure 3, our sensitivity-based scoring reveals pronounced row-wise and column-wise distribution patterns compared to gradient-based methods. Comparative analysis with RoSA’s gradient-based approach, visualized in Figure 4, demonstrates that our normalized scores exhibit clearer differentiation, enabling precise selection of task-relevant rows and columns.

To further illustrate its effectiveness in selecting structured sparsity components, we computed average gradient magnitudes using Eq. (17), and compared RoSA’s gradient-based scheme with our sensitivity-based method. As shown in Figure 4, the normalized scores from our method exhibit much clearer and more distinguishable patterns across both dimensions. This improved contrast enables more accurate and reliable selection of important rows and columns for parameter updates, enhancing the effectiveness of structured sparsity identification process.

3.2 StructTune Pipeline

StructTune employs a two-stage pipeline, as illustrated in Figure 5. In the first stage, a subset of the training data is used to perform standard LoRA fine-tuning, enabling comprehensive weight importance analysis via sensitivity scores.

Algorithm 1 StructTune’s first-stage sparsity selection

Require: $T \leftarrow$ Training steps, $\mathcal{D}_s \leftarrow$ Sampled dataset, $(d_m, k_m) \leftarrow$ Layer shape
Require: $\beta \leftarrow$ Moving average coeff., $p\% \leftarrow$ Preset trainable parameter percentage threshold.
Require: $R^m, C^m \leftarrow$ Selected rows/columns lists, initialized as []
1: **for** τ in T **do**
2: $W_\tau^{m'} \leftarrow W_0^m + B_\tau^m A_\tau^m$
3: $\delta B_\tau^m, \delta A_\tau^m \leftarrow \Delta \mathcal{L}(\mathcal{D}_s; W_\tau^{m'})$
4: $I_\tau(w_{ij}^m) \leftarrow |\delta W_\tau^m \cdot W_\tau^{m'}|_{ij}$
5: $\bar{I}_\tau(w_{ij}^m) \leftarrow \beta \times \bar{I}_{\tau-1}(w_{ij}^m) + (1 - \beta) \times I_\tau(w_{ij}^m)$
6: **end for**
7: Compute normalized row/column scores: $\bar{I}(w_{i*}^m), \bar{I}(w_{*j}^m)$
8: **StructTune-FA:** Greedily select columns with highest $\bar{I}(w_{*j}^m)$ until $|C^m| = p\% \times |W^m|$
9: **StructTune-FAB:** Greedily select rows/columns with highest scores until $|C^m + R^m| = p\% \times |W^m|$

This stage generates binary masks to guide sparsity pattern selection. Two variants are implemented: StructTune-FA uses a greedy column-wise selection protocol, sequentially incorporating columns based on descending normalized importance scores until a predefined $p\%$ parameter budget is met. StructTune-FAB dynamically selects both rows and columns based on their importance rankings, adhering to the same budget. The selection process is formalized in Algorithm 1. StructTune-FAB is a generalized framework that adaptively allocates the parameter budget across both rows and columns in a unified pool, capturing diverse importance patterns across layers. StructTune-FA is a constrained, column-only version which enables direct comparison with existing column-wise approaches (Po et al., 2024). Additionally, we tested StructTune-FB and present those results in the appendix §B.2.

In the second stage, training focuses on the identified row-wise and column-wise sparse components. Other than RoSA’s SDDMM-based approach (Nikdan et al., 2024), StructTune could adopt SBoRA’s architectural innovations (Po et al., 2024), using standard basis vectors to construct projection matrices. In SBoRA-FA, the projection-down matrix A_{sb} (replacing A) comprises r row-oriented standard basis vectors from the identity matrix $W_i \in \mathbb{R}^{k \times k}$:

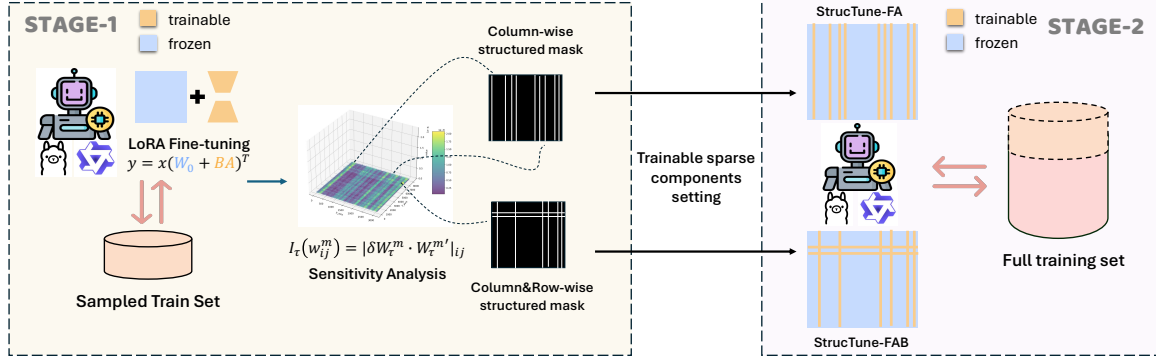


Figure 5: StructTune’s two-stage structured sparse fine-tuning methodology. Stage 1: Weight importance evaluation on sampled data using standard LoRA fine-tuning to generate sensitivity-based importance scores and binary masks. Stage 2: Structured sparse fine-tuning on the full dataset using high-importance parameters while maintaining sparsity constraints.

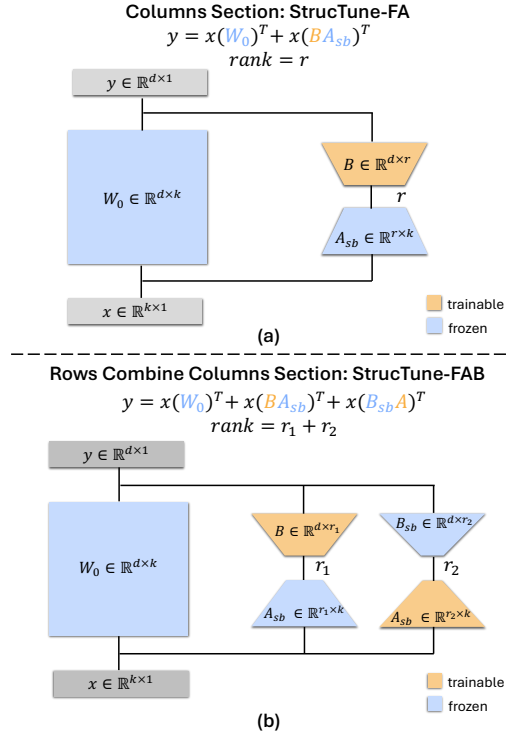


Figure 6: Architectural design of (a) StrucTune-FA and (b) StrucTune-FAB using standard-basis low-rank matrices for superior efficiency over SDDMM. StrucTune-FAB combines BA_{sb} and $B_{sb}A$ with summed ranks.

$$W_i = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad (18)$$

and A_{sb} consists of one-hot vectors e_p with a unit entry at index p :

$$e_p = [0 \ \cdots \ 1 \ 0 \ \cdots] \quad (19)$$

$\uparrow$
 p^{th} element

The weight adaptation $\Delta W = BA_{sb}$ results in a column-wise sparse matrix, with non-zero entries in columns corresponding to indices in A_{sb} . For example, if:

$$B = [b_1, b_2, b_3], \quad \text{where } b_1, b_2, b_3 \in \mathbb{R}^{d \times 1}, \quad (20)$$

$$A_{sb} = \begin{bmatrix} e_1 \\ e_5 \\ e_3 \end{bmatrix} \quad (21)$$

Then ΔW has non-zero entries in the 1^{st} , 5^{th} , and 3^{rd} columns, populated by b_1 , b_2 , and b_3 , respectively. The SBoRA-FB variant constructs B_{sb} similarly, yielding row-wise sparsity. As shown in Figure 6, StrucTune-FAB integrates both approaches, expressing the weight adaptation as:

$$\Delta W = BA_{sb} + B_{sb}A. \quad (22)$$

StructTune introduces a sensitivity-based selection strategy that outperforms random sampling by generating binary masks to configure adaptation modules, thereby enhancing both task relevance and computational efficiency.

4 Experiments

We assessed StructTune’s effectiveness through fine-tuning across tasks spanning mathematics, coding, and medicine, using LLaMA models (Touvron et al., 2023; Grattafiori et al., 2024) spanning different sizes and versions and Qwen

GSM8K												
method	rank	LLaMA2-7B		Score		LLaMA3.2-3B		Score		Qwen3-4B		Score
		#TP	Spa_density	Epoch1	Best of E2-E4	#TP	Spa_density	Epoch1	Best of E2-E4	#TP	Spa_density	Epoch1
LoRA	64	159.9M	-	26.61	35.56	97.3M	-	32.68	39.95	132.1M	-	72.93
RoSA(spa)	-	155.4M	2.4%	21.23	32.52	97.2M	3.45%	34.57	36.24	132.3M	3.64%	67.93
SBoRA-FA		155.0M	2.4%	19.48	30.48	96.5M	3.45%	37.75	42.68	132.1M	3.64%	74.45
FAB_random		154.0M	2.4%	20.62	31.99	95.9M	3.45%	36.92	40.86	130.6M	3.64%	72.48
StructTune-FA(ours)		155.0M	2.4%	29.42	36.16	96.5M	3.45%	40.11	45.56	132.1M	3.64%	75.13
StructTune-FAB(ours)		154.6M	2.4%	29.49	36.69	96.4M	3.45%	39.58	43.52	131.3M	3.64%	75.51
LoRA	32	80.0M	-	24.64	33.21	48.6M	-	34.04	41.85	66.1M	-	72.33
RoSA(spa)	-	77.7M	1.2%	25.25	30.71	48.6M	1.725%	34.72	38.21	66.1M	1.82%	57.85
SBoRA-FA		77.5M	1.2%	17.36	25.7	47.9M	1.725%	34.57	39.35	65.5M	1.82%	73.69
FAB_random		77.0M	1.2%	18.95	28.13	47.9M	1.725%	33.51	37.91	65.3M	1.82%	71.49
StructTune-FA(ours)		77.4M	1.2%	26.61	33.43	47.9M	1.725%	36.69	40.49	65.5M	1.82%	75.59
StructTune-FAB(ours)		77.5M	1.2%	24.49	35.25	48.1M	1.725%	37.91	46.1	65.6M	1.82%	75.44
LoRA	16	40.0M	-	30.25	33.13	24.3M	-	35.63	38.74	33.0M	-	72.33
RoSA(spa)	-	38.9M	0.6%	16.76	24.41	24.3M	0.862%	32.75	38.74	33.1M	0.91%	72.93
SBoRA-FA		38.1M	0.6%	14.78	20.62	23.9M	0.862%	32.14	35.33	32.7M	0.91%	74.53
FAB_random		38.2M	0.6%	14.86	18.88	24.0M	0.862%	32.45	36.54	32.5M	0.91%	70.66
StructTune-FA(ours)		38.1M	0.6%	25.4	29.11	23.9M	0.862%	33.43	38.06	32.7M	0.91%	75.44
StructTune-FAB(ours)		38.4M	0.6%	25.93	32.07	24.0M	0.862%	35.03	39.12	32.7M	0.91%	75.44

Table 1: Performance comparison of fine-tuning LLaMA2-7B, LLaMA3.2-3B, and Qwen3-4B on GSM8K across three parameter budgets. #TP denotes trainable parameters, controlled by rank in LoRA and by sparsity density (threshold p%) in sparse methods.

model (Yang et al., 2025). Comparison baselines include LoRA (Hu et al., 2022), SBoRA-FA (Po et al., 2024), RoSA with sparsity components only (RoSA(spa)) (Nikdan et al., 2024), and a random row-column selection baseline (FAB_random). Since standard RoSA combines LoRA with element-wise sparse components rather than pure sparse updates, we created LoRA + StructTune-FAB to further demonstrate the superiority of structured sparsity and for fair comparison. We implemented StructTune using both SDDMM framework (for consistency with RoSA) and SBoRA-based implementation (for computational efficiency analysis).

4.1 Fine-tuning Evaluation

Mathematical Evaluation: We evaluated StructTune on the GSM8K dataset (Cobbe et al., 2021), which contains 8.5K grade-school math problems for assessing basic arithmetic proficiency and reasoning ability. Using LLaMA2-7B, LLaMA3.2-3B, and Qwen3-4B models (Touvron et al., 2023; Grattafiori et al., 2024; Yang et al., 2025), we followed RoSA’s hyperparameter settings with LoRA rank $r = 64$ and 25% of the training data for sparsity selection. Experiments were conducted on NVIDIA RTX 3090 and 4090 GPUs.

Following RoSA’s experimental design, we report single-epoch (Epoch1) results for all models and extended-epoch (E2-4) results only for

LLaMA backbones in Table 1. Given its robust base performance, a single epoch sufficed for Qwen3-4B in this evaluation. We evaluated three parameter budgets, with LoRA’s count determined by rank r and sparse methods’ by sparse density (Spa_density):

$$Spa_density = \sum_{m=1}^M \frac{r_m \times (d_m + k_m)}{d_m \times k_m}, \quad (23)$$

where r_m , d_m , and k_m denote the rank, input dimension, and output dimension of layer m . Due to structured selection constraints, StructTune’s parameter counts slightly deviate from unstructured methods. On Epoch1, StructTune shows clear advantages across both LLaMA and Qwen backbones, and in the LLaMA extended-epoch setting it continues to outperform baselines at higher parameter budgets, confirming that structured sparse methods benefit from higher ranks. At the lowest budget (e.g., $r = 16$), StructTune remains competitive but does not always consistently surpass dense LoRA, suggesting that its structured selection advantage becomes more stable when sufficient rank capacity is available. Ablations show StructTune variants significantly outperform random selection counterparts.

To further demonstrate structured sparsity advantages over RoSA’s element-wise approach and

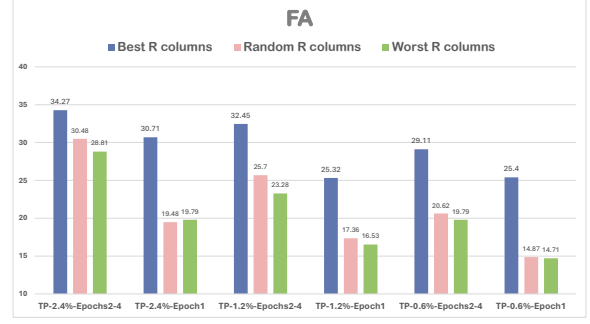
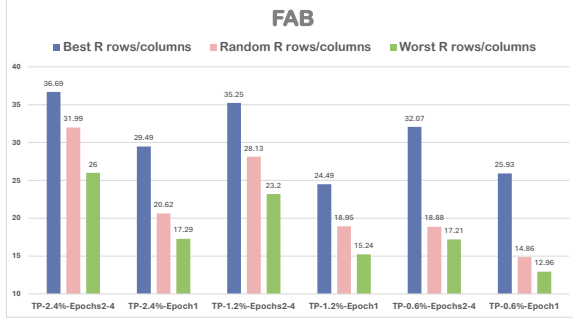


Figure 7: Ablation study on structured row-wise/column-wise sparse components using best, random, and worst R rows/columns based on sensitivity scores. Conducted with LLaMA2-7B on GSM8K. Performance follows the order: best R > random R > worst R .

GSM8K					
method	LLaMA3.2-3B			Score	
	#TP	rank	Spa_density	Epoch1	Best of E2-E4
LoRA	97.3M	64	-	32.68	39.95
RoSA	97.2M	48	0.862%	36.09	-
	97.3M	32	1.725%	37.22	-
	97.2M	16	2.587%	38.66	40.11
LoRA + StructTune-FAB	96.9M	48	0.862%	35.25	-
	96.8M	32	1.725%	38.51	-
	96.7M	16	2.587%	39.04	42.91
LoRA	48.6M	32	-	34.04	41.85
RoSA	48.6M	24	0.431%	35.86	39.73
	48.6M	16	0.862%	35.25	-
	48.6M	8	1.294%	34.87	-
LoRA + StructTune-FAB	48.4M	24	0.431%	34.5	-
	48.3M	16	0.862%	35.86	-
	48.3M	8	1.294%	39.12	42.84
LoRA	24.3M	16	-	35.63	38.74
RoSA	24.3M	12	0.216%	34.95	40.56
	24.3M	8	0.431%	32.6	-
	24.3M	4	0.647%	32.22	-
LoRA + StructTune-FAB	24.0M	12	0.216%	36.16	41.09
	24.1M	8	0.431%	35.1	-
	24.1M	4	0.647%	35.86	-

Table 2: Performance comparison between RoSA and LoRA+StructTune on LLaMA3.2-3B across three parameter budgets. #TP denotes trainable parameters; Rank and Spa_density control LoRA and sparse component parameters respectively.

established a through comparative with standard RoSA, we compared LoRA+StructTune-FAB against standard RoSA on LLaMA3.2-3B across varying parameter budgets. Table 2 shows LoRA+StructTune-FAB achieves superior performance across all configurations, surpassing even standard LoRA in the lowest-parameter setting.

Coding Evaluation: To evaluate StructTune’s code generation performance, we used the Magicoder-Evol-Instruct-110K dataset (Wei et al., 2023) for training and evaluated on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) benchmarks, which test Python programming fundamentals and comprehensive coding abili-

ties respectively. We fine-tuned both LLaMA (Grattafiori et al., 2024) and Qwen (Yang et al., 2025) models for one epoch. For first-stage weight importance analysis, we used 10% of training data, with other hyperparameters matching the GSM8K experiments. Table 3 shows that for Qwen3-4B, StructTune-FAB achieving the highest performance on both benchmarks for the two higher-parameter budgets and largely outperforms StructTune-FA, this gap suggests that restricting updates to a single dimension (columns only) may ignore critical task-relevant knowledge stored in row-aligned neurons. At the lowest parameter setting, it leads on HumanEval and closely follows the best performer on MBPP. For Llama3.2-3B, StructTune-FA achieves the best HumanEval scores across all three parameter budgets and the best (or tied-best) MBPP results in higher-budget settings, while remaining competitive at the lowest budget. Random-selection methods (SBoRA-FA and FAB_random) show greater sensitivity to parameter reductions, with significant performance degradation compared to StructTune. This demonstrates that coding tasks are particularly sensitive to sparse component selection, highlighting the effectiveness of our sensitivity-based approach.

MedicalQA Evaluation: For domain-specific medical adaptation, we fine-tuned on the MedQA-USMLE dataset (Jin et al., 2021) and evaluated accuracy on its test split using the LLaMA and Qwen models. We report results under the same three parameter budgets as other tasks for fair comparison (LoRA ranks 64/32/16 and matched sparse densities), as summarized in the Medical columns of Table 3. Across both model families and different parameter budgets, StructTune-FA achieves the best performance. StructTune-FAB shows an overall second-best trend and remains close to the

Coding fine-tuning: Magicoder-Evol-Instruct-110K; Medical fine-tuning: MedQA											
method	rank	Llama3.2-3B					Qwen3-4B				
		#TP	_density	Coding		Medical	#TP	Spa_density	Coding		Medical
				HumanEval	MBPP	MedQA			HumanEval	MBPP	MedQA
LoRA	64	97.3M	-	34	42.07	48.94	132.1M	-	64.63	51.6	57.27
RoSA(spa)	-	97.2M	3.45%	33.2	43.9	19.72	132.3M	3.64%	67.07	57.2	41.24
SBoRA-FA		96.5M	3.45%	36.4	44.51	48.47	132.1M	3.64%	62.8	41	57.82
FAB_random		95.9M	3.45%	35.4	41.46	47.05	130.6M	3.64%	56.71	40	57.03
StructTune-FA(ours)		96.5M	3.45%	39	45.12	49.88	132.1M	3.64%	65.85	47	59.23
StructTune-FAB(ours)		96.4M	3.45%	35.8	43.9	47.29	131.3M	3.64%	68.29	59.4	57.5
LoRA	32	48.6M	-	34.2	38.14	49.41	66.1M	-	68.9	49.8	58.05
RoSA(spa)	-	48.6M	1.725%	33.2	42.68	36.21	66.1M	1.82%	71.34	58.4	50.04
SBoRA-FA		47.9M	1.725%	35.8	40.85	49.25	65.5M	1.82%	26.22	9.8	58.21
FAB_random		47.9M	1.725%	36	39.63	48.39	65.3M	1.82%	25.61	8	58.13
StructTune-FA(ours)		47.9M	1.725%	39.2	42.68	50.12	65.5M	1.82%	65.85	40	59.86
StructTune-FAB(ours)		48.1M	1.725%	37.8	41.46	49.25	65.6M	1.82%	73.17	61	59.47
LoRA	16	24.3M	-	33.2	42.68	49.1	33.0M	-	62.8	51.8	57.19
RoSA(spa)	-	24.3M	0.862%	33	40.85	19.17	33.1M	0.91%	70.73	58.6	53.97
SBoRA-FA		23.9M	0.862%	36.2	38.41	48.47	32.7M	0.91%	25	6.2	57.19
FAB_random		24.0M	0.862%	34.8	37.2	47.45	32.5M	0.91%	19.51	6.2	57.66
StructTune-FA(ours)		23.9M	0.862%	36.6	42.07	49.96	32.7M	0.91%	46.95	20.8	59.07
StructTune-FAB(ours)		24.0M	0.862%	35.6	42.07	49.49	32.7M	0.91%	71.95	55.2	59

Table 3: Performance comparison of Llama3.2-3B and Qwen3-4B across three parameter budgets: coding benchmarks (HumanEval, MBPP) after Magicoder-Evol-Instruct-110K fine-tuning, and accuracy after MedQA fine-tuning (Medical). #TP denotes trainable parameters, controlled by rank in LoRA and by sparsity density (threshold p%) in sparse methods.

top in most settings, indicating that the structured row/column sparse design is robust for medical QA. In contrast, RoSA(spa) is clearly behind, especially on Llama3.2-3B where its MedQA scores drop sharply (e.g., 19.72 and 19.17), revealing the limitation of element-wise sparsity for this task.

4.2 Ablation Study

To validate that sensitivity-based selection identifies genuinely important components, we compared three selection strategies on LLaMA2-7B using GSM8K: best R (selecting rows/columns with the highest sensitivity scores), random R (baseline random selection), and worst R (selecting rows/columns with the lowest sensitivity scores). Figure 7 shows a clear performance hierarchy: best $R >$ random $R >$ worst R , with substantial margins. The pronounced gap between best and worst selection demonstrates that sensitivity scores provide meaningful parameter importance rankings. Furthermore, the performance differences confirm that intelligent component selection is critical for mathematically intensive tasks, validating our sensitivity-based approach.

To validate sensitivity-based selection, we compared our method against selecting R rows/columns by gradient magnitude (similar to RoSA but using structured selection). Table 5 shows sensitivity-based selection outperforms

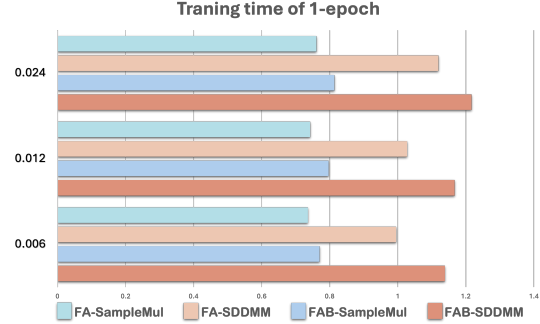


Figure 8: Single epoch training time comparison between SDDMM (RoSA) and SBoRA’s SampleMul implementations. Conducted with LLaMA2-7B on GSM8K across three sparse density levels: 2.4%, 1.2%, and 0.6%.

gradient-based selection across most sparsity ratios. This superiority stems from two factors: (1) sensitivity analysis captures the full impact of parameter removal on model performance, while gradient magnitude only reflects immediate loss reduction, and (2) sensitivity values exhibit clearer row/column patterns, making them more suitable for structured sparsity selection.

To further examine the advantage of structured sparsity, we conduct a metric-level ablation by applying sensitivity to element-wise sparsity and comparing it with the standard gradient-based element-wise baseline. Results in Table 6 show that sensitivity generally improves element-

Method	Density (Sparse)	Inference Speed (tokens/s) $\uparrow$	Memory (GiB) $\downarrow$
FAB-SDDMM	2.4%	873.50	21.28
	1.2%	910.27	20.27
	0.6%	930.77	19.78
FAB-SampleMul	2.4%	1301.90	20.57
	1.2%	1343.65	19.89
	0.6%	1379.16	19.49
FA-SDDMM	2.4%	948.98	21.28
	1.2%	1034.67	20.27
	0.6%	1067.92	19.83
FA-SampleMul	2.4%	1395.54	20.09
	1.2%	1431.02	19.29
	0.6%	1443.52	18.86

Table 4: Computation and memory efficiency comparison between SDDMM (RoSA) and SBoRA’s SampleMul implementations. Conducted with LLaMA2-7B on GSM8K across three sparse density levels: 2.4%, 1.2%, and 0.6%.

		FAB		FA	
		Score		Score	
Selection	Spa_density	Epoch1	Epochs2-4	Epoch1	Epochs2-4
Sensitivity	2.4%	29.49	36.69	30.71	34.27
Gradient		21	32.07	23.5	31.31
Sensitivity	1.2%	24.49	35.25	25.32	32.45
Gradient		18.8	31.39	18.57	28.2
Sensitivity	0.6%	25.93	32.07	25.4	29.11
Gradient		22.44	32.63	20.77	27.22

Table 5: Ablation study comparing sensitivity-based vs. gradient-based selection for choosing R rows/columns using LLaMA2-7B on GSM8K. Results show sensitivity-based selection outperforms gradient magnitude-based approaches.

wise sparsity over gradient in most cases, but it still remains behind structured sparsity.

4.3 Computation & Memory efficiency comparison

Our structured row/column-wise sparse design enables implementation through standard low-rank matrices that selectively update specific rows or columns rather than using SDDMM. This selective updating improves computational efficiency by eliminating redundant calculations and reducing memory access overhead.

We evaluated the computation and memory efficiency of standard-basis sample-based multiplication (SampleMul) versus SDDMM implementations on LLaMA2-7B with GSM8K, testing FA and FAB variants across various parameter levels using NVIDIA GeForce RTX 3090 GPUs. Results in Table 4 show that SampleMul consistently surpasses SDDMM across configurations in terms of inference speed and GPU occupation, with FA marginally outperforming FAB in both implemen-

Spa_density	Epoch	Element-wise Spa		Structured Spa
		RoSA		StructTune
		Gradient	Sensitivity	Sensitivity
2.4%	Epoch1	21.23	<u>29.49</u>	30.71
	Epochs2-4	32.52	<u>36.16</u>	36.69
1.2%	Epoch1	<u>25.25</u>	23.73	25.32
	Epochs2-4	30.71	<u>30.86</u>	35.25
0.6%	Epoch1	16.76	<u>24.41</u>	25.93
	Epochs2-4	24.41	<u>29.57</u>	32.07

Table 6: Structured sparsity ablation at three sparsity levels: element-wise sparsity (RoSA) with gradient (baseline) vs. sensitivity, and structured sparsity (StructTune) with sensitivity, reported at Epoch1 and Epochs2-4. The median performance is underlined.

tations. Similar trends can be observed in training time (Figure 8). These findings confirm the computational benefits of our structured sparsity design, enabling more efficient sparse computation strategies than traditional SDDMM operations.

4.4 Additional Experiments

Appendix §B collects the follow-up studies that extend the main evaluation. Firstly, due to space constraints, additional evaluation on LLaMA2-7B is provided in §B.1. Moreover, we include evaluation of row-wise StructTune-FB for completeness (§B.2). Then we introduce StructTune-TA/TAB, which keeps the structured sparsity pattern but swap frozen unit entries for trainable scalars (§B.3). Finally, we compare zero-shot MMLU accuracy before and after one epoch of GSM8K fine-tuning for StructTune and unstructured RoSA_spa at matched sparsity densities, to assess how different sparse update styles interact with general-purpose knowledge when adapting on a downstream task (§B.4).

5 Discussion & Conclusion

In this paper, we introduced StructTune, a novel approach for sparse weight updates in LLM fine-tuning through row-wise and column-wise structured sparse components. It stems from the key observation that LLM weights exhibit distinct row-wise and column-wise distribution patterns, with fine-tuning providing non-uniform amplification across different rows and columns, and draws inspirations from neuron level LLM internal mechanisms analyzing and rows/columns-based pruning research.

Our sensitivity-based analysis approach circumvents the computational burden of full weight

updates while maintaining analytical rigor through multi-step accumulation. Experimental evaluation on challenging tasks including mathematical reasoning and code generation demonstrates that structured sparse weight updates outperform both dense weight updates and unstructured sparse alternatives. Ablation studies further confirms the effectiveness of our sensitivity-based weight importance analysis framework. A limitation of our approach is the extra “pretraining” stage needed to locate salient parameters, which adds training overhead versus single-stage fine-tuning; future work could unify sensitivity estimation with the main adaptation loop.

By exploiting structured sparsity for computational efficiency and principled weight selection through rigorous analysis, we advance understanding of structured sparsity in LLM fine-tuning and offer practical solutions for efficient large-scale parameter updates.

Acknowledgments

We thank all co-authors for their dedicated efforts and contributions throughout this work. We are also grateful to the Action Editor, Professor Hai Zhao, and the anonymous reviewers for their insightful and constructive comments, which substantially improved this paper.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Aradhye Agarwal, Suhas Kamasetty Ramesh, Ayan Sengupta, and Tanmoy Chakraborty. 2025. Step-by-step unmasking for parameter-efficient fine-tuning of large language models. *Transactions of the Association for Computational Linguistics*, 13:1767–1788.
- Shuang Ao, Yi Dong, Jinwei Hu, and Sarvapali D Ramchurn. 2025. Safe pruning lora: Robust distance-guided pruning for safety alignment in adaptation of llms. *Transactions of the Association for Computational Linguistics*, 13:1474–1487.
- Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicept: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.
- Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. 2020. Sparse gpu kernels for deep learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

- Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022. [SparseAdapter: An easy approach for improving the parameter-efficiency of adapters](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2184–2190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.
- Yijiang Liu, Huanrui Yang, Youxin Chen, Rongyu Zhang, Miao Wang, Yuan Du, and Li Du. 2025. Pat: Pruning-aware tuning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24686–24695.
- Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272.
- Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. 2023. Fact finding: Attempting to reverse-engineer factual recall on the neuron level. In *Alignment Forum*, page 6.
- Mahdi Nikdan, Soroush Tabesh, Elvir Crnčević, and Dan Alistarh. 2024. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*.
- Lai-Man Po, Yuyang Liu, Haoxuan Wu, Tianqi Zhang, Wing-Yin Yu, Zhuohan Wang, Zeyu Jiang, and Kun Li. 2024. Sbor: Low-rank adaptation with regional weight updates. *arXiv preprint arXiv:2407.05413*.
- Daking Rai and Ziyu Yao. 2024. [An investigation of neuron activation as a unified lens to explain chain-of-thought eliciting arithmetic reasoning of LLMs](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7174–7193, Bangkok, Thailand. Association for Computational Linguistics.
- Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*.
- Jie Shao and Jianxin Wu. 2025. [Who reasons in the large language models?](#)
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama

2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Cheng, Tuo Zhao, and Jing Gao. 2024. Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. *arXiv preprint arXiv:2406.10777*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Empowering code generation with oss-instruct. *arXiv preprint arXiv:2312.02120*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zeping Yu and Sophia Ananiadou. 2023. Neuron-level knowledge attribution in large language models. *arXiv preprint arXiv:2312.12141*.

Zeping Yu and Sophia Ananiadou. 2024. Interpreting arithmetic mechanism in large language models through comparative neuron analysis. *arXiv preprint arXiv:2409.14144*.

Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. 2021. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*.

Bo Zhang, Hui Ma, Dailin Li, Jian Ding, Jian Wang, Bo Xu, and HongFei Lin. 2025a. Efficient tuning of large language models for knowledge-grounded dialogue generation. *Transactions of the Association for Computational Linguistics*, 13:1007–1031.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025b. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *arXiv preprint arXiv:2504.07448*.

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

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR.

A Supplementary Details

A.1 Weight Importance Estimation Logic

Preliminaries: Linear Layer and Adaptation.

Consider a linear layer m with input $x^m \in \mathbb{R}^{1 \times k}$, the output $y^m \in \mathbb{R}^{1 \times d}$ is computed using the pre-trained weight matrix $W_0^m \in \mathbb{R}^{d \times k}$ and an auxiliary bias $\tilde{b}^m \in \mathbb{R}^{1 \times d}$:

$$y^m = x^m (W_0^m)^T + \tilde{b}^m \quad (24)$$

In the context of PEFT, the weight matrix is adapted by adding a trainable component ΔW . The fine-tuned forward pass becomes:

$$y^m = x^m (W_0^m + \Delta W^m)^T + \tilde{b}^m \quad (25)$$

Following the LoRA framework, we decompose the adaptation ΔW into two low-rank matrices, $B^m \in \mathbb{R}^{d \times r}$ and $A^m \in \mathbb{R}^{r \times k}$, where $r \ll \min(d, k)$.

Cumulative and Instantaneous States. We distinguish between the weight state at the start of a training step and the state after an infinitesimal update. Let B_τ^m and A_τ^m represent the cumulative weights from previous steps $\{1, \dots, \tau - 1\}$. The adapted weight matrix at this step is:

$$W_\tau^{m'} = W_0^m + B_\tau^m A_\tau^m \quad (26)$$

During backpropagation at step τ , the optimizer computes the instantaneous gradient updates δB_τ^m and δA_τ^m . The weight state is briefly perturbed to $(B_\tau^m + \delta B_\tau^m)$ and $(A_\tau^m + \delta A_\tau^m)$ to evaluate the loss change. The modified forward pass is expressed as:

$$y^m = x^m [W_0^m + (B_\tau^m + \delta B_\tau^m)(A_\tau^m + \delta A_\tau^m)]^T + \tilde{b}^m \quad (27)$$

Derivation of the Instantaneous Matrix Update δW_τ^m . To quantify the total change in the weight matrix at step τ , we define the instantaneous update δW_τ^m as the difference between the updated state and the current adapted state $W_\tau^{m'}$:

$$\delta W_\tau^m = [W_0^m + (B_\tau^m + \delta B_\tau^m)(A_\tau^m + \delta A_\tau^m)] - W_\tau^{m'} \quad (28)$$

Substituting Equation 26 into the expression and cancel the common terms W_0^m and $B_\tau^m A_\tau^m$, we obtain the final form of $W_\tau^{m'}$:

$$\delta W_\tau^m = B_\tau^m(\delta A_\tau^m) + (\delta B_\tau^m)A_\tau^m + (\delta B_\tau^m)(\delta A_\tau^m) \quad (29)$$

A.2 Physical Meaning of Sensitivity Score

The sensitivity score $I(w_{ij}^m)$ is defined to quantify the importance of a weight element relative to the task loss $\mathcal{L}$.

Using a first-order Taylor series expansion, the change in loss $\mathcal{L}$ when a weight w is perturbed can be approximated as:

$$\mathcal{L}(w + \Delta w) \approx \mathcal{L}(w) + \frac{\partial \mathcal{L}}{\partial w} \Delta w \quad (30)$$

In the context of parameter selection, we are interested in the impact of setting a weight to zero, i.e., $\Delta w = -w$. The absolute change in loss is:

$$|\Delta \mathcal{L}| = |\mathcal{L}(0) - \mathcal{L}(w)| \approx \left| \frac{\partial \mathcal{L}}{\partial w} \cdot (0 - w) \right| = |\delta w \cdot w| \quad (31)$$

Thus, the sensitivity score $I_\tau(w_{ij}^m)$ in Equation (14) represents the first-order approximation of the loss magnitude change:

$$I_\tau(w_{ij}^m) = |\delta W_\tau^m \cdot W_\tau^{m'}|_{ij} \quad (32)$$

Higher sensitivity indicate that the model's performance is more dependent on those parameters, justifying their selection for structured updates.

A.3 Theoretical Motivation for Row- and Column-Wise Sparsity

Rows and Columns as Atomic Computational Units. In the matrix-vector product $y = x(W)^T$ with $W \in \mathbb{R}^{d \times k}$, entire rows and columns constitute the minimal *complete* functional units of the computation: each row uniquely and exclusively determines one output scalar, while each column provides the complete directional contribution of one input feature to the full output vector.

Proof. We examine two equivalent decompositions of the same product $y = x(W)^T$.

Row-wise decomposition and output completeness. The i -th output scalar can be written as an inner product with the i -th row:

$$y_i = \langle w_{i,*}, x \rangle = \sum_{j=1}^k w_{ij} x_j, \quad i = 1, \dots, d. \quad (33)$$

Row i is both *necessary* and *sufficient* for y_i : it is the sole row that contributes to y_i , and its contribution covers y_i entirely. Consequently, zeroing

the entire i -th row produces a *complete and isolated* suppression of the corresponding output dimension,

$$w_{i,*} \leftarrow 0 \implies \Delta y = -y_i \mathbf{e}_i, \quad (34)$$

where $\mathbf{e}_i$ is the i -th standard basis vector. The perturbation is confined strictly to a single output coordinate and eliminates it completely, with no side-effects on any other dimension.

Column-wise decomposition and input-feature completeness. Rewriting the same product as a sum of rank-one terms,

$$y = \sum_{j=1}^k x_j w_{*,j}, \quad (35)$$

reveals that column j is the unique carrier of all information about input feature x_j in the output vector y . The column $w_{*,j}$, scaled by x_j , contributes to *every* output dimension simultaneously and does so in a single, coherent direction. Zeroing the entire j -th column therefore produces a complete removal of input feature x_j from the computation,

$$w_{*,j} \leftarrow 0 \implies \Delta y = -x_j w_{*,j}, \quad (36)$$

an effect that is semantically equivalent to *disconnecting the j -th input channel* from all downstream outputs.

Fragmentation under element-wise sparsity. By contrast, zeroing a single element w_{ij} produces a *fragmented* perturbation,

$$w_{ij} \leftarrow 0 \implies \Delta y = -x_j w_{ij} \mathbf{e}_i, \quad (37)$$

which is simultaneously partial with respect to output dimension i (it only removes the x_j contribution to y_i , not the full y_i) and partial with respect to input feature j (it disrupts x_j 's influence only on y_i , not on all outputs). This mixed, fragmentary effect corresponds to no semantically coherent functional unit in the model's computation graph.

Summary of output perturbations.

$$\begin{aligned} \text{Row sparsity: } \Delta y &= -y_i \mathbf{e}_i \\ &\text{(complete output-neuron removal),} \\ \text{Column sparsity: } \Delta y &= -x_j w_{*,j} \\ &\text{(complete input-channel removal),} \\ \text{Element sparsity: } \Delta y &= -x_j w_{ij} \mathbf{e}_i \\ &\text{(partial, semantically ambiguous).} \end{aligned} \quad (38)$$

Medical fine-tuning: MedQA				
method	rank	Llama2-7B		
		#TP	Spa_density	Medical MedQA
LoRA	64	159.9M	-	34.72
RoSA(spa)	-	155.4M	2.4%	19.8
SBoRA-FA		155.0M	2.4%	37.71
FAB_random		155.4M	2.4%	37.31
StructTune-FA(ours)		155.0M	2.4%	38.1
StructTune-FAB(ours)		154.6M	2.4%	39.98
LoRA	32	80.0M	-	34.48
RoSA(spa)	-	77.7M	1.2%	19.95
SBoRA-FA		77.5M	1.2%	35.58
FAB_random		77.0M	1.2%	37.55
StructTune-FA(ours)		77.4M	1.2%	37.31
StructTune-FAB(ours)		77.5M	1.2%	37.86
LoRA	16	40.0M	-	35.11
RoSA(spa)	-	38.9M	0.6%	19.8
SBoRA-FA		38.1M	0.6%	27.49
FAB_random		38.2M	0.6%	25.06
StructTune-FA(ours)		38.1M	0.6%	34.48
StructTune-FAB(ours)		38.4M	0.6%	35.19

Table 7: Performance comparison of Llama2-7B across three parameter budgets on MedQA after fine-tuning. #TP denotes trainable parameters, controlled by rank in LoRA and by sparsity density (threshold p%) in sparse methods.

This establishes that structured sparsity at the row or column level produces well-defined, *semantically complete* changes to the model’s output, whereas element-wise sparsity introduces partial disruptions that are interpretable neither as output-neuron ablations nor as input-channel ablations. Rows and columns are therefore not merely convenient groupings—they are the natural atomic units through which the network’s computational function is organized, and the appropriate granularity at which sparsity should be applied. $\square$

B Additional Experimental Results

B.1 Additional LLaMA2-7B Fine-tuning Evaluation Results

Due to space constraints, we only report the results for LLaMA2-7B on GSM8K in the main text. Here we provide the additional results for LLaMA2-7B on the medical evaluation (Table 7). However, the coding evaluation for LLaMA2-7B is omitted due to GPU memory limitations that exceed our available hardware capacity.

B.2 StructTune-FB Evaluation

For completeness, we also evaluate StructTune-FB on the GSM8K under the same setup as the main experiments (LLaMA2-7B, LLaMA3.2-3B, and Qwen3-4B; single-epoch evaluation at

three matched sparsity densities). Results are summarized in Table 8. Across the two LLaMA backbones, StructTune-FB consistently trails StructTune-FAB and StructTune-FA, especially in lower-budget settings, whereas on Qwen3-4B it remains broadly competitive and sometimes approaches the best variant. This model-dependent trend suggests that preserving column-wise structure in linear layers may play a more critical role in mathematical reasoning for LLaMA-family models.

B.3 Replacing Frozen Unit Entries with Learned Scalars

In StructTune’s architecture design, one of the low-rank projection matrices is constructed using frozen standard basis vectors (A_{sb} for example). Considering this may constrain the expressive capacity imposed by this freezing, we conducted further testing by replacing the frozen unit entries with trainable scalars. Instead of using a frozen scalar of 1 in the original standard-basis low-rank matrix, we now employ a trainable scaling factor. This modification allows the model to learn optimal scaling during training while maintaining the structured low-rank formulation.

Originally for example:

$$A_{sb} = \begin{bmatrix} e_1 \\ e_5 \\ e_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix} \quad (39)$$

Now we replace frozen 1 with trainable parameter a_1 , a_5 and a_7 :

$$A'_{sb} = \begin{bmatrix} e'_1 \\ e'_5 \\ e'_3 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & a_5 & \dots \\ 0 & 0 & a_3 & 0 & 0 & \dots \end{bmatrix} \quad (40)$$

This approach preserves the structured sparse weight update property while introducing a trainable scaling factor for each adapted column. We conducted additional experiments using this structure (StructTune-TA and StructTune-TAB) and compared the results with the original StructTune-FAB and StructTune-FA variants. As shown in Table 9, in LLaMA2-7B, StructTune with trainable scaling factors achieves better performance at higher parameter budgets but underperforms at smaller parameter budgets. While for LLaMA3.2-3B and Qwen3-4B, it shows comparable performance. These results suggest that using only one trainable low-rank projection matrix is sufficient, justifying the use of frozen standard basis vectors in the original design.

Method	LLaMA2-7B		LLaMA3.2-3B		Qwen3-4B	
	Sparsity	Score	Sparsity	Score	Sparsity	Score
		1-Epoch		1-Epoch		1-Epoch
StructTune-FAB	2.4%	29.49	3.45%	39.58	3.64%	75.51
StructTune-FA	2.4%	29.42	3.45%	40.11	3.64%	75.13
StructTune-FB	2.4%	28.66	3.45%	38.21	3.64%	74.37
StructTune-FAB	1.2%	24.49	1.725%	37.91	1.82%	75.44
StructTune-FA	1.2%	26.61	1.725%	36.69	1.82%	75.59
StructTune-FB	1.2%	21.3	1.725%	35.63	1.82%	75.13
StructTune-FAB	0.6%	25.93	0.862%	35.03	0.91%	75.44
StructTune-FA	0.6%	25.4	0.862%	33.43	0.91%	75.44
StructTune-FB	0.6%	17.89	0.862%	31.61	0.91%	76.12

Table 8: Extended evaluation results for StructTune-FB on LLaMA and Qwen models fine-tuned on GSM8K. StructTune-FB underperforms StructTune-FAB and StructTune-FA on LLaMA while achieves comparable performance on Qwen models.

Method	LLaMA2-7B		LLaMA3.2-3B		Qwen3-4B	
	Sparsity	Score	Sparsity	Score	Sparsity	Score
		1-Epoch		1-Epoch		1-Epoch
StructTune-FAB	2.4%	29.49	3.45%	39.58	3.64%	75.51
StructTune-TAB	2.4%	31.31 (↑ 1.82)	3.45%	37.83 (↓ 1.75)	3.64%	73.16 (↓ 2.35)
StructTune-FA	2.4%	29.42	3.45%	40.11	3.64%	75.13
StructTune-TA	2.4%	30.1 (↑ 0.68)	3.45%	40.94 (↑ 0.83)	3.64%	72.63 (↓ 2.5)
StructTune-FAB	1.2%	24.49	1.725%	37.91	1.82%	75.44
StructTune-TAB	1.2%	28.58 (↑ 4.09)	1.725%	37.3 (↓ 0.61)	1.82%	74.53 (↓ 0.91)
StructTune-FA	1.2%	26.61	1.725%	36.69	1.82%	75.59
StructTune-TA	1.2%	27.82 (↑ 1.21)	1.725%	40.11 (↑ 3.42)	1.82%	75.44 (↓ 0.15)
StructTune-FAB	0.6%	25.93	0.862%	35.03	0.91%	75.44
StructTune-TAB	0.6%	25.32 (↓ 0.61)	0.862%	35.18 (↑ 0.15)	0.91%	74.91 (↓ 0.53)
StructTune-FA	0.6%	25.4	0.862%	33.43	0.91%	75.44
StructTune-TA	0.6%	22.59 (↓ 2.81)	0.862%	32.22 (↓ 1.21)	0.91%	76.12 (↑ 0.68)

Table 9: Extended evaluation comparing StructTune variants with trainable scaling factors (StructTune-TAB and StructTune-TA) for selected row/column updates on LLaMA and Qwen models fine-tuned on GSM8K.

Method	LLaMA2-7B		LLaMA3.2-3B		Qwen3-4B	
	Sparsity	Accuracy	Sparsity	Accuracy	Sparsity	Accuracy
Baseline (Pre-trained)	-	41.71	-	54.55	-	70.40
+StructTune-FAB	2.4%	39.56 (↓ 2.15)	3.45%	52.87 (↓ 1.68)	3.64%	71.22 (↑ 0.82)
+StructTune-FA	2.4%	37.61 (↓ 4.10)	3.45%	51.88 (↓ 2.67)	3.64%	70.73 (↑ 0.33)
+RoSA_spa	2.4%	33.65 (↓ 8.06)	3.45%	46.49 (↓ 8.06)	3.64%	65.75 (↓ 4.65)
+StructTune-FAB	1.2%	37.16 (↓ 4.55)	1.725%	52.07 (↓ 2.48)	1.82%	70.64 (↑ 0.24)
+StructTune-FA	1.2%	37.28 (↓ 4.43)	1.725%	52.74 (↓ 1.81)	1.82%	70.84 (↑ 0.44)
+RoSA_spa	1.2%	34.63 (↓ 7.08)	1.725%	45.39 (↓ 9.16)	1.82%	66.48 (↓ 3.92)
+StructTune-FAB	0.6%	36.88 (↓ 4.83)	0.862%	51.09 (↓ 3.46)	0.91%	71.41 (↑ 1.01)
+StructTune-FA	0.6%	37.99 (↓ 3.72)	0.862%	52.57 (↓ 1.98)	0.91%	71.15 (↑ 0.75)
+RoSA_spa	0.6%	34.02 (↓ 7.69)	0.862%	47.62 (↓ 6.93)	0.91%	67.68 (↓ 2.72)

Table 10: Comparison of MMLU accuracy before and after 1-epoch GSM8K fine-tuning for LLaMA and Qwen models using structured sparsity (StructTune-FA, StructTune-FAB) and unstructured sparsity (RoSA_spa) methods at varying sparsity densities.

B.4 General Capabilities Preservation Investigation

To investigate the catastrophic forgetting effect following fine-tuning with sparse weight updates, we conducted experiments comparing the models’ general capabilities before and after fine-tuning. We evaluated performance on the MMLU

(Hendrycks et al., 2020) dataset using a zero-shot setting.

We fine-tuned LLaMA and Qwen models on GSM8K for one epoch and compared the structured sparse update method (StructTune) against the unstructured sparse update method (RoSA). According to the results in Table 10, both meth-

ods demonstrate accuracy drops on the LLaMA model, though StrucTune exhibits a smaller decline than RoSA. For the Qwen model, StrucTune preserves general capabilities, while RoSA continues to show degradation. These experiments demonstrate that although sparse fine-tuning can affect general capabilities, structured sparse updates provide a superior solution compared to unstructured approaches.