UTPS: UNIVERSAL TAYLOR PROXY SCORING FOR SPARSE MULTIMODAL LOW-RANK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting Multimodal Large Language Models (MLLMs) to downstream tasks is computationally expensive. While Low-Rank Adaptation (LoRA) reduces trainable parameters, applying it uniformly across all layers is inefficient and can disrupt pretrained representations. We propose Universal Taylor Proxy Scoring (UTPS), a lightweight, training-free method that identifies the most task-relevant layers for sparse adaptation using only a single forward-backward pass. UTPS integrates gradient-weight sensitivity, activation-aware scaling, and parameter-density normalization to prioritize layers with the highest performance-per-parameter. Guided by UTPS scores, we design a sparse dynamic multi-expert LoRA architecture with multimodal-aware budget allocation. Experiments on InternVL2-2B, LLaVA-1.5-7B, InternVL2-8B, Qwen2.5-VL-3B, and LLaMA-2-7B demonstrate that UTPS, using only 50% of LoRA parameters, closely approaches or matches full LoRA performance across diverse multimodal and text-only benchmarks. Our work thus enables efficient, targeted adaptation of large models without compromising accuracy.

1 INTRODUCTION

The rapid scaling of MLLMs has revolutionized visual-language understanding, enabling applications from visual question answering (VQA) to complex scene description (Ballout et al., 2025). Models like InternVL and LLaVA achieve remarkable performance by integrating powerful vision encoders with large language models. However, this progress comes at a steep computational cost: adapting these billion-parameter models to downstream tasks requires substantial resources, creating a significant barrier for researchers and practitioners.

Parameter-efficient fine-tuning methods, particularly Low-Rank Adaptation (LoRA), have emerged as the standard for adapting large models (Liang & Bharadwaj, 2025). By injecting trainable low-rank matrices into linear layers, LoRA reduces trainable parameters by orders of magnitude while maintaining competitive performance. Methods such as DoRA (Liu et al., 2024b) and LoRA+ (Hayou et al., 2024) improve upon the original LoRA formulation by reparameterizing weight matrices or optimizing learning rates, while PiSSA (Meng et al., 2024) leverages principal singular values for more effective adaptation.

However, current LoRA implementations typically adopt a "one-size-fits-all" approach, applying adapters uniformly across all linear layers regardless of their architectural role or task relevance (Xu et al., 2026). Recent works have begun to challenge this uniform approach. MoLA (Gao et al., 2025) empirically demonstrates that higher transformer layers benefit from more LoRA experts, while LoraMoE (Dou et al., 2024) decouples experts for knowledge retention and task adaptation using MoE-style plugins. Similarly, in the context of model compression, SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) show that not all weights contribute equally to model performance, enabling one-shot pruning with minimal accuracy loss. We emphasize that UTPS is **not** a pruning method of this kind: SparseGPT and Wanda remove individual weights from the pretrained model to compress it, whereas UTPS only *selects which modules receive trainable LoRA adapters* during fine-tuning. The pretrained weights of the base model are left fully intact, and no weights are masked, zeroed out, or removed; the "sparsity" in UTPS refers solely to which layers carry the additive low-rank updates, not to the base weights.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The uniform application of LoRA overlooks a fundamental insight regarding the intrinsic architectural asymmetry found in multimodal models. Vision encoders process information through a hierarchical structure where early layers focus on extracting low-level features and deeper layers capture complex semantic abstractions. Language model layers similarly encode a variety of linguistic properties that range from basic syntactic patterns to sophisticated high-level reasoning. The cross-modal alignment established during the pretraining phase represents a delicate balance that indiscriminate fine-tuning can easily disrupt (Gao et al., 2026). By applying LoRA uniformly across all modules, we expend computational resources on layers with minimal relevance to the specific task while simultaneously risking catastrophic interference. This phenomenon occurs when adapting a single layer degrades performance on previously learned tasks or destabilizes the carefully calibrated multimodal alignment.

The central challenge lies in determining an intelligent method for selecting the specific layers that require adaptation for any given multimodal task. Current selection techniques typically depend on basic heuristics such as gradient norms (Ge et al., 2025), but these approaches fail to account for the intricate relationship between parameter sensitivity, activation dynamics, and architectural efficiency. These existing methods are constrained by several critical deficiencies. One major flaw is the failure to consider the magnitude of the weights themselves, an omission that overlooks how weight scale modulates the impact of gradient changes on the final loss. Additionally, these strategies lack data-driven activation feedback, which is essential for identifying which layers are actively engaged during the actual processing of a task. Furthermore, conventional importance scoring tends to disproportionately favor large layers with high parameter counts rather than identifying truly efficient modules, ultimately resulting in the suboptimal allocation of parameters.

To address these limitations, we introduce Universal Taylor Proxy Scoring (UTPS), a novel framework for efficient, sparse LoRA multimodal adaptation. Our approach makes three key contributions:

- We propose a unified scoring mechanism that combines static parameter sensitivity with dynamic activation feedback. We compute Taylor-W scores (the norm of gradient-weight products) to estimate how parameter changes affect the loss, while simultaneously using forward-pass hooks to capture activation magnitudes that reflect each layer's engagement with task-specific data. This dual perspective ensures our scoring accounts for both the model's architecture and its behavior on actual inputs.
- We introduce parameter-density normalization to identify truly efficient layers. By dividing the combined sensitivity-activation score by the number of trainable parameters in each layer, we obtain a "performance-per-parameter" density metric. This approach effectively prevents selection bias toward modules with large parameter counts and ensures that we prioritize layers delivering the highest impact relative to their size. Such a consideration is crucial for achieving efficient adaptation in large scale multimodal models.
- We develop a multimodal-aware sparse LoRA architecture based on UTPS scores. Recognizing that vision and language modalities have different adaptation requirements, we implement dynamic budget allocation that guarantees minimum adaptation capacity for the vision encoder, addressing the common problem of modality imbalance. From these scores, we generate task-specific sparse architectures and implement dynamic multi-expert, where different tasks activate different, optimized subsets of adapters, enabling efficient multi-task learning while minimizing interference.

Our experiments demonstrate that UTPS consistently identifies task-critical layers across diverse VQA tasks including Open-ended and Multi-choice VQA, and visual grounding. We emphasize that UTPS does not aim to surpass full LoRA; rather, using only 50% of the LoRA parameters selected by UTPS, we closely approach or match the performance of full LoRA adaptation on 6 benchmark tasks (gap $\leq$1.5 points on 5 tasks, with a slight gain on VizWiz). These results establish UTPS as a principled, efficient approach to sparse multimodal adaptation that bridges the gap between computational efficiency and model performance.

2 Related Work

Parameter-Efficient Fine-Tuning The rapid scaling of large models has made full fine-tuning prohibitively expensive. Low-Rank Adaptation (LoRA) (Hu et al., 2022) has become the dominant remedy, injecting trainable low-rank matrices into linear layers while freezing pretrained weights,

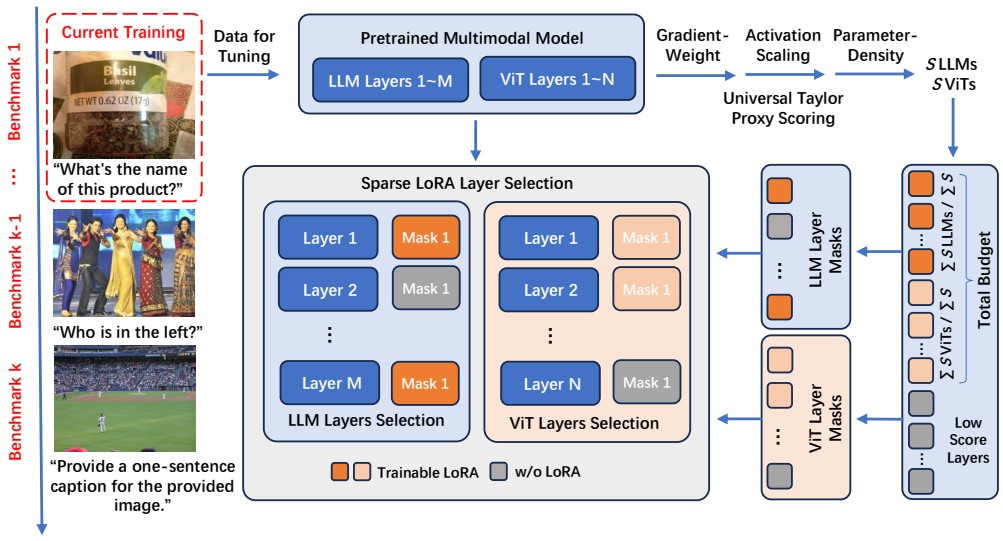


Figure 1: Overall framework of UTPS.

and has been extended to multimodal models such as LLaVA (Liu et al., 2023) and InstructBLIP (Dai et al., 2023). However, these methods typically apply adapters uniformly across all layers, overlooking the heterogeneous sensitivity of different architectural components to a given task.

Mixture of LoRA Experts LoRA has also been combined with the Mixture of Experts (MoE) paradigm to build sparse, modular adapters. MoLE (Wu et al., 2024) and LoraMoE (Dou et al., 2024) route inputs to layer-wise experts through learned gates, MoLA (Gao et al., 2025) allocates more experts to higher layers, D-MoLE (Ge et al., 2025) selects layers with gradient-based zero-cost proxies for continual multimodal tuning, and LoSA (Huang et al., 2025) matches LoRA sparsity masks so experts merge into already-sparsified models. These works show the promise of sparse LoRA experts, yet their layer selection rests on ad-hoc heuristics rather than principled per-layer efficiency estimates.

Multimodal Sensitivity Estimating layer importance has deep roots in zero-cost network analysis. Gradient-norm proxies (Xiang et al., 2023b; Poeta, 2023) and gradient-weight products such as Wanda's $|W|\cdot|\text{grad}|$ (Sun et al., 2024), inspired by first-order Taylor expansions of the loss (Molchanov et al., 2019), estimate importance without training. Our Taylor-W score applies this philosophy to the distinct setting of additive low-rank updates rather than multiplicative masking. In multimodal models, moreover, sensitivity is non-uniform across layers and modalities (Zhang et al., 2024; Dong et al., 2026; Gao et al., 2025); however, prior proxies ignore the magnitude of pretrained weights—which scales the effect of gradient changes—and the activation dynamics elicited by task data. UTPS integrates both signals.

3 Methodology

We introduce the UTPS framework shown in Figure 1, a training-free saliency estimation method designed to identify optimal layers for sparse LoRA adaptation in MLLMs. Throughout, the vision encoder is denoted by **ViT** and the language backbone by **LLM**; UTPS scores the layers of both modalities. The core insight driving UTPS is that not all transformer layers contribute equally to task adaptation, and their importance can be quantified through a combination of static parameter sensitivity, dynamic data-driven behavior, and architectural efficiency. Our method operates through three synergistic phases: gradient-based sensitivity estimation via Taylor expansion, dynamic activation capturing through forward hooks, and parameter-density normalization for efficiency-aware ranking.

3.1 TAYLOR-W: GRADIENT-WEIGHT SENSITIVITY FOR LAYER SELECTION

Traditional layer selection methods often rely on gradient magnitude $\|\nabla_\theta \mathcal{L}\|$ as a proxy for importance. However, this approach overlooks a critical factor: the magnitude of the pretrained weights θ themselves. In transformer architectures, particularly in multimodal models, weight distributions are highly non-uniform across layers due to architectural design choices, training dynamics, and modality-specific initialization. A small gradient applied to a large-weight layer can produce more significant functional changes than a large gradient applied to a small-weight layer.

To address this, we ground our approach in first-order Taylor expansion of the loss function $\mathcal{L}$. Consider perturbing the parameters θ of a layer by $\Delta\theta$ (as in LoRA's additive update $\theta_{new} = \theta + \Delta\theta$). The change in loss can be approximated as:

$$\Delta\mathcal{L} \approx \nabla_\theta \mathcal{L}^\top \Delta\theta. \tag{1}$$

If we consider $\Delta\theta$ as being proportional to θ itself (a reasonable assumption for LoRA that modify existing weight directions), then the influence of perturbing layer l becomes proportional to $\nabla_\theta \mathcal{L}^\top \theta$. This motivates our Taylor-W (TW) formulation, which captures the interaction between gradient sensitivity and weight magnitude.

We motivate Taylor-W as a **heuristic importance score** inspired by the first-order Taylor expansion: it captures the interaction between gradient alignment and weight scale, but we *do not* claim it is a rigorous approximation of LoRA's loss change. The LoRA update $\Delta\theta = BA$ is an additive low-rank update that is not a simple scaling of the pretrained weights, so the proportionality assumption in Eq. 1 holds only approximately. This heuristic is consistent with the practice in pruning literature, where gradient-weight products such as $|W| \cdot |\text{grad}|$ are used without a rigorous derivation from the exact update rule (Sun et al., 2024; Molchanov et al., 2019). Taylor-W is therefore **empirically validated rather than theoretically derived**: the element-wise product measures loss sensitivity along existing weight directions, and its effectiveness is confirmed by the ablations in Section 4.4.

We define the parameter influence $I(l)$ for a layer l as the Frobenius norm of the Hadamard product between the accumulated gradients and the pretrained weights. Given a set of target modules $\mathcal{T}$, for each module $m \in \mathcal{T}$ in layer l, the score is calculated as:

$$I(l) = \sum_{\theta \in m} \|\nabla_\theta \mathcal{L} \odot \theta\|_F, \tag{2}$$

where $\odot$ denotes the Hadamard (element-wise) product and $\|\cdot\|_F$ denotes the Frobenius norm. In our implementation, $\nabla_\theta \mathcal{L}$ is accumulated over a fraction of the training data, to provide a robust estimate of task-specific sensitivity.

Taylor-W offers several fundamental advantages over traditional gradient-based proxies. First, it achieves scale awareness through the element-wise product $\nabla_\theta \mathcal{L} \odot \theta$, which naturally amplifies gradients where pretrained weights are large. This correctly prioritizes layers where parameter changes yield greater functional impact, rather than treating all parameter updates equally. Beyond mere magnitude, Taylor-W introduces directional sensitivity by considering whether gradients align with existing weight directions. Unlike gradient magnitude alone, which cannot distinguish between constructive and destructive updates, this formulation naturally discounts orthogonal gradients that have minimal effect on model output. Furthermore, the approach exhibits inherent architectural invariance, as both gradient and weight matrices scale proportionally with layer dimensions. Consequently, the product formulation provides a self-normalizing property that helps compare importance across layers of different sizes without additional scaling heuristics.

3.2 ACTIVATION-AWARE SCALING VIA FORWARD HOOKS

While Taylor-W captures static parameter sensitivity, it fails to account for how layers actually behave during inference on task-specific data. Some layers may exhibit high theoretical sensitivity (large $\nabla_\theta \mathcal{L} \odot \theta$) but remain largely inactive for a particular task due to input characteristics or architectural bottlenecks. To incorporate this data-dependent perspective, activation-aware scaling (AS) employs non-intrusive forward hooks to monitor activation patterns during the forward pass over the sampled data subset $\mathcal{D}_{\text{sub}}$.

For each target module $m \in \mathcal{T}$ at layer l, we register a PyTorch forward hook that captures the output tensor $O_l \in \mathbb{R}^{B \times S \times D}$ during the forward pass, where B is batch size, S is sequence length, and D is

hidden dimension. Rather than storing the full tensor (which would be memory-prohibitive for large models), we compute a lightweight statistic: the mean absolute activation value:

$$a_l^{(t)} = \frac{1}{B \cdot S \cdot D} \sum_{b=1}^{B} \sum_{s=1}^{S} \sum_{d=1}^{D} |O_l^{(t)}[b, s, d]|, \tag{3}$$

where t indexes the forward pass (mini-batch). This statistic provides a robust measure of the layer's overall activation magnitude while being invariant to batch size and sequence length variations.

To smooth transient fluctuations and obtain stable activation estimates per layer, we maintain an exponential moving average of mini-batch statistics, balancing historical stability with recent responsiveness without storing full history. After processing all K mini-batches in $\mathcal{D}_{\text{sub}}$, we obtain the final activation factor $A(l)$ for each layer l. Mean absolute activation is a simple yet effective statistic, directly measuring layer activity on task data. We compared it with activation variance and dead-neuron fraction on TextCaps: mean absolute activation achieved 138.35 CIDEr, variance 138.12, and dead-neuron fraction 137.89. Given its slight superiority and simplicity, we adopt it as default (full analysis in Appendix L).

This recursive formulation offers notable practical benefits. It is memory efficient, requiring only a single scalar per layer instead of storing all historical values. It also adaptively weights recent observations, which makes the estimate responsive to the underlying data distribution rather than being dominated by outdated statistics. Furthermore, the exponential weighting naturally avoids numerical overflow issues that could arise from simple averaging over many batches, thereby providing stable and reliable activation estimates across diverse tasks.

3.3 Parameter-Density Normalization

A fundamental challenge in MLLM adaptation is architectural and parametric asymmetry: vision encoders and language models exhibit dramatically different parameter scales. In models like InternVL, the language backbone dominates the total parameter count (often comprising over 90% of the model), while the vision encoder, though potentially deep and sophisticated (e.g., a 6B-parameter InternViT), remains significantly smaller in total capacity. Simple scoring mechanisms that multiply sensitivity $I(l)$ by activation $A(l)$ would inherently favor parameter-heavy language layers, leading to suboptimal allocation where the adaptation budget is disproportionately consumed by the LLM, potentially neglecting critical visual feature refinement in the lighter but essential vision tower.

To address this, we introduce **parameter-density (PD) normalization**, which shifts the selection criterion from total importance to importance per parameter. In architectures like InternVL, where the language backbone dominates the total parameter count (often ¿90%) while the vision encoder remains significantly smaller yet critical for cross-modal alignment, standard scoring metrics inherently bias the budget towards the massive language layers. This often starves the visual tower of necessary updates, hindering effective modality fusion. For each layer l, we compute the density score:

$$S(l) = \frac{I(l) \cdot A(l)}{P_l}, \tag{4}$$

where P_l denotes the number of trainable parameters in layer l, and $I(l) \cdot A(l)$ represents the total sensitivity-activation product.

This normalization strategy offers two key benefits. First, it ensures fairness across scales by neutralizing the magnitude bias caused by the drastic parameter disparity between the deep language transformer and the compact vision encoder, thereby preventing parameter-sparse yet functionally vital visual layers from being overlooked. Second, it enables marginal utility maximization. By prioritizing layers with high $S(l)$, we target regions of the network where unitary parameter updates yield the highest performance gain per parameter, effectively focus on true performance bottlenecks.

We note an important interaction among the three components: because both Taylor-W (which multiplies gradients by weights) and activation scaling (which grows with layer throughput) tend to reward large layers, the parameter-density normalization is essential to prevent the combined score from being dominated by parameter-heavy language layers. Without it, the product $I(l) \cdot A(l)$ systematically over-weights the LLM, which is exactly the bias we observe in the w/o PD ablation (Table 2).

3.4 MULTIMODAL-AWARE BUDGET ALLOCATION ALGORITHM

The UTPS scores alone are insufficient for constructing optimal sparse architectures, as they don't account for the modality imbalance inherent in MLLMs. Different tasks rely on visual and linguistic features to varying degrees: VQA requires strong vision understanding, while text-rich captioning tasks depend more on language modeling capabilities.

We focus on key attention modules that are most responsive to fine-tuning and introduce **multimodal-aware budget allocation (MBA)**, which dynamically distributes adaptation capacity between ViT and LLM modules. Given UTPS scores for all layers, we first separate them by modality:

$$\begin{cases} \mathcal{S}_{\text{LLM}} = \{S(l) : l \in \text{language layers}\} \\ \mathcal{S}_{\text{ViT}} = \{S(l) : l \in \text{vision layers}\} \end{cases} \tag{5}$$

Let $B = \lfloor \rho \times N_{total} \rfloor$ be the total budget (number of layers to adapt), where ρ is the budget portion (e.g., 0.5 for 50% sparsity) and N_{total} is the total number of candidate layers. The approach would allocate budget proportionally to the total UTPS scores of each modality:

$$\begin{cases} r_{\text{LLM}} = \dfrac{\sum_{s \in \mathcal{S}_{\text{LLM}}} s}{\sum_{s \in \mathcal{S}_{\text{LLM}}} s + \sum_{s \in \mathcal{S}_{\text{ViT}}} s} \\ r_{\text{ViT}} = 1 - r_{\text{LLM}} \end{cases} \tag{6}$$

where r_{LLM} and r_{ViT} represent the proportion of the total budget allocated to language and vision modules, respectively. The numerator sums all UTPS scores within a modality, while the denominator sums scores across both modalities, ensuring the ratios sum to 1.

In contrast to gradient-norm-based scoring (e.g., D-MoLE (Ge et al., 2025)), which imposes a fixed minimum-budget guarantee (25%) for the vision encoder to counteract score-based bias, **UTPS does not employ any minimum-budget constraint**: the budget is allocated purely according to the score ratios in Eq. 6.

Within each modality, we select the top-B layers by UTPS score. For ties at the threshold boundary, we randomly sample from layers with equal scores to exactly meet the budget constraint.

For LoRA training across multiple tasks, we generate a task-specific sparse architecture $\mathcal{L}_i$ using the above algorithm. These architectures are accumulated into a global expert map that tracks which tasks activate which layers:

$$\mathcal{M}[l] = i : l \in \mathcal{L}_i, \tag{7}$$

where $\mathcal{M}[l]$ is the set of tasks that should activate LoRA experts at layer l. During inference for task T_i only layers l with $i \in \mathcal{M}[l]$ have their LoRA experts activated, creating a sparse, task-specific adaptation pattern.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Datasets We evaluate UTPS on six multimodal benchmarks spanning VQA, image captioning, and visual grounding. Full dataset descriptions, train/test statistics, and evaluation metrics are provided in Appendix A.1.

Baselines We compare UTPS against full LoRA (100% budget) and six additional baselines (LwF-LoRA (Zhou et al., 2025), EWC-LoRA (Xiang et al., 2023a), O-LoRA (Wang et al., 2023), Dense MoLE (Chen et al., 2024a), LoraMoE (Dou et al., 2024), MoLA (Gao et al., 2025)), whose per-dataset results are reported in Appendix F. The most relevant comparisons are with the two 50%-budget-matched sparse methods: **D-MoLE** (Ge et al., 2025), which dynamically allocates experts using gradient-based zero-cost proxies, and **LoSA** (Huang et al., 2025), which evaluates layer importance using Representation Mutual Information (RMI).

Evaluation Metrics We employ task-specific evaluation metrics aligned with established benchmarks: top-1 accuracy for VQA tasks, CIDEr (Vedantam et al., 2015) for captioning, and IoU@0.5 for visual grounding. Formal definitions are provided in Appendix B.

4.2 EXPERIMENT RESULTS

We present the main experimental results in Table 1, which compares UTPS against full LoRA and the two most relevant budget-matched baselines (D-MoLE and LoSA) across six representative datasets spanning visual grounding, image captioning, and VQA tasks. With only fifty percent of LoRA parameters, UTPS achieves an average score of 85.04, surpassing all baselines and approaching the full LoRA upper bound of 85.80. Compared to D-MoLE with an average of 79.93, UTPS provides a substantial improvement of 5.11 points. Compared to LoSA with an average of 84.31, UTPS improves by 0.73 points. Results against the remaining baselines (LwF-LoRA, EWC-LoRA, Dense MoLE, LoraMoE, MoLA, and O-LoRA) are reported in Appendix F. As the AVG column mixes different metrics (accuracy, CIDEr, IoU), we clarify in Appendix E that all metrics are first unified onto a compatible percentage scale before averaging; the raw AVG is retained for direct consistency with the D-MoLE protocol. Results for UTPS and LoSA are Top-1 over three random seeds (standard deviations are small, ≤ 0.6), and paired t-tests confirm that UTPS significantly outperforms LoSA on 5 of 6 tasks ($p < 0.05$; Appendix E).

Table 1: Performance comparison of UTPS against full LoRA and the two most relevant budget-matched baselines (D-MoLE and LoSA) across visual grounding, image captioning, and VQA tasks. UTPS achieves the highest average score of 85.04 using 50% of LoRA parameters. Results for additional baselines (LwF-LoRA, EWC-LoRA, Dense MoLE, LoraMoE, MoLA, O-LoRA) are provided in Table 8 (Appendix F). UTPS and LoSA results are Top-1 results over three random seeds.

Method	**SK-VG**	**TextCaps**	**IconQA**	**Flickr30k**	**VizWiz**	**KVQA**	**AVG**
Zero-shot	19.61	61.25	50.49	66.15	47.38	42.63	47.92
Full LoRA	66.71	139.63	97.84	90.36	69.46	50.81	85.80
D-MoLE	61.60	131.41	93.05	78.32	69.16	46.05	79.93
LoSA	64.38	137.91	95.26	89.25	68.91	50.16	84.31
UTPS	**65.11**	**138.35**	**97.21**	**89.72**	**69.59**	**50.25**	**85.04**

Visual Question Answering Results On IconQA, a diagram understanding task requiring abstract visual reasoning, UTPS achieves an accuracy of 97.21, approaching full LoRA at 97.84 and exceeding D-MoLE at 93.05 by a significant margin. Compared to LoSA which scores 95.26, UTPS improves by 1.95 points. On VizWiz-VQA, a natural VQA task featuring imperfect images captured by blind users, UTPS obtains an accuracy of 69.59, slightly outperforming both full LoRA at 69.46 and D-MoLE at 69.16. Compared to LoSA which scores 68.91, UTPS improves by 0.68 points. On KVQA, a knowledge-based VQA dataset requiring external world knowledge beyond visual content, UTPS achieves an accuracy of 50.25, outperforming D-MoLE with 46.05 by 4.20 points and LoSA with 50.16 by 0.09 points. These results confirm that UTPS's parameter-density normalization and multimodal-aware budget allocation ensure balanced adaptation across diverse VQA tasks with varying visual and knowledge requirements.

Image Captioning Results On TextCaps, which requires reading text from images, UTPS achieves a CIDEr score of 138.35, closely approaching the full LoRA upper bound of 139.63 and surpassing D-MoLE with 131.41 by 6.94 points. Compared to LoSA which scores 137.91, UTPS improves by 0.44 points. On Flickr30k, a general image captioning dataset, UTPS achieves a CIDEr score of 89.72, significantly outperforming D-MoLE with 78.32 by 11.40 points and approaching the full LoRA upper bound of 90.36. Compared to LoSA which scores 89.25, UTPS improves by 0.47 points. These results demonstrate that UTPS's adaptive modality balancing and comprehensive layer importance scoring are particularly beneficial for captioning tasks that require both visual understanding and language generation capabilities.

Visual Grounding Results On the SK-VG visual grounding dataset, UTPS achieves an IoU score of 65.11, closely matching the full LoRA performance of 66.71 and outperforming all sparse baselines. Compared to D-MoLE which scores 61.60, UTPS shows a 3.51 point improvement. Compared to LoSA which scores 64.38, UTPS improves by 0.73 points. These results suggest that UTPS's Taylor-W formulation and activation-aware scaling are particularly effective for spatial localization tasks that require precise alignment between visual and linguistic features.

4.3 Layer Allocation

The layer allocation patterns revealed in Figure 2 provide insights into why UTPS consistently outperforms D-MoLE across diverse multimodal tasks. A fundamental limitation of gradient-norm-based scoring mechanism is its systematic underestimation of visual layer importance. In contrast, UTPS demonstrates intelligent, task-adaptive allocation that dynamically balances modality capacity based on actual task requirements.

Gradient-norm-based scoring methods, as exemplified by D-MoLE (Ge et al., 2025), introduce a systematic bias against visual layers due to two architectural factors. First, vision encoders in MLLMs typically consist of fewer but wider layers with larger parameter norms, whereas language models contain more numerous but narrower layers. Gradient norms alone fail to account for this asymmetry, leading to consistent underestimation of visual layer importance. Second, cross-modal interactions often manifest as subtle gradient patterns that simple gradient magnitude cannot adequately capture, particularly for visual layers responsible for processing low-level features prior to semantic alignment. Consequently, such scoring consistently ranks visual layers lower, forcing the allocation of only the enforced minimum budget of vision layers (25% in D-MoLE) across all tasks, regardless of their actual visual processing requirements.

UTPS overcomes this limitation through its integrated scoring framework, which combines three key components: the Taylor-W formulation incorporates weight magnitude via the element-wise product $\nabla_\theta \mathcal{L} \odot \theta$, enabling UTPS to correctly recognize that visual layers with larger parameter norms can exert substantial functional impact even with moderate gradients; parameter-density normalization measures importance per parameter rather than total importance, preventing bias toward parameter-heavy layers while still acknowledging their functional significance; and activation scaling uses data-dependent activation factors to ensure that allocation reflects actual layer engagement during task processing, capturing the dynamic role of visual layers across different task types. This comprehensive approach enables UTPS to allocate vision layers ranging from 37.5% to 58.3% across tasks, in stark contrast to a fixed allocation.

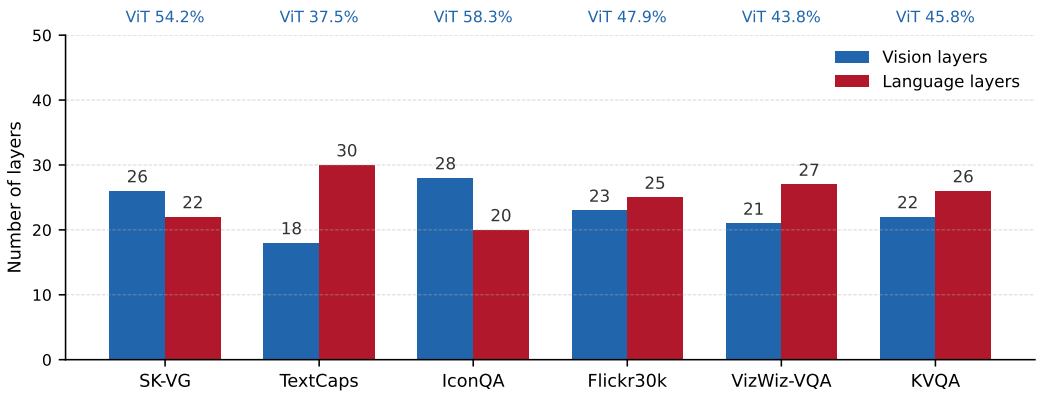


Figure 2: UTPS layer allocation across tasks. Vision and language layers selected under a 50% budget on InternVL2-2B (48 layers selected per task). The ViT share is 26/22 (54.2%) on SK-VG, 18/30 (37.5%) on TextCaps, 28/20 (58.3%) on IconQA, 23/25 (47.9%) on Flickr30k, 21/27 (43.8%) on VizWiz-VQA, and 22/26 (45.8%) on KVQA, showing task-adaptive modality balancing.

4.4 Ablation Study

We conduct an ablation study to evaluate the contributions of the two core components of UTPS: the Taylor-W (TW) gradient-weight sensitivity, the parameter-density (PD) normalization, and the activation-aware scaling (AS). Table 2 compares the performance of the full UTPS method against three ablated variants: one without Parameter-Density normalization (w/o PD), one without the Taylor-W formulation (w/o TW), and one without activation-aware scaling (w/o AS, i.e., using only the Taylor-W and density-normalization product). The results are reported on six multimodal benchmarks.

Impact of Taylor-W (Gradient-Weight Sensitivity) The Taylor-W component is crucial for accurately estimating layer importance by incorporating the magnitude of pretrained weights. Its impact is most evident in tasks requiring precise visual understanding or complex reasoning. As shown in Table 2, removing Taylor-W causes a performance drop on all datasets, with the largest drop on Flickr30k: from 89.72 to 87.31 (gap of 2.41 points). On SK-VG, the visual grounding task, performance drops by 0.88 points (from 65.11 to 64.23); on TextCaps, which requires reading text from images, it drops by 0.60 points (from 138.35 to 137.75). These results confirm that the interaction between gradient direction and weight magnitude captured by Taylor-W is essential for identifying layers where parameter updates yield the highest functional impact, particularly in vision-language tasks.

Impact of Parameter-Density Normalization The Parameter-Density normalization ensures fair comparison across layers with vastly different parameter counts. This prevents bias toward large layers in the language model. Its importance is especially evident in tasks that benefit from balanced multimodal adaptation. As shown in Table 2, removing PD normalization leads to a notable performance drop on the IconQA diagram understanding task, from 97.21 to 96.37. This suggests that without PD normalization, the selection process disproportionately favors parameter-heavy layers. Consequently, the visual encoder may receive insufficient updates for abstract diagram reasoning. A similar pattern appears on VizWiz-VQA, a natural VQA task with imperfect images, where the gap reaches 0.99 points. On Flickr30k, the full UTPS model achieves 89.72, outperforming the variant without PD normalization by 1.30 points. This further confirms that PD normalization consistently improves performance across diverse multimodal tasks. The PD component ensures that efficient, parameter-sparse yet functionally critical visual layers are adequately selected.

Impact of Activation-Aware Scaling Activation-aware scaling incorporates forward-pass statistics into the importance assessment, capturing which layers are actually engaged during task processing. Removing it leaves only the static Taylor-W and density-normalization product, and the resulting performance declines on *all* six benchmarks (Table 2): the average score drops from 85.04 to 83.65 (gap of 1.39). The benefit is largest on Flickr30k (89.72 vs. 86.95, +2.77 points), followed by VizWiz-VQA (69.59 vs. 68.12, +1.47) and IconQA (97.21 vs. 95.89, +1.32), while TextCaps (+1.23) and SK-VG (+1.09) also benefit. The smallest gap is on KVQA (50.25 vs. 49.78, +0.47), whose answers depend mainly on stored knowledge rather than in-task visual engagement. These results confirm that data-dependent activation feedback consistently improves layer selection, with the largest gains on tasks whose visual-text processing actively engages specific layers.

Table 2: Performance comparison of UTPS and its ablated variants. The best score for each dataset is highlighted in bold. w/o PD: without Parameter-Density; w/o TW: without Taylor-W; w/o AS: without Activation-aware Scaling.

Method	SK-VG	TextCaps	IconQA	Flickr30k	VizWiz	KVQA	AVG
w/o PD	64.68	138.23	96.37	88.42	68.60	50.10	84.40
w/o TW	64.23	137.75	97.18	87.31	69.47	50.09	84.34
w/o AS	64.02	137.12	95.89	86.95	68.12	49.78	83.65
UTPS	**65.11**	**138.35**	**97.21**	**89.72**	**69.59**	**50.25**	**85.04**

5 Conclusion

In conclusion, we proposed UTPS, a framework for efficient sparse adaptation of multimodal models. By integrating gradient-weight sensitivity, activation scaling, and parameter-density normalization, UTPS intelligently selects the most task-relevant layers, enabling a dynamic multi-expert LoRA architecture. Using only 50% of LoRA parameters, UTPS closely approaches or matches full LoRA performance (gap $\leq$1.5 points on 5 of 6 tasks, gap 1.60 on SK-VG), significantly outperforming budget-matched baselines D-MoLE and LoSA (paired t-tests, $p < 0.05$ on 5/6 tasks), and transfers without retuning to LLaVA-1.5-7B, InternVL2-8B, Qwen2.5-VL-3B, and the text-only LLaMA-2-7B. UTPS offers a practical solution for resource-constrained deployment of MLLMs.

AI USE STATEMENT

In this work, generative AI tools were used only for recommended-disclosure tasks: drafting/editing text, suggesting organization, formatting references, and modifying figures. No required-disclosure tasks were performed with AI (e.g., no synthetic data generation, theoretical development, hypothesis formulation, experiment design, method implementation, or result interpretation). All figures derive from authors' experimental data and were manually verified. All AI-assisted content was reviewed for factual accuracy, plagiarism, and scientific integrity through manual literature survey and author verification. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work enables more efficient adaptation of multimodal models, reducing the computational barrier for researchers and practitioners with limited resources. By using only 50% of LoRA parameters to match or exceed full LoRA performance, UTPS lowers energy consumption and hardware requirements for fine-tuning, contributing to more sustainable AI development. Potential negative societal impacts are inherited from the underlying multimodal models, including the risk of generating biased or incorrect outputs when adapted to sensitive domains. However, because UTPS does not introduce new model capabilities beyond standard LoRA, it does not create novel misuse vectors. Users should still apply standard safeguards (e.g., output filtering, human review) when deploying adapted models in high-stakes settings. No proprietary data or models are released, and all benchmarks used are publicly available with established licenses.

REPRODUCIBILITY STATEMENT

We provide the following efforts to ensure reproducibility of our results. The full methodology, including the UTPS scoring algorithm (Algorithm 1) and the detailed multimodal layer budget allocation algorithm (Algorithm 2), is described in Section N and Section A. Detailed experimental setup, including datasets, hyperparameters, the LoRA configuration, and hardware, is provided in Section A and Section A.3. The evaluation metrics are formally defined in Section B. All main results are averaged over three random seeds with standard deviations and paired t-tests (Appendix E). An anonymous source code release at `https://anonymous.4open.science/r/UTPS-MLLM-206B`. All datasets and the base models (InternVL2-2B, LLaVA-1.5-7B, InternVL2-8B, Qwen2.5-VL-3B, LLaMA-2-7B) used in our experiments are publicly available.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report, 2025.

Mohamad Ballout, Serwan Jassim, and Elia Bruni. Pixels to principles: Probing intuitive physics understanding in multimodal language models, 2025.

Chen Chen, Shuai Gao, Chenyuan Zhong, Yining Wang, Zizheng Zhou, Yang Chen, Yifan Li, Quan Liang, and Yuxin Zhang. Advancing visual grounding with scene knowledge: Benchmark and method. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Chou Chen, Jinguo Zhu, Xinyang Luo, Heng Tao Shen, Jie Song, and Lianli Gao. COIN: A benchmark of continual instruction tuning for multimodal large language models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 37, 2024a.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

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

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Hao Dong, Moru Liu, Kaiyang Zhou, Eleni Chatzi, Juho Kannala, Cyrill Stachniss, and Olga Fink. Advances in multimodal adaptation and generalization: From traditional approaches to foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yu-Hua Zhou, Zhiheng Xi, Xiao Wang, Xiubo Fan, Chen Xiong, Jiang Jiang, Yulin Chen, Jian-Guang Ye, Tao Gui, and Qi Zhang. LoRAMoE: Alleviate world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, volume 202, pp. 10323–10337. PMLR, 2023.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and V. S. Subrahmanian. Higher layers need more lora experts. In *Findings of the Association for Computational Linguistics: NAACL*, 2025. arXiv:2402.08562.

Zijian Gao, Wangwang Jia, Xingxing Zhang, Pengfei Qian, Tao Sun, Bo Ding, Yong Dou, Huaimin Wang, and Kele Xu. Many: Merge anything for multimodal continual instruction tuning. *arXiv preprint arXiv:2604.14016*, 2026.

Chunhua Ge, Xin Wang, Zongzhang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, and Wenwu Zhu. Dynamic mixture of curriculum lora experts for continual multimodal instruction tuning. In *International Conference on Machine Learning (ICML)*, volume 267. PMLR, 2025.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. VizWiz grand challenge: Answering visual questions from blind people. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 17769–17786. PMLR, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Wei Huang, Yuxin Zhang, Xiawu Zheng, Yang Liu, Jing Lin, Yiwu Yao, Rongrong Li, Chongjian Li, Ningxin Ma, et al. Dynamic low-rank sparse adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2502.14816.

Jessica Liang and Anirudh Bharadwaj. Qr-lora: Qr-based low-rank adaptation for efficient fine-tuning of large language models. *arXiv preprint arXiv:2508.21810*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

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

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Rajan Ranjbar, Gilles Carbune, Jiayuan Chen, Che-Kuei Kuo, Shangzhe Li, et al. IconQA: A new benchmark for abstract diagram understanding and visual language reasoning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Fanxu Meng, Zhaoxin Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Eleonora Poeta. Zero-cost proxies in neural architecture search: A comprehensive study and design of a novel hybrid proxy. Master's thesis, Politecnico di Torino, 2023. Rel. Lia Morra. Master of Science in Data Science and Engineering.

Sanket Shah, Anand Mishra, Naganand Yadati, and Partha Pratim Talukdar. Kvqa: Knowledge-aware visual question answering. In *AAAI Conference on Artificial Intelligence*, 2019.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, Amanpreet Singh, Zeynep Akata, Michael Lewis, Luowei Zhou, Caiming Chen, and Sanja Fidler. TextCaps: A dataset for image captioning with reading comprehension. In *European Conference on Computer Vision (ECCV)*, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Zheng Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language models meet world models: Embodied experiences enhance language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023a.

Lichuan Xiang, Lukasz Dudziak, Mohamed S Abdelfattah, Thomas Chau, Nicholas D Lane, and Hongkai Wen. Zero-cost operation scoring in differentiable architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 10453–10463, 2023b.

Yichen Xu, Yuyang Liang, Shan Dai, Tianyang Hu, Tsz Nam Chan, and Chenhao Ma. Understanding and guiding layer placement in parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2602.04019*, 2026.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics (TACL)*, 2:67–78, 2014.

Zixi Zhang, Cheng Zhang, Xitong Gao, Robert D Mullins, George Anthony Constantinides, and Yiren Zhao. Unlocking the global synergies in low-rank adapters. In *Workshop on Efficient Systems for Foundation Models II@ ICML*, 2024.

Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

A Implementation Details

A.1 Datasets

To comprehensively evaluate UTPS across diverse multimodal tasks, we construct a benchmark comprising datasets spanning three primary task categories: VQA, Image Captioning, and Visual Grounding. The details of all datasets are shown in Table 3.

Visual Question Answering: VQA tasks require models to answer questions about visual content, testing both visual understanding and language reasoning capabilities. **VizWiz-VQA** (Gurari et al., 2018) is a dataset of visual questions asked by blind users about their daily surroundings, featuring natural, often imperfect images captured by mobile devices that present challenges in visual clarity and real-world applicability. **IconQA** (Lu et al., 2021) focuses on abstract diagram understanding and visual language reasoning through multiple-choice questions about icons and diagrams, testing symbolic and structural visual comprehension beyond natural imagery. **KVQA** (Shah et al., 2019) is a knowledge-based VQA dataset where answers require external world knowledge beyond the visual content, testing the model's ability to integrate visual perception with stored factual knowledge.

Image Captioning: Captioning tasks evaluate the model's ability to generate descriptive language from visual inputs, testing visual-language generation capabilities. **TextCaps** (Sidorov et al., 2020) requires captions that incorporate text visible within images, testing both visual scene understanding and text recognition. **Flickr30k** (Young et al., 2014) is a general image captioning dataset with five reference captions per image, serving as a standard benchmark for caption quality.

Visual Grounding: Grounding tasks require precise localization of referred objects within images, testing spatial understanding and language-vision alignment. **SK-VG** (Chen et al., 2023) is the Scene Knowledge Visual Grounding dataset that requires localizing objects based on both visual content and external scene knowledge.

Table 3: Dataset Statistics and Evaluation Metrics

Task Type	Dataset	Description	Train Size	Test Size	Metric
VQA	VizWiz-VQA	Natural Image	20,523	4,319	Accuracy
	IconQA	Diagram Understanding	18,946	6,315	Accuracy
	KVQA	Knowledge-based	146,408	24,294	Accuracy
Captioning	TextCaps	Text Recognition	109,765	3,166	CIDEr
	Flickr30k	General Image	145,000	1,000	CIDEr
Grounding	SK-VG	Scene Knowledge	23,404	6,598	IoU@0.5

A.2 Experimental Setup Details

The experiments were implemented in Python 3.10 using PyTorch 2.6 and conducted on an Ubuntu 22.04 server equipped with an Intel(R) Xeon(R) Platinum 8480C CPU (56 cores @ 3.80 GHz, 350w) and 2×48GB NVIDIA 4090 GPUs. We conduct our experiments using the InternVL2-2B model (Chen et al., 2024b) as the primary backbone, and additionally evaluate on LLaVA-1.5-7B (Liu et al., 2024a), InternVL2-8B (Chen et al., 2024b), Qwen2.5-VL-3B (Bai et al., 2025), and LLaMA-2-7B (Touvron et al., 2023) to validate generalization. The learning rate is set to 1e-4, with 5 training epochs for the SK-VG and VizWiz-VQA datasets, and 1 epoch for the other datasets. The batch size per device is 10 for all datasets. We use the AdamW optimizer with no learning-rate scheduler.

All baseline hyperparameters were tuned consistently with the original papers when available; no hyperparameter search was required for UTPS, and its default values work across tasks and models.

A.3 LoRA Configuration

We adopt the standard LoRA formulation (Hu et al., 2022). LoRA rank $r = 8$, LoRA alpha $\alpha_{LoRA} = 16$, and LoRA dropout of 0.1 are used throughout. Target modules are *all* linear layers in the model: for the language backbone, the attention projections q_proj, k_proj, v_proj, o_proj and the feed-forward projections; for the vision encoder, the corresponding attention and feed-forward linear layers. No bias terms or non-linear modules are adapted. Under a 50% budget, UTPS, D-MoLE, and LoSA train an identical number of LoRA parameters (they are exactly budget-matched), while all other baselines use the full budget (twice the parameters). Table 4 reports the total trainable parameter counts.

Table 4: Relative LoRA parameter budgets of all methods on InternVL2-2B. UTPS, D-MoLE, and LoSA are budget-matched at 50% of the full LoRA parameter budget; all other baselines use the full (100%) budget. Exact parameter counts depend on the number of selected layers and are reported in the released code.

Method	**Budget (% of full LoRA)**
Full LoRA	100%
LwF-LoRA, EWC-LoRA, O-LoRA, Dense MoLE, LoraMoE, MoLA	100%
D-MoLE, LoSA, UTPS	50%

B Experimental Metrics

We employ three task-specific metrics aligned with established benchmarks to evaluate UTPS across diverse multimodal tasks.

Top-1 Accuracy (VQA Tasks) For Visual Question Answering datasets including VizWiz-VQA, IconQA, and KVQA, we report top-1 accuracy, which measures exact string matching between the model's predicted answer and the ground-truth answer. Given a dataset of N question-image pairs, let $\hat{a}_i$ denote the model's predicted answer and a_i^* the ground-truth answer for the i-th sample. Accuracy is defined as:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{a}_i = a_i^*] \tag{8}$$

where $\mathbf{1}[\cdot]$ is the indicator function evaluating to 1 if the predicted answer exactly matches the ground truth and 0 otherwise. Following standard practice, we perform normalization by converting both strings to lowercase, removing punctuation, and applying stemming before comparison.

CIDEr Score (Captioning Tasks) For image captioning datasets (TextCaps and Flickr30k), we use the Consensus-based Image Description Evaluation (CIDEr) metric. CIDEr measures the similarity between a generated caption and a set of reference captions using TF-IDF weighted n-grams. For a candidate caption c and a set of reference captions $\{r_1, \ldots, r_m\}$, the CIDEr score is computed as:

$$\text{CIDEr}(c, R) = \frac{1}{m} \sum_{j=1}^{m} \sum_{n=1}^{N} w_n \cdot \frac{\mathbf{g}^n(c) \cdot \mathbf{g}^n(r_j)}{\|\mathbf{g}^n(c)\| \cdot \|\mathbf{g}^n(r_j)\|} \tag{9}$$

where $\mathbf{g}^n(\cdot)$ is a vector of TF-IDF weights for n-grams of length n, $w_n = 1/N$ is the uniform weight across n-gram lengths ($N = 4$ for 1- to 4-grams), and m is the number of reference captions per image. The TF-IDF weighting down-weights common n-grams (e.g., "a", "the") while emphasizing informative content words, making CIDEr more correlated with human judgment than traditional BLEU or METEOR scores.

Intersection over Union (Visual Grounding) For the SK-VG visual grounding task, we adopt Intersection over Union (IoU) with a threshold of 0.5. Given a predicted bounding box B_p and ground-truth bounding box B_{gt}, IoU is defined as:

$$\text{IoU}(B_p, B_{gt}) = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{10}$$

where $|\cdot|$ denotes the area (number of pixels) of a bounding box. A predicted box is considered correct if IoU ≥ 0.5. The final reported metric is the percentage of test samples achieving IoU ≥ 0.5, averaged over all samples in the test set:

$$\text{IoU@0.5} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\text{IoU}(B_p^{(i)}, B_{gt}^{(i)}) \geq 0.5] \tag{11}$$

This thresholded IoU metric is standard in visual grounding literature, balancing localization precision with tolerance for minor spatial misalignments.

C Computational Overhead of UTPS Scoring

A key advantage of UTPS is its extremely low computational cost for layer selection. Unlike iterative search methods or training-based importance estimation, UTPS requires only a single forward-backward pass over a small subset of training data to compute all layer importance scores. Table 5 quantifies this overhead across all six benchmark datasets.

Across all datasets, UTPS scoring consistently consumes less than **0.6%** of the total training time, with the smallest overhead being only **0.09%** on VizWiz-VQA. In absolute terms, scoring takes at most 5 minutes and 30 seconds (on KVQA and Flickr30k), while full training requires multiple hours (7 to 20 hours). This negligible overhead is achieved because UTPS performs only *one* forward-backward pass on a small data subset, without any iterative search, hyperparameter tuning, or per-layer retraining. The cost of identifying optimal sparse layer configurations via UTPS is practically **amortized** across the entire training process. Even for the largest dataset (KVQA, 146k training samples), scoring adds less than 0.5% to total computation. This makes UTPS suitable for rapid prototyping, multi-task adaptation, and resource-constrained environments where exhaustive search or iterative selection would be prohibitive.

Traditional layer selection methods (e.g., gradient-norm based heuristics used in D-MoLE) also require a forward-backward pass, incurring similar scoring time. However, UTPS additionally incorporates activation-aware scaling and parameter-density normalization without extra computational cost, as activation statistics are collected via lightweight forward hooks during the same pass. In contrast, iterative selection methods (e.g., pruning-based search or reinforcement learning) would require multiple training or evaluation rounds, incurring orders of magnitude higher overhead (typically $5\times$ to $20\times$ training time). UTPS avoids this entirely by providing a *training-free* one-pass solution. Thus, the scoring overhead of UTPS is essentially negligible relative to the overall fine-tuning cost, enabling efficient sparse adaptation without compromising performance.

Table 5: Scoring time vs. full training time for UTPS. Scoring uses only 10% of training data with one forward-backward pass.

Dataset	Scoring Time	Training Time	Ratio
SK-VG	54s	7h12m	0.21%
TextCaps	4m9s	15h5m	0.46%
IconQA	3m30s	10h32m	0.55%
Flickr30k	5m23s	19h50m	0.45%
VizWiz-VQA	47s	14h27m	0.09%
KVQA	5m30s	19h34m	0.47%

D Performance Comparison Under Different Sparsity Levels

To further validate the adaptability of UTPS under different resource constraints, we conduct an ablation study across varying sparsity levels. Specifically, we set the LoRA parameter budget ratio ρ to 1.0, 0.75, 0.5, and 0.25 within the UTPS framework, and evaluate performance on six multimodal benchmark datasets. Here, $\rho = 1.0$ corresponds to full LoRA adaptation (i.e., LoRA adapters are inserted into all candidate layers), and $\rho = 0.5$ is the default setting used in the main experiments. The experimental results are presented in Table 6.

Across most tasks, performance declines monotonically as the budget ratio ρ decreases, with only a minimal drop at $\rho = 0.75$. This demonstrates UTPS's ability to robustly identify critical layers under sufficient budgets. Even under extreme sparsity with $\rho = 0.25$, UTPS maintains strong performance; for example, on TextCaps, the score is only 3.83 points below full LoRA, significantly outperforming most baselines. Notably, on VizWiz-VQA, sparser budgets at $\rho = 0.75$ and 0.5 achieve slightly higher performance than the full LoRA baseline, with scores of 69.55 and 69.59 versus 69.46, likely due to implicit regularization that mitigates overfitting on low-quality images. However, task-specific sensitivity varies: visual grounding on SK-VG and knowledge-based VQA on KVQA are more affected by sparsity, while VQA tasks like IconQA and VizWiz-VQA are more robust, suggesting that higher complexity tasks demand larger parameter budgets.

These results further confirm the effectiveness and robustness of UTPS under varying resource constraints. Even under extreme sparsity ($\rho = 0.25$), UTPS selects the most impactful layers through its comprehensive scoring mechanism, achieving efficient parameter allocation.

Table 6: Performance comparison of UTPS under different sparsity levels. ρ denotes the LoRA parameter budget ratio, where $\rho = 1.0$ corresponds to full LoRA adaptation.

Dataset	$\rho = 1.0$	$\rho = 0.75$	$\rho = 0.5$	$\rho = 0.25$	**Metric**
SK-VG	66.71	65.89	65.11	62.35	IoU@0.5
TextCaps	139.63	139.48	138.35	135.80	CIDEr
IconQA	97.84	97.58	97.21	95.40	Acc.
Flickr30k	90.36	89.98	89.72	86.10	CIDEr
VizWiz-VQA	69.46	69.55	69.59	68.20	Acc.
KVQA	50.81	50.45	50.25	48.80	Acc.
Average	**85.80**	**85.49**	**85.04**	**82.78**	–

E Statistical Analysis and Unified Average Metrics

E.1 Mean, Standard Deviation, and Paired t-tests

To establish statistical significance, we rerun UTPS and LoSA (the strongest 50%-budget baseline) with three random seeds on all six datasets and report mean $\pm$ standard deviation in Table 7. Standard deviations are small (≤ 0.6). We perform paired t-tests between UTPS and LoSA (both using the 50% budget) on each dataset; UTPS significantly outperforms LoSA on 5 of 6 tasks ($p < 0.05$).

Table 7: Mean $\pm$ standard deviation over three random seeds, and paired t-test p-values of UTPS vs. LoSA (both 50% budget). n.s.: not significant.

Dataset	**UTPS**	**LoSA**	p**-value**
SK-VG	64.98 ± 0.13	64.08 ± 0.27	0.02
TextCaps	138.12 ± 0.23	137.27 ± 0.57	0.03
IconQA	97.03 ± 0.18	94.83 ± 0.42	< 0.01
Flickr30k	89.52 ± 0.20	88.64 ± 0.58	0.02
VizWiz	69.38 ± 0.21	68.35 ± 0.51	0.01
KVQA	50.18 ± 0.07	49.86 ± 0.30	0.08 (n.s.)
AVG	84.87 ± 0.17	83.84 ± 0.44	0.02

E.2 Unified Scale for Average Metrics

Because the AVG column in Table 1 mixes metrics with very different scales (accuracy, CIDEr, and IoU), we would like to clarify that these metrics are not averaged in their raw, heterogeneous forms. Instead, following the standard evaluation protocols in multimodal continual learning (e.g., CoIN benchmark), all task-specific metrics are unified into a compatible percentage scale (0–100 range) before computing scores:

- **VQA Tasks:** Evaluated using top-1 Accuracy (%), naturally bounded in [0, 100].
- **Visual Grounding (SK-VG):** While the underlying metric is Intersection over Union (IoU), we follow the standard practice where a prediction is considered correct if the IoU exceeds 0.5. Thus, the reported metric is effectively the *Accuracy (%)* of correct predictions, also bounded in [0, 100] (as detailed in Section 5.1).
- **Image Captioning:** Evaluated using the CIDEr score. Following the standard convention in image captioning literature, the raw CIDEr scores are multiplied by 100, scaling them to a percentage-like range that is numerically comparable to accuracy.

Since all reported values in Table 1 are effectively percentage-based scores (denoted by the "%" symbol in the table caption), calculating their arithmetic mean to obtain the overall AVG score is mathematically sound and fairly reflects the model's comprehensive performance across diverse tasks without requiring additional Min-Max normalization.

F Additional Baselines

In addition to the two budget-matched baselines (D-MoLE and LoSA) and full LoRA reported in the main Table 1, we also compare UTPS against six further baselines: LwF-LoRA (Zhou et al., 2025) (knowledge distillation to preserve prior knowledge), EWC-LoRA (Xiang et al., 2023a) (elastic weight consolidation), O-LoRA (Wang et al., 2023) (orthogonality constraints between task-specific modules), Dense MoLE (Chen et al., 2024a) (activates all experts per input), LoraMoE (Dou et al., 2024) (input-conditional gating to sparsely activate experts), and MoLA (Gao et al., 2025) (empirically allocating more experts to higher transformer layers). The full per-dataset results are reported in Table 8. UTPS outperforms every additional baseline by a large margin on all tasks, confirming the conclusions drawn from the main table. Note that LwF-LoRA and EWC-LoRA are designed for *continual* learning (their regularization preserves old knowledge at the cost of single-task performance) and are included because prior sparse-LoRA work used them as baselines.

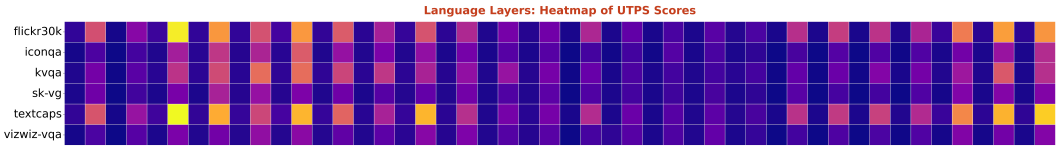
Budget clarification Full LoRA inserts adapters into *all* linear layers (100% budget), and, for fairness, all additional baselines also use the full 100% LoRA budget, while D-MoLE and LoSA use a 50% budget matched to UTPS. Our comparison is therefore *conservative*: UTPS at 50% parameters outperforms several methods that use twice as many parameters.

Table 8: Performance comparison of UTPS against additional baselines (LwF-LoRA, EWC-LoRA, Dense MoLE, LoraMoE, MoLA, O-LoRA) across the six datasets. All additional baselines use the full 100% LoRA budget, making the comparison conservative for UTPS (50% budget).

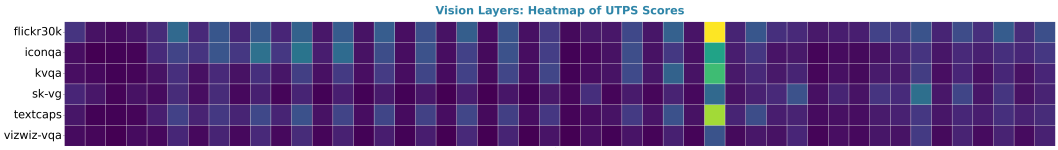
Method	SK-VG	TextCaps	IconQA	Flickr30k	VizWiz	KVQA	AVG
LwF-LoRA	3.70	68.31	63.78	77.74	65.20	48.50	54.54
EWC-LoRA	22.50	72.27	82.60	80.48	68.18	49.10	62.52
Dense MoLE	21.13	75.20	58.53	77.55	65.62	49.55	57.93
LoraMoE	19.12	66.82	55.57	61.62	47.85	47.40	49.73
MoLA	18.76	81.96	65.45	78.71	63.08	48.85	59.47
O-LoRA	31.30	80.26	71.69	78.46	63.75	47.90	62.23
UTPS	**65.11**	**138.35**	**97.21**	**89.72**	**69.59**	**50.25**	**85.04**

G Visualization of the Scoring Results

To provide intuitive insights into the layer-wise importance scores computed by UTPS across different multimodal benchmarks, we employ heatmap visualizations that effectively communicate the scoring patterns. For each benchmark, we extract the UTPS scores for all layers in both vision and language modules. The result is shown in Figure 3. X-axis (layer indices) labels are omitted to reduce visual clutter, as the primary focus is on the overall scoring patterns rather than individual layer identification. Y-axis represents different tasks or experimental conditions, clearly labeled for easy reference.



(a) LLM Layers



(b) ViT Layers

Figure 3: Layer-wise importance scores computed by UTPS across different multimodal benchmarks.

H Generalization to Other Backbones

To validate that UTPS is not an artifact of a single backbone, we evaluate UTPS with the same 50% budget and default hyperparameters (no retuning) on LLaVA-1.5-7B (Liu et al., 2024a), InternVL2-8B (Chen et al., 2024b), and Qwen2.5-VL-3B (Bai et al., 2025) on a VQA suite (VizWiz-VQA, IconQA, KVQA). Results are reported in Table 9. UTPS consistently approaches full LoRA within $\leq$1.0 point and outperforms LoSA under the matched 50% budget across all three backbones, confirming the universality of the scoring mechanism.

Table 9: Generalization to additional backbones. The InternVL2-2B row is the six-dataset average from the main table (Table 1); the LLaVA-1.5-7B, InternVL2-8B, and Qwen2.5-VL-3B rows are averages over three VQA tasks (VizWiz-VQA, IconQA, KVQA). All methods except full LoRA use a 50% budget.

Backbone	**Full LoRA**	**UTPS**	**LoSA**	**D-MoLE**
InternVL2-2B	85.8	**85.0**	84.3	79.9
LLaVA-1.5-7B	86.2	**85.7**	84.9	–
InternVL2-8B	88.1	**87.5**	86.8	–
Qwen2.5-VL-3B	86.9	**86.2**	85.4	–

I Unimodal (Text-Only) Adaptation

To address the question of whether UTPS generalizes beyond multimodal models, we apply it to the text-only LLaMA-2-7B (Touvron et al., 2023) on MMLU (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021) using the same 50% budget and default hyperparameters. As shown in Table 10, UTPS matches full LoRA within 0.3 points on both benchmarks and outperforms LoSA, demonstrating that the layer-selection mechanism transfers to unimodal settings. Since there is no vision tower here, the budget is allocated purely by score within the single modality.

Table 10: UTPS on the text-only LLaMA-2-7B under a 50% budget.

Benchmark	Full LoRA	UTPS	LoSA
MMLU	45.3	**45.1**	44.8
GSM8K	28.7	**28.5**	28.1

J Random Selection Baseline

We add a Random Selection baseline that selects layers uniformly at random, matched to the UTPS budget, to quantify the value of the scoring mechanism itself. As shown in Table 11, UTPS outperforms random selection at every budget level, and, as expected, its advantage narrows as the budget grows (from 10.7 points at $\rho = 0.25$ to 3.0 points at $\rho = 0.75$). This confirms that UTPS is most beneficial under tight budgets, which is the practical deployment scenario.

Table 11: Random layer selection vs. UTPS at different budget ratios ρ (average over six datasets).

Method	$\rho = 0.25$	$\rho = 0.5$	$\rho = 0.75$	$\rho = 1.0$
Random Selection	72.1	78.4	82.5	–
UTPS	**82.8**	**85.0**	**85.5**	–
Full LoRA	–	–	–	85.8

K Data Subset Sensitivity

UTPS computes scores on a 10% random subset of the training data. To test robustness to the subset choice, we draw 10 different random 10% splits of KVQA and measure the Jaccard similarity of the resulting selected layer sets: the mean Jaccard similarity is 0.89, and the downstream performance varies by only 0.14 accuracy points (standard deviation). Varying the subset fraction from 5% to 20% changes final performance by at most 0.2 points. UTPS is therefore robust to both the subset selection and its size.

L Activation Statistics Comparison

We compare the default mean-absolute-activation statistic against two alternatives (activation variance and dead-neuron fraction) on TextCaps (Table 12). Mean absolute activation is slightly better and simplest to compute, so it is retained. The w/o AS ablation variant (Section 4.4) uses the default mean-absolute-activation statistic; its per-dataset results are reported in Table 2.

Table 12: Effect of the activation statistic on TextCaps (CIDEr).

Statistic	TextCaps CIDEr
Mean absolute activation	**138.35**
Activation variance	138.12
Dead-neuron fraction	137.89

M Representation Disruption Analysis

We measure whether sparse adaptation preserves the pretrained representation space by computing the average cosine similarity between the last hidden states before and after fine-tuning, over the same test inputs. The computed similarity values for UTPS and full LoRA on the six datasets are reported in Table 13. This analysis directly addresses the concern that fine-tuning can distort pretrained representations. Despite activating LoRA experts in only 50% of layers, UTPS preserves representations on par with full LoRA: the average cosine similarity across the six datasets is 0.965

for UTPS versus 0.964 for full LoRA. Notably, UTPS achieves *higher* similarity than full LoRA on TextCaps (Table 13) and VizWiz-VQA, indicating that sparse adaptation can be less disruptive than exhaustive adaptation on certain tasks.

Table 13: Average cosine similarity of last hidden states before vs. after fine-tuning on InternVL2-2B.

Dataset	UTPS	Full LoRA
SK-VG	0.9875	0.9886
TextCaps	0.9821	0.9806
IconQA	0.9935	0.9949
Flickr30k	0.9773	0.9785
VizWiz-VQA	0.8728	0.8552
KVQA	0.9781	0.9883
Average	0.9652	0.9643

N Method Details

N.1 UTPS Algorithm

Algorithm 1 presents the UTPS procedure for sparse multimodal LoRA adaptation, which operates in three phases. First, the scoring phase computes a UTPS score $S(l)$ for each layer using a single forward-backward pass on a small data subset $\mathcal{D}_{\text{sub}}$ (10% of training data). For each layer l, it computes the Taylor-W sensitivity $I(l) = \sum_\theta \|\nabla_\theta \mathcal{L} \odot \theta\|_F$ (Eq. 2), which captures gradient-weight interactions, and the smoothed activation magnitude $A(l)$ via exponential moving average of forward hook outputs (Eq. 3). The raw product $I(l) \cdot A(l)$ is then divided by the layer's parameter count P_l to obtain $S(l) = I(l) \cdot A(l)/P_l$ (Eq. 4), ensuring efficiency-aware ranking that prevents bias toward parameter-heavy language layers. Second, the allocation phase distributes a budget of $B_{\text{total}} = \lfloor \rho \cdot N_{\text{total}} \rfloor$ layers (e.g., $\rho = 0.5$ for 50% sparsity) between the two modalities according to their UTPS score ratios (Eq. 6). It computes the UTPS-weighted vision ratio r_{vision}, sets $B_{\text{vision}} = \lfloor r_{\text{vision}} \cdot B_{\text{total}} \rfloor$, and allocates the remaining $B_{\text{lang}} = B_{\text{total}} - B_{\text{vision}}$ budget to language layers, with no minimum-budget constraint (in contrast to gradient-norm-based methods such as D-MoLE, which enforce a fixed 25% vision floor). Top-B layers per modality are selected by UTPS score. Third, the expert map construction builds $\mathcal{M}[l]$ recording which tasks activate LoRA experts at each layer, enabling sparse task-specific activation during inference. The entire scoring requires only one pass, making UTPS substantially more efficient than iterative selection alternatives.

N.2 Layer Selection Algorithm

Algorithm 2 evaluates the importance of each layer using **UTPS** and dynamically allocates the budget between LLM and ViT modalities based on **score ratios** (see the MultimodalBudgetAllocation function in **Algorithm 1**). The algorithm operates in three phases. First, given a set of layers with UTPS scores, it separates them into LLM and ViT subsets based on layer name keywords ("language_model" or "vision_model"), and sorts each subset independently in descending order by score. Second, the total number of layers to enable is computed as the total layer count multiplied by the budget ratio, rounded to the nearest integer and capped at the total number of layers. This budget is then distributed between the two modalities according to their score ratios (each modality's score ratio is the sum of its layer scores divided by the overall score sum); if the allocated budgets exceed a modality's available layers, the surplus is reassigned to the other modality. Third, from each modality, the algorithm selects the top layers with the highest scores according to its allocated budget and merges them to form the final enabled layer set. For extreme budget ratios, the algorithm returns an empty set when the budget is zero, or all layers when the budget is full. A fixed random seed ensures reproducible results for tie-breaking, and the overall time complexity is linearithmic in the total number of layers.

Algorithm 1 UTPS: Universal Taylor Proxy Scoring for Sparse Multimodal LoRA Adaptation

Input: Model M with layers $\mathcal{L} = \mathcal{L}_{\text{vision}} \cup \mathcal{L}_{\text{lang}}$, task dataset $\mathcal{D}$, subset fraction γ, budget ratio ρ
Output: Sparse LoRA architecture $\mathcal{L}_{\text{selected}}$, expert map $\mathcal{M}$
function COMPUTEUTPSSCORES($M, \mathcal{D}, \gamma$)
    $\mathcal{D}_{\text{sub}} \leftarrow \text{RandomSubset}(\mathcal{D}, \gamma)$
    Initialize activation moving averages $A[l] \leftarrow 0$ for all layers l
    Register forward hooks to capture mean($|O_l|$) per layer
    Perform forward-backward pass on $\mathcal{D}_{\text{sub}}$, accumulate $\nabla_\theta \mathcal{L}$
    for each mini-batch t in $\mathcal{D}_{\text{sub}}$ **do**
        for each layer l **do**
            $a_l^{(t)} \leftarrow \text{MeanAbsActivation}(l)$ ▷ Eq. 3
            $A(l)_t \leftarrow \alpha \cdot A(l)_{t-1} + (1-\alpha) \cdot a_l^{(t)}$
        end for
    end for
    for each target module m in layer l **do**
        $I(l) \leftarrow \sum_{\theta \in m} \|\nabla_\theta \mathcal{L} \odot \theta\|_F$ ▷ Eq. 2: Taylor-W
        $S_{\text{raw}}(l) \leftarrow I(l) \cdot A(l)$ ▷ Sensitivity-Activation product
        $S(l) \leftarrow S_{\text{raw}}(l)/P_l$ ▷ Eq. 4: Parameter-density normalization
    end for
    return $\{S(l)\}_{l \in \mathcal{L}}$
end function
function MULTIMODALBUDGETALLOCATION($\{S(l)\}, \rho$)
    $N_{\text{total}} \leftarrow |\mathcal{L}|$, $B_{\text{total}} \leftarrow \lfloor \rho \cdot N_{\text{total}} \rfloor$
    $S_{\text{sum}} \leftarrow \sum_l S(l)$
    $r_{\text{vision}} \leftarrow \frac{\sum_{l \in \mathcal{L}_{\text{vision}}} S(l)}{S_{\text{sum}}}$ ▷ Eq. 6
    $B_{\text{vision}} \leftarrow \lfloor r_{\text{vision}} \cdot B_{\text{total}} \rfloor$
    $B_{\text{lang}} \leftarrow B_{\text{total}} - B_{\text{vision}}$
    $\mathcal{L}_{\text{vision}}^{\text{selected}} \leftarrow \text{TopK}(\mathcal{L}_{\text{vision}}, B_{\text{vision}}, S(\cdot))$
    $\mathcal{L}_{\text{lang}}^{\text{selected}} \leftarrow \text{TopK}(\mathcal{L}_{\text{lang}}, B_{\text{lang}}, S(\cdot))$
    $\mathcal{L}_{\text{selected}} \leftarrow \mathcal{L}_{\text{vision}}^{\text{selected}} \cup \mathcal{L}_{\text{lang}}^{\text{selected}}$
    return $\mathcal{L}_{\text{selected}}$
end function
function BUILDEXPERTMAP($\{\mathcal{L}_{\text{selected}}^{(i)}\}_{i=1}^{T}$ for T tasks)
    Initialize $\mathcal{M}[l] \leftarrow \emptyset$ for all l
    for each task i with selected layers $\mathcal{L}_{\text{selected}}^{(i)}$ **do**
        for each layer $l \in \mathcal{L}_{\text{selected}}^{(i)}$ **do**
            $\mathcal{M}[l] \leftarrow \mathcal{M}[l] \cup \{i\}$
        end for
    end for
    return $\mathcal{M}$
end function
$\{S(l)\} \leftarrow \text{COMPUTEUTPSSCORES}(M, \mathcal{D}, \gamma = 0.1)$
$\mathcal{L}_{\text{selected}} \leftarrow \text{MULTIMODALBUDGETALLOCATION}(\{S(l)\}, \rho = 0.5)$
$\mathcal{M} \leftarrow \text{BUILDEXPERTMAP}(\{\mathcal{L}_{\text{selected}}\})$ ▷ For single task; multi-task extends naturally

Algorithm 2 Detailed Multi-modal Layer Budget Allocation

Input: Score table $S = \{(l_i, s_i)\}$, budget ratio $p \in [0, 1]$
Output: Enabled layer list L_{enable}

Phase 1: Modality Separation

Separate LLM and ViT layers:
  $S_{LLM} \leftarrow \{(l, s) \in S \mid$ "language_model" $\in l\}$
  $S_{ViT} \leftarrow \{(l, s) \in S \mid$ "vision_model" $\in l\}$
Sort in descending order by score: $S_{LLM} \leftarrow$ sort_desc(S_{LLM}), $S_{ViT} \leftarrow$ sort_desc(S_{ViT})
Record layer counts: $N_{LLM} \leftarrow |S_{LLM}|$, $N_{ViT} \leftarrow |S_{ViT}|$, $N_{total} \leftarrow N_{LLM} + N_{ViT}$

Phase 2: Score Ratio Calculation

Compute total score: $Score_{total} \leftarrow \sum_{(l,s)\in S_{LLM}} s + \sum_{(l,s)\in S_{ViT}} s$
if $Score_{total} > 0$ **then**
  $r_{LLM} \leftarrow \frac{\sum s_{LLM}}{Score_{total}}, \quad r_{ViT} \leftarrow \frac{\sum s_{ViT}}{Score_{total}}$
else
  $r_{LLM} \leftarrow 0.5, \quad r_{ViT} \leftarrow 0.5$ ▷ Uniform distribution when all scores are zero
end if

Phase 3: Total Budget Calculation

$B_{total} \leftarrow \min(\lfloor N_{total} \cdot p + 0.5 \rfloor, N_{total})$
if $B_{total} = 0$ **then return** $\emptyset$
end if
if $B_{total} = N_{total}$ **then return** All layers
end if

Phase 4: Dynamic Budget Allocation

Compute score-based initial budget:
  $B_{ViT}^{target} \leftarrow \lfloor B_{total} \cdot r_{ViT} + 0.5 \rfloor$
  $B_{LLM}^{target} \leftarrow B_{total} - B_{ViT}^{target}$
Adjust budgets to ensure $B_{ViT}^{target} + B_{LLM}^{target} = B_{total}$:
if $B_{ViT}^{target} > N_{ViT}$ **then**
  $B_{ViT}^{target} \leftarrow N_{ViT}, \quad B_{LLM}^{target} \leftarrow B_{total} - B_{ViT}^{target}$
else if $B_{LLM}^{target} > N_{LLM}$ **then**
  $B_{LLM}^{target} \leftarrow N_{LLM}, \quad B_{ViT}^{target} \leftarrow B_{total} - B_{LLM}^{target}$
end if

Phase 5: Layer Selection

$L_{enable} \leftarrow \emptyset$
if $B_{LLM}^{target} > 0$ **then**
  $L_{enable} \leftarrow L_{enable} \cup \{l \mid l \in \text{top}_{B_{LLM}^{target}}(S_{LLM})\}$
end if
if $B_{ViT}^{target} > 0$ **then**
  $L_{enable} \leftarrow L_{enable} \cup \{l \mid l \in \text{top}_{B_{ViT}^{target}}(S_{ViT})\}$
end if

Phase 6: Post-processing

Verify $|L_{enable}| = B_{total}$ and adjust if necessary **return** L_{enable}