TALE: TRAINING-AWARE LOW-RANK EMULATION FOR ON-DEVICE ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning foundation models on edge devices remains challenging because training often requires 3–4× more memory than inference. Emulator-based approaches address this challenge by using a lightweight surrogate for the full model, enabling adaptation within an inference-level memory budget. Existing emulators are typically constructed to preserve the full model's forward outputs. However, forward fidelity alone does not ensure that the resulting emulator reproduces the training signals needed for fine-tuning. To address this, we propose TALE, a *Training-Aware Low-rank Emulation* for memory-efficient on-device adaptation. Specifically, our theoretical analysis reveals that adapter divergence is fundamentally governed by the gradient discrepancy between the emulator and the full model. Guided by this insight, TALE optimizes a direct gradient-matching objective, extracting the compact weights in closed form via SVD. This principled approach intrinsically aligns with the full model's loss geometry, yielding an emulator that faithfully replicates the training signals essential for adaptation. Across multimodal VQA, LLM personalization, and image generation, TALE consistently outperforms forward-only emulators and achieves performance competitive with full-model fine-tuning. Furthermore, it demonstrates robust generalization capabilities to unseen users and tasks, supporting decoupled server-side generation for on-device use.

1 INTRODUCTION

Foundation models are increasingly migrating from centralized data centers to everyday edge devices Liu et al. (2024); Chu et al. (2024). Mobile phones, vehicles, and household robots are emerging as important platforms for deploying vision and language models locally. The success of these local deployments depends crucially on adapting the pretrained models to specific target tasks Hu et al. (2021), individual users Tan et al. (2024); Ruiz et al. (2023), or unique environments Wang et al. (2021); Niu et al. (2023). This adaptation poses a major challenge as edge devices operate under strict power and memory budgets that server environments do not face; fine-tuning incurs a memory overhead considerably higher than standard inference. To mitigate this overhead, prior work has focused on reducing the auxiliary memory footprint during fine-tuning. These approaches generally fall into two main categories: compressing optimizer states Zhao et al. (2024); Hao et al. (2024); Wang et al. (2026) or minimizing the storage of intermediate activations Chen et al. (2016).

Even with auxiliary memory minimized, frozen base parameters still consume the majority of the memory budget. Accordingly, an equally important line of research focuses on reducing the parameter footprint itself. One approach is to compress the original model weights via low-bit quantization, as demonstrated by QLoRA Dettmers et al. (2023). Alternatively, methods like Offsite-Tuning Xiao et al. (2023) and LoRAM Zhang et al. (2025) create a smaller proxy model via layer-dropping or pruning to serve as a surrogate during training. However, a shared limitation of these methods is their inherent reliance on full-model computations during the initial construction phase. To circumvent this bottleneck, EMLoC Lin et al. (2026) constructs an activation-aware SVD emulator, a low-rank surrogate on which standard LoRA fine-tuning can be performed with substantially reduced parameter memory. After fine-tuning, EMLoC performs LoRA correction to align the adapter with the full model before merging the corrected LoRA weights into it for deployment. This procedure enables practical fine-tuning within the standard inference memory budget.

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 effectiveness of emulator-based fine-tuning hinges on an implicit assumption: an emulator that matches the full model's forward pass will also faithfully reproduce its backward pass. Driven by this assumption, existing methods Xiao et al. (2023); Lin et al. (2026) primarily minimize output discrepancies on a calibration set. This objective is intuitively appealing, since closely matching the full model's outputs may seem sufficient to preserve the gradients that govern adapter updates. Questioning the validity of this premise, we empirically test this assumption through a layer-wise analysis of output and gradient alignment, where each alignment is the cosine similarity between the corresponding full-model and emulator quantities. To mitigate the confounding effects of depth-dependent trends, we center both metrics within each transformer block. In this analysis, EMLoC's output alignment shows no meaningful correlation with its gradient alignment ($r = 0.072$, $p = 0.44$). This implies that a layer that exhibits high output fidelity does not necessarily yield high gradient fidelity. These findings demonstrate that optimizing an emulator solely to replicate the full model's outputs fails to reliably preserve its layer-wise training signals during adaptation.

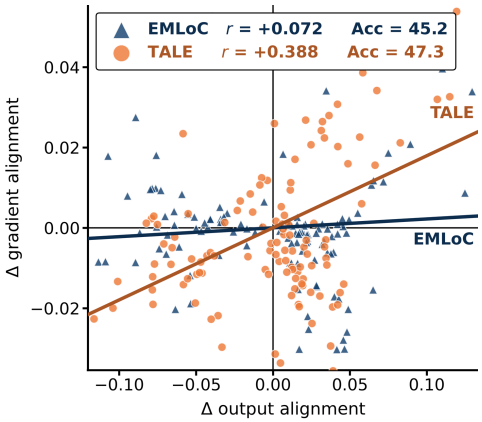


Figure 1: **Correlation between output and gradient alignment.** Each point represents one of the 119 layers of InternVL2.5-2B, averaged over 20 calibration draws. TALE shows a clear positive correlation (r = 0.388, p < 10^{-4}), whereas EMLoC exhibits negligible correlation (r = 0.072, p = 0.44). For reference, legend accuracies reflect the 8B model at 25% compression (2B equivalent).

The inadequacy of forward-only calibration necessitates a fundamental shift in how we construct emulators. This motivates our central research question: Under what conditions can an emulator's backward pass accurately reflect the full model's true training signals? To address this, we introduce the concept of *training-aware* emulation. Our theoretical analysis reveals that the discrepancy between LoRA adapters trained on the full model and those trained on the emulator is fundamentally governed by the accumulated gradient mismatch, ΔG_{LoRA}. Accordingly, we formulate training-aware emulation as finding a low-rank emulator that minimizes this gradient discrepancy. By employing a first-order K-FAC approximation, we translate this intractable objective into one consisting of second moments of forward inputs and output gradients. This reformulation transforms the complex gradient-matching objective into a tractable low-rank approximation problem, which we subsequently solve in closed form via SVD.

Translating these theoretical insights into a practical framework, we present TALE (Training-Aware Low-rank Emulation), an on-device fine-tuning approach that is intrinsically aligned with the full model's loss geometry. A crucial step in this practical realization involves overcoming an inherent challenge: empirically estimating the Fisher metric inevitably introduces numerical instability. To reliably compute the emulator weights, TALE incorporates an objective-based adaptive regularization strategy that optimally stabilizes these empirical gradient statistics. Together, these design choices enable reliable direct transfer of emulator-trained adapters to the full model without post-hoc correction. Our experiments on multimodal VQA, LLM personalization, and personalized image generation demonstrate that TALE consistently improves adaptation performance over forward-only emulators, achieving results highly comparable to full-model fine-tuning. Crucially for scalable deployment, TALE's robustness to distribution shifts allows a single, server-constructed emulator to generalize across unseen users and tasks, eliminating the heavy on-device overhead of data collection and emulator reconstruction.

Our contributions are summarized as follows:

- We identify a fundamental limitation of existing emulator-based fine-tuning: preserving forward outputs does not reliably preserve the layer-wise gradients essential for effective adaptation.
- We introduce TALE, a novel framework that faithfully retains the critical backward signals, effectively addressing the blind spot of forward-only emulator designs.
- We demonstrate both TALE's overall effectiveness and its generalizability to unseen users and tasks through experiments across multimodal VQA, LLM personalization, and image generation.

2 Related Works

Parameter-Efficient Fine-Tuning Parameter-efficient fine-tuning (PEFT) adapts pretrained models by optimizing a small set of additional or selected parameters, such as adapters (Houlsby et al., 2019), bias terms (Zaken et al., 2022), or low-rank updates (Hu et al., 2021). In particular, LoRA (Hu et al., 2021) introduces trainable low-rank matrices while keeping the pretrained weights frozen. Although PEFT substantially reduces training memory by minimizing trainable parameters and optimizer states, it still requires retaining the frozen backbone for forward and backward computation, leaving model weights a major source of a memory bottleneck during on-device finetuning.

Memory-Efficient Fine-Tuning On-device training requires fine-tuning models within limited memory budgets. To address this constraint, prior work has explored various memory-efficient fine-tuning strategies, ranging from optimizer state compression (Zhao et al., 2024; Hao et al., 2024; Wang et al., 2026) and activation checkpointing (Chen et al., 2016) to CPU offloading (Ren et al., 2021) and weight quantization (Dettmers et al., 2023). However, these methods do not reduce the model size itself, leaving a substantial memory gap between inference and fine-tuning under tight memory budgets.

Emulator-based Fine-Tuning A more direct approach to closing the training–inference memory gap is to replace the full model with a compact emulator during adaptation (Xiao et al., 2023; Zhang et al., 2025; Lin et al., 2026). However, most existing methods construct emulators without considering training signals, relying instead on random layer pruning or matching the full model's output activations at initialization. Recent approaches such as OBD-LLM (Li et al., 2026) and GradOT (Yao et al., 2025) incorporate gradient information into model compression to better preserve the properties of the original full model. Building on this line of work, we introduce the concept of a *training-aware emulator* and theoretically and empirically show that it better preserves the training signals of the full model during adaptation.

3 Method

3.1 Preliminaries

We consider the standard setting in which a pretrained model is adapted to a downstream task through lightweight adapters such as LoRA. Our goal is to construct a compressed model, referred to as an emulator, which preserves the *training signals* of the full model, whereas conventional compression aims to preserve its *outputs*.

Setup. Consider the weight matrix $W_{\text{full}} \in \mathbb{R}^{m\times n}$ of a given layer in the original model and a corresponding adapter BA. To isolate the effect of the backbone weights on the training dynamics, we assume that the full model and the emulator share the same input activations $X \in \mathbb{R}^{n\times N}$ over a calibration batch of N tokens, the same adapter factors A and B at initialization, and the same output-to-loss mapping $\mathcal{L} = \Phi(Y)$.

For the full model, the layer output is given by $Y_{\text{full}} = W_{\text{full}}X + BAX$, and the gradient of the loss with respect to this output is $G_{\text{out}} = \nabla\Phi(Y_{\text{full}}) \in \mathbb{R}^{m\times N}$.To characterize the training signals of the layer, we collect the second moments of the inputs and the output gradients as $R_X := XX^\top \in \mathbb{R}^{n\times n}$ and $R_G := G_{\text{out}}G_{\text{out}}^\top \in \mathbb{R}^{m\times m}$, respectively. Applying eigendecomposition to these statistics, we define their invertible square-root factors S_X and S_G such that:

$$R_X = S_X S_X^\top, \qquad R_G = S_G S_G^\top. \tag{1}$$

To evaluate how the emulator affects adapter training, we analyze the gradient gap between the full model and the emulator. To make this analysis mathematically tractable in the subsequent sections, we adopt the Kronecker-factored approximate curvature (K-FAC) framework, which models the output-space curvature of Φ as block-diagonal across tokens, sharing a common per-token factor R_G.

3.2 Defining a Training-Aware Emulator

In emulator-based fine-tuning, the adapter is trained on the emulator and then transferred to the full model. Thus, the utility of an emulator depends directly on its ability to yield an adapter close

to the one learned by the full model. While conventional compression preserves forward outputs, output fidelity does not directly control the resulting adapter drift. Instead, constructing an effective emulator requires focusing on preserving the backward training signals. To formalize this intuition, we examine the learning dynamics of the adapter. (Hao et al., 2024) show that when a LoRA adapter BA is trained from $B_0 = 0$ with a small learning rate η, the effective update $\Delta U_T := B_T A_T$ is dominated by the accumulated merged gradients passed through the initial projection $A_0^\top A_0$. Since both the full model and the emulator start from the identical initial adapter state, the primary source of adapter drift is the discrepancy in the merged gradient, G_{LoRA}, generated by each backbone at every step.

Proposition 1. *Let the adapters of the full model and the emulator be trained with SGD, and let $\Delta G_{\mathrm{LoRA},t} := G^{\mathrm{full}}_{\mathrm{LoRA},t} - G^{\mathrm{emu}}_{\mathrm{LoRA},t}$ be the gap between their merged gradients at step t, both evaluated at the adapter state of the full model. Let $\mathcal{F}_t := R_{X,t} \otimes R_{G,t}$ be the Kronecker-factored Fisher built from the full-model statistics at step t, and $\|V\|^2_{\mathcal{F}_t^{-1}} := \mathrm{vec}(V)^\top \mathcal{F}_t^{-1} \mathrm{vec}(V)$. Then, for a small learning rate η,*

$$\left\|\Delta U_T^{\mathrm{full}} - \Delta U_T^{\mathrm{emu}}\right\|_{\mathrm{F}} \le \eta\, C_T \sum_{t=1}^{T} \left\|\Delta G_{\mathrm{LoRA},t}\right\|_{\mathcal{F}_t^{-1}} + O(\eta^2), \tag{2}$$

where $C_T := \|A_0^\top A_0\|_2 \max_{t \le T} \|S_{X,t}\|_2 \|S_{G,t}\|_2$.

The proof is given in Appendix A.1.

Proposition 1 establishes a critical upper bound: the terminal divergence between the adapters of the full model and the emulator is controlled by the accumulated per-step gradient gaps. By minimizing this per-step discrepancy, we effectively constrain the overall adapter drift. Crucially, this gap is measured in the dual-Fisher metric ($\mathcal{F}_t^{-1}$), dictating that a faithful emulator must jointly account for both input statistics (R_X) and backward error curvature (R_G). Given that the backbones remain frozen and the adapter evolves slowly in the small-learning-rate regime, these curvature statistics vary primarily with the sampled data. We can therefore reliably approximate this trajectory-wide upper bound using a single objective computed over a representative calibration batch, thereby motivating our emulator construction criterion.

Definition 1. Under a deployment budget for memory-efficient fine-tuning, the feasible set of emulator weights for each layer is $\mathcal{C}_r := \{ W \in \mathbb{R}^{m \times n} : \mathrm{rank}(W) \le r \}$. Let $\Delta G_{\mathrm{LoRA}} := G^{\mathrm{full}}_{\mathrm{LoRA}} - G^{\mathrm{emu}}_{\mathrm{LoRA}}$ be the merged-gradient gap on the calibration batch, and $\mathcal{F} := R_X \otimes R_G$ the Fisher built from the calibration statistics. A *training-aware emulator* is a feasible emulator that minimizes the inverse-Fisher gradient gap:

$$W^\star_{\mathrm{emu}} \in \operatorname*{arg\,min}_{W_{\mathrm{emu}} \in \mathcal{C}_r} \left\|\Delta G_{\mathrm{LoRA}}\right\|^2_{\mathcal{F}^{-1}}. \tag{3}$$

In essence, this definition requires the emulator to deliver nearly the identical training signal as the full model, directly aligning the emulator's construction with the goal of accurate adapter transfer.

3.3 Training-aware Emulator Construction

Having established the theoretical objective in Eq. 3, the next step is to translate this intractable gradient-space problem into a computationally feasible weight-space objective. To this end, we leverage the empirical statistics, S_X and S_G, obtained from a full-model calibration batch, which allows us to bridge the theoretical dual-Fisher metric with a practical weight-space Frobenius norm objective.

Theorem 1. *Let $\Delta W := W_{\mathrm{full}} - W_{\mathrm{emu}}$, the difference between the original model weights and the constructed emulator. Under the linearized Kronecker-factored curvature model, let $\widehat{\Delta G}_{\mathrm{LoRA}}$ denote the resulting first-order approximation to the true LoRA-gradient gap ΔG_{LoRA}. Then:*

$$\left\|\widetilde{\Delta G}_{\mathrm{LoRA}}\right\|_{\mathcal{F}^{-1}} \;=\; \left\| S_G^{\top}\,\Delta W\,S_X \right\|_{\mathrm{F}} \tag{4}$$

The detailed proof is provided in Appendix A.2.

We can now directly find the optimal low-rank approximation within this doubly whitened space. By applying the Eckart–Young–Mirsky theorem Eckart & Young (1936), we obtain the exact closed-form solution for our training-aware emulator.

Corollary 1. *Let $\widetilde{W}_{\mathrm{full}} := S_G^{\top} W_{\mathrm{full}} S_X$ have singular value decomposition $\widetilde{W}_{\mathrm{full}} = U\Sigma V^{\top}$, and write U_r, Σ_r, V_r for its rank-r truncation. Then, under Theorem 1, the training-aware emulator in Definition 1 admits the following closed-form solution:*

$$W^{\star}_{\mathrm{emu}} \;=\; \underset{W_{\mathrm{emu}}\in\mathcal{C}_r}{\arg\min} \left\| S_G^{\top}\big(W_{\mathrm{full}} - W_{\mathrm{emu}}\big)S_X \right\|_{\mathrm{F}}^2 \;=\; S_G^{-\top} U_r \Sigma_r V_r^{\top} S_X^{-1}. \tag{5}$$

A memory-efficient factorization follows by splitting the truncated spectrum symmetrically, thereby balancing the scale of the two factors:

$$W^{\star}_{\mathrm{emu}} = W_u W_v, \qquad W_u = S_G^{-\top} U_r \Sigma_r^{1/2} \in \mathbb{R}^{m\times r}, \qquad W_v = \Sigma_r^{1/2} V_r^{\top} S_X^{-1} \in \mathbb{R}^{r\times n}. \tag{6}$$

This factorization makes the proposed emulator practical for on-device fine-tuning: it replaces the frozen $m \times n$ backbone with two compact matrices, W_v and W_u. As a result, the emulator serves as a memory-efficient surrogate for training, preserving the full model's training signals within stringent on-device memory budgets.

3.4 TALE: Training-Aware Low-rank Emulation

Building upon the theoretical insights from the previous sections, we propose **TALE** (**T**raining-**A**ware **L**ow-rank **E**mulation), a practical framework for memory-efficient fine-tuning. The framework operates in three streamlined stages without requiring iterative optimization.

Statistics Collection and Regularization. We first estimate the second moments of the inputs and output gradients, R_X and R_G, via a single forward and backward pass on a small calibration set. As activations and gradients are immediately accumulated into these matrices and discarded, this step requires neither optimizer states nor large activation buffers. Furthermore, given that R_G is typically ill-conditioned due to the limited calibration data, we apply Tikhonov regularization Kress (1999) by modifying the matrix to $R_G + \lambda I$ prior to eigendecomposition. This effectively stabilizes the inverse whitening factor $S_G^{-\top}$, while λ smoothly interpolates between training-aware and activation-aware compression.

Emulator Construction. Using the regularized whitening matrices S_X and S_G, we project the weights into the doubly whitened space $\widetilde{W}_{\mathrm{full}} = S_G^{\top} W_{\mathrm{full}} S_X$. Computing the rank-$r$ truncated SVD of this matrix yields the compact factors W_u and W_v (Eq. 6). Replacing the dense $m \times n$ layer with the low-rank product $W_u W_v$ reduces the parameter footprint to $r(m + n)$, adhering to the on-device memory budget while preserving the essential loss geometry.

Fine-Tuning and Direct Transfer. During fine-tuning, standard LoRA modules (BA) are attached to the frozen emulator. Since TALE is mathematically formulated to minimize the gradient discrepancy (Theorem 1), the adapter is optimized with training signals nearly identical to those of the full model. Consequently, the learned adapter is intrinsically aligned with the original backbone. This fundamental alignment eliminates the need for the complex post-hoc LoRA correction required by prior work Lin et al. (2026). After fine-tuning, the adapter is directly merged into the full model ($W_{\mathrm{ft}} = W_{\mathrm{full}} + BA$) for inference, incurring zero deployment overhead.

Objective-based Adaptive Regularization A crucial challenge in our practical realization is that empirically estimating the Fisher metric to compute R_G inevitably introduces numerical instability. While standard Tikhonov regularization ($R_G + \lambda I$) (Kress, 1999) can mitigate this, applying a

globally fixed λ is highly suboptimal due to the significant variance in gradient scales and spectral properties across different layers. To reliably compute the emulator weights, we propose an *objective-based, layer-wise adaptive regularization* strategy. Rather than relying on heuristic tuning, we parameterize the regularized factor for each layer l as $S_G + \lambda_l I$ and systematically determine the optimal intensity λ_l by directly minimizing our gradient-matching objective (Eq. 4). By strictly tying λ to our theoretical framework, this approach optimally stabilizes the empirical gradient statistics while preserving the distinct training signals of each layer. Further implementation details are provided in Appendix D.

4 Experiments

4.1 Experimental Setup

Benchmark Datasets We evaluate the downstream fine-tuning performance of LoRA adapters trained with emulators constructed by EMLoC and TALE on seven VQA benchmarks, including four standard datasets (ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), InfoVQA (Mathew et al., 2022), and TextVQA (Singh et al., 2019)) and three domain-specific datasets (PMC-VQA (Zhang et al., 2024), WebSRC (Chen et al., 2021), and WC-VQA (Winata et al., 2025)). We further evaluate TALE in a practical server-side construction setting, where an emulator is built once on the server and subsequently deployed for on-device adaptation. We examine two forms of generalization in this setting. To evaluate *user generalization*, we use LaMP (Salemi et al., 2024) to test whether a server-constructed emulator can support personalization across users with private local data. To evaluate *task generalization*, we conduct task-customization experiments on VQA-Med-2019 (Ben Abacha et al., 2019). We additionally evaluate TALE on personalized text-to-image generation using the subject images from DreamBooth (Ruiz et al., 2023).

Baselines and Backbones We evaluate our method on three types of backbone models. For our main experiments, we follow the experimental setup of EMLoC (Lin et al., 2026) and use InternVL2.5 (Chen et al., 2024) as the VLM backbone. For LLM personalization, we follow the setup of OPPU (Tan et al., 2024) and adopt Llama-2-7B (Touvron et al., 2023) as the base language model. To evaluate the applicability of our approach beyond autoregressive models, we further conduct personalized text-to-image generation experiments with FLUX.1-dev (Labs et al., 2025), a flow-matching-based diffusion model.

Implementation Details. For the main experiments, we construct emulators using 64 calibration samples and fine-tune LoRA adapters with rank 8 for 500 iterations. We use a learning rate of 4×10^{-5} with cosine annealing. We evaluate EMLoC after its LoRA correction step, whereas TALE directly transfers the LoRA adapters trained on the emulator to the full model without post-hoc correction. For fixed regularization, we apply a common coefficient to every layer, using $\lambda \in \{0.0, 1.0\}$, where $\lambda = 0.0$ corresponds to no regularization. For adaptive regularization, we use a backbone-specific candidate set of global λ values shared across all experiments. Implementation details for the additional tasks are provided in Appendix B.

4.2 Main Results

We evaluate TALE with direct fine-tuning of smaller InternVL models and EMLoC across four memory configurations (Table 1). Collectively, TALE's fixed and adaptive variants consistently outperform both the smaller baselines and EMLoC on the shared benchmarks. Specifically, TALE yields gains such as $+2.7$ points on InfoVQA at a 50% memory budget and $+2.9$ points on WebSRC at a 25% budget, bringing emulator-based fine-tuning closer to the full-model upper bound while preserving memory efficiency. To reduce tuning costs, a predefined layer-wise regularization is applied universally across all downstream tasks. While the adaptive variant generally achieves the best overall performance under this shared setting, it does not uniformly surpass the fixed variants. Furthermore, because TALE replaces the frozen backbone with a compact surrogate, it is orthogonal to other efficient-training techniques Chen et al. (2016); Dettmers et al. (2023). This orthogonality is supported by the fact that TALE retains its advantage when combined with QLoRA Dettmers et al. (2023). Importantly, TALE directly transfers the adapter trained on the emulator to the original model,

Table 1: **Performance on seven VQA benchmarks under different memory budgets.** Ours is shown in blue with fixed and adaptive λ. Designed to preserve the training signals of downstream tasks, our method consistently outperforms the output-preserving EMLoC and narrows the gap to full-model LoRA fine-tuning in most cases.

Ratio	Method	λ	ChartQA	DocVQA	InfoVQA	TextVQA	PMC-VQA	WebSRC	WC-VQA
100%	Zero-Shot	-	80.3	84.9	40.7	79.8	45.9	72.3	48.1
	Full Model	-	85.1	85.1	47.2	80.0	49.5	86.8	53.5
50%	InternVL 4B	-	83.6	83.8	43.4	79.8	48.8	**85.7**	46.2
	EMLoC	-	84.0	**85.0**	43.4	79.7	48.6	82.5	48.9
	TALE	0.0	84.6	**85.0**	45.7	**80.0**	48.5	81.8	50.7
	TALE	1.0	**85.3**	**85.0**	45.0	79.6	**49.7**	84.3	50.8
	TALE	Adaptive	85.1	84.9	**46.1**	79.7	49.5	84.3	**51.2**
50% + QLoRA	InternVL 4B	-	83.1	82.4	42.3	78.3	47.2	**84.4**	46.1
	EMLoC	-	83.0	84.3	41.2	79.6	48.8	81.4	48.0
	TALE	0.0	82.0	**84.7**	43.0	**79.9**	46.9	78.0	48.1
	TALE	1.0	83.2	83.9	43.7	79.6	49.0	83.2	**48.8**
	TALE	Adaptive	**83.4**	83.9	**44.3**	79.8	**49.2**	83.0	**48.8**
25%	InternVL 2B	-	79.6	80.4	37.6	77.7	43.3	**77.1**	30.8
	EMLoC	-	81.9	85.2	40.8	79.5	45.2	73.0	47.9
	TALE	0.0	82.0	**85.7**	41.2	**79.7**	**47.3**	73.1	48.2
	TALE	1.0	82.8	85.3	**42.1**	79.6	43.6	75.5	48.1
	TALE	Adaptive	**84.4**	85.5	41.6	79.6	44.5	75.9	**48.4**
25% + QLoRA	InternVL 2B	-	79.0	78.5	36.1	77.1	42.0	**76.1**	30.8
	EMLoC	-	79.4	84.1	40.1	79.5	**45.1**	72.9	46.7
	TALE	0.0	76.0	84.1	40.1	**79.7**	45.0	71.8	46.9
	TALE	1.0	**79.6**	84.2	**40.2**	79.5	**45.1**	73.6	47.2
	TALE	Adaptive	79.3	**84.5**	40.0	79.4	43.6	74.0	**47.3**

whereas EMLoC is evaluated with its post-hoc LoRA correction step. Thus, these improvements are achieved without requiring any additional correction procedures after fine-tuning.

4.3 Server-side Emulator Construction

In practical deployment environments, securing a reliable calibration dataset on the device and reconstructing the emulator for each fine-tuning process can be challenging. To address this, we explore a server-side emulator construction scenario, where a single emulator is built on the server and shared across multiple users and downstream tasks. A primary challenge in this context is that the distribution mismatch between calibration and downstream task data may lead to inaccurate statistical estimates, potentially degrading adaptation performance. To evaluate the generalizability of the emulator under these conditions, we designed two key experimental settings: user generalization and task generalization. Our findings demonstrate that the TALE methodology is effective at producing a generalizable emulator that adapts to unseen users and tasks.

User Personalization We first explore LLM personalization by adopting the experimental framework of OPPU (Tan et al., 2024). This allows us to evaluate whether a single, server-constructed emulator can effectively generalize across diverse users. The adaptation process consists of two stages: initially, a LoRA module is fine-tuned on the server using general data aggregated from multiple users to acquire task-specific knowledge. Subsequently, the updated full model is distributed to individual users, who further fine-tune a new LoRA module on-device using their personal data. For this second stage, we compare two distinct emulator construction paradigms: a *user-side* setting and a *server-side* setting. Under the user-side setting, emulators are created locally based on individual data. In contrast, the server-side setting broadcasts a single, pre-constructed emulator to all users, allowing users to bypass local construction.

As shown in Figure 2, the server-constructed emulator achieves performance comparable to both the user-side setting and the full model fine-tuning. TALE substantially improves Movie Tagging performance over EMLoC. It improves accuracy and F1 by 5.3 and 2.9 points, respectively, in the

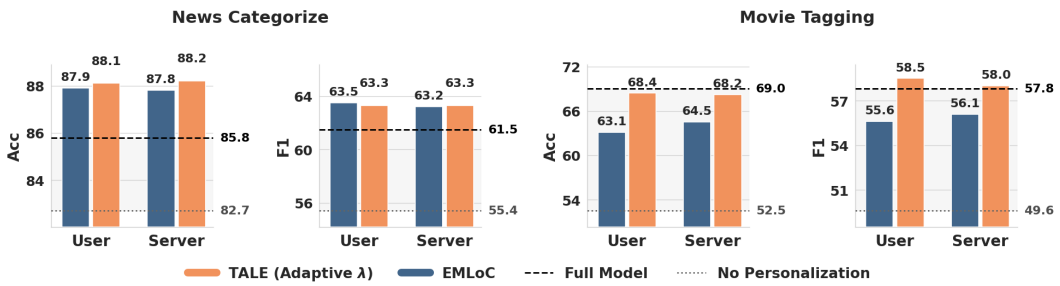


Figure 2: **Robustness in LLM Personalization.** In OPPU setting, TALE consistently outperforms EMLoC under both in-distribution *(User)* and shifted settings *(Server)*, with a larger gap on the challenging Movie Tagging task. This highlights the robustness of training-aware emulation to distribution mismatch.

Table 2: **Robustness in Task Customization** on VQA-Med-2019. TALE remains robust under task distribution mismatch, while EMLoC suffers substantial degradation, particularly in the Modality domain, when constructed with mismatched calibration samples.

Setting	Method	Modality	Plane	Organ	Abnorm.	Avg.
–	Full Model	77.6	77.8	76.4	6.2	59.5
User	EMLoC	72.0	70.8	68.8	1.6	53.3
	TALE (Adaptive λ)	**72.8**	**73.8**	**74.6**	**2.4**	**55.9**
Server	EMLoC	50.4	73.4	69.4	0.2	48.4
	TALE (Adaptive λ)	**72.6**	**74.8**	**71.6**	**2.0**	**55.3**

user-side setting, and by 3.7 and 1.9 points in the server-side setting. These results indicate that TALE transfers effectively to unseen users when the emulator is constructed centrally.

Task Customization We next extend our evaluation to task customization using the VQA-MED-2019 dataset Ben Abacha et al. (2019), which comprises four distinct classification tasks: modality, plane, organ, and abnormality. This setup evaluates the emulator's robustness against distribution shifts between the calibration and downstream data. Similar to the personalization experiment, we compare two emulator construction paradigms. In the *user-side* setting, the calibration set is sampled from the exact same distribution as the downstream fine-tuning data. Conversely, the *server-side* setting simulates a distribution mismatch by constructing the calibration set from the remaining three tasks, excluding the target downstream domain.

As shown in Table 2, EMLoC suffers a substantial performance drop under this distribution mismatch, with its average accuracy decreasing from 53.3 to 48.4. In contrast, TALE maintains highly comparable performance across both paradigms, achieving 55.9 and 55.3 respectively, and notably outperforms EMLoC by 6.9 points in the server-side setting. This demonstrates that TALE enables robust task-specific adaptation even when constructed centrally without access to the downstream task distribution, supporting its applicability to practical server-side deployments.

4.4 Diffusion

We further evaluate TALE on text-to-image personalization to assess whether its training-aware construction extends beyond autoregressive models to diffusion-based personalization. We evaluate personalization with FLUX.1-dev (Labs et al., 2025) on a reduced DreamBooth (Ruiz et al., 2023) benchmark comprising 8 subjects, 10 prompts per subject, and 4 random seeds. For EMLoC and TALE, we construct subject-specific emulators by applying SVD with a rank ratio of 0.45 to the transformer layers, train LoRA on each emulator, and transfer the resulting adapters to the original model for inference. As an upper bound, we also report LoRA fine-tuning results on the uncompressed model. We report DINO and CLIP-I for subject fidelity and CLIP-T for prompt fidelity.

Table 3: Evaluation of TALE on Diffusion Models with DreamBooth.

	DINO	CLIP-I	CLIP-T
Full Model	0.727	0.825	0.299
EMLoC	0.679	0.799	**0.306**
TALE	**0.693**	**0.807**	0.304



Figure 3: Qualitative Result of Generated Images.

Table 4: **Ablation study of Training-Aware Low-Rank Emulation.** We evaluate the contributions of training-aware SVD, LoRA correction, and adaptive regularization on VQA benchmarks.

Components			Datasets		
Training-Aware SVD	LoRA Correction	Adaptive Regularization	ChartQA	TextVQA	WC-VQA
✗	✗	✗	83.8	79.6	48.9
✗	✓	✗	84.0	79.7	48.9
✓	✗	✗	84.6	**80.0**	50.7
✓	✓	✗	84.8	79.7	50.7
✓	✓	✓	84.9	79.5	**51.2**
✓	✗	✓	**85.1**	79.7	**51.2**

Our method outperforms EMLoC on both subject-fidelity metrics, improving DINO from 0.679 to 0.693 and CLIP-I from 0.799 to 0.807, while maintaining comparable prompt fidelity (CLIP-T 0.304 vs. 0.306). This narrows the gap with the full model baseline (DINO 0.727, CLIP-I 0.825) by about 30% on both metrics. The results show that TALE effectively preserves the training signals required for subject-specific generative adaptation.

4.5 Ablation Study

Here, we examine the contributions of TALE's key components: training-aware SVD, LoRA correction, and adaptive regularization. In Table 4, the configuration without training-aware SVD but with LoRA correction corresponds to EMLoC, while TALE uses training-aware SVD and adaptive regularization without LoRA correction. Replacing conventional output-preserving SVD with our training-aware SVD provides the largest performance gain among the components. For example, without correction or regularization, it improves ChartQA from 83.8 to 84.6, indicating that most of the improvement arises from constructing the emulator to preserve training-relevant signals.

TALE remains effective without LoRA correction, outperforming EMLoC even without regularization on all reported benchmarks. Adding LoRA correction to TALE yields only a marginal gain on ChartQA ($+0.2$) and reduces TextVQA performance (-0.3), suggesting that post-hoc correction is unnecessary when the emulator is directly optimized to preserve the full model's training signals rather than inferring training fidelity solely from output alignment. Finally, adaptive regularization produces the best results on ChartQA and WC-VQA, improving the overall average across the three benchmarks.

5 Conclusion

In this paper, we identified a fundamental limitation of existing emulator-based fine-tuning: preserving forward outputs alone fails to reliably retain the essential training signals. To address this, we proposed TALE, a Training-Aware Low-rank Emulation that directly minimizes gradient discrepancy, intrinsically aligning with the full model's loss geometry. Our experiments demonstrate that TALE consistently outperforms forward-only emulators across a wide range of downstream tasks and personalization scenarios, while its robust generalization enables practical server-side emulator construction. Looking ahead, we anticipate that our training-aware emulation paradigm will fundamentally alleviate existing hardware constraints, enabling broader and more practical adaptation of foundation models on edge devices.

AI USE STATEMENT

In this work, we used generative AI tools for finding relevant literature, polishing the writing, and writing the code. We have not used generative AI tools for drafting initial text, generating original research ideas, developing the core methodology. We have reviewed all AI-assisted work. Specifically, we manually verified the existence, relevance, and accuracy of all AI-suggested citations through our own literature survey, and authors thoroughly reviewed and edited all AI-generated text to ensure scientific correctness and clarity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on algorithmic advancements and does not involve human subjects, sensitive or proprietary datasets, or applications that pose immediate societal harm. Therefore, we do not foresee any specific ethical concerns or conflicts of interest arising directly from this research.

REPRODUCIBILITY STATEMENT

We are committed to ensuring that the results presented in this paper are fully reproducible. To achieve this, we have included the complete source code used for our experiments in the supplementary material. Furthermore, a comprehensive description of the experimental setup, including dataset processing steps, model architectures, and hyperparameter configurations, is provided in Appendix B.

REFERENCES

Asma Ben Abacha, Sadid A Hasan, Vivek V Datla, Dina Demner-Fushman, and Henning Müller. Vqa-med: Overview of the medical visual question answering task at imageclef 2019. In *Proceedings of CLEF (Conference and Labs of the Evaluation Forum) 2019 Working Notes*. 9-12 September 2019, 2019.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. Websrc: A dataset for web-based structural reading comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4173–4185, 2021.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, et al. Mobilevlm v2: Faster and stronger baseline for vision language model. *arXiv preprint arXiv:2402.03766*, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *Forty-first International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.03293.

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

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. *arXiv preprint arXiv:2106.09685*, 2021.

Rainer Kress. *Tikhonov Regularization*, pp. 290–307. Springer New York, New York, NY, 1999. ISBN 978-1-4612-0559-3. doi: 10.1007/978-1-4612-0559-3_16. URL https://doi.org/10.1007/978-1-4612-0559-3_16.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Yuhang Li, Donghyun Lee, Ruokai Yin, and Priyadarshini Panda. Optimal brain decomposition for accurate llm low-rank approximation. *arXiv preprint arXiv:2604.00821*, 2026.

Hsi-Che Lin, Yu-Chu Yu, Kai-Po Chang, and Frank Wang. Emloc: Emulator-based memory-efficient fine-tuning with lora correction. *Advances in Neural Information Processing Systems*, 38:166270–166296, 2026.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2199–2208. IEEE, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2582–2591. IEEE, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pp. 551–564, 2021.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510. IEEE, 2023.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8309–8318. IEEE, 2019.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6476–6491, 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. *URL https://arxiv. org/abs/2307.09288*, 2307, 2023.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Taming momentum: Rethinking optimizer states through low-rank approximation. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Genta Indra Winata, Frederikus Hudi, Patrick Amadeus Irawan, David Anugraha, Rifki Afina Putri, Wang Yutong, Adam Nohejl, Ubaidillah Ariq Prathama, Nedjma Ousidhoum, Afifa Amriani, et al. Worldcuisines: A massive-scale benchmark for multilingual and multicultural visual question answering on global cuisines. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3242–3264, 2025.

Guangxuan Xiao, Ji Lin, and Song Han. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*, 2023.

Kai Yao, Zhaorui Tan, Penglei Gao, Lichun Li, Kaixin Wu, Yinggui Wang, Yuan Zhao, Yixin Ji, Jianke Zhu, and Wei Wang. GradOT: Training-free gradient-preserving offsite-tuning for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5115–5130, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.255. URL https://aclanthology.org/2025.acl-long.255/.

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

Jun Zhang, Jue WANG, Huan Li, Lidan Shou, Ke Chen, Yang You, Guiming Xie, Xuejian Gong, and Kunlong Zhou. Train small, infer large: Memory-efficient loRA training for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=s7DkcgpRxL.

Xiaoman Zhang, Chaoyi Wu, Ziheng Zhao, Weixiong Lin, Ya Zhang, Yanfeng Wang, and Weidi Xie. Pmc-vqa: Visual instruction tuning for medical visual question answering, 2024. *URL https://arxiv.org/abs/2305.10415*, 40, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Appendix

A Additional Lemma and Proof

We assume that the compression errors of all other layers are treated as zero, when analyzing a given layer following Wang et al. (2025). Hence the emulator is expected to receive the same input X, and the same output-to-loss map Φ, so that $\mathcal{L} = \Phi(Y)$ with $G_{\text{out}} = \nabla\Phi(Y)$.

A.1 Proof of Proposition 1

Proposition 1 (restated). *Let the adapters of the full model and the emulator be trained with SGD, and let $\Delta G_{\text{LoRA},t} := G^{\text{full}}_{\text{LoRA},t} - G^{\text{emu}}_{\text{LoRA},t}$ be the gap between their merged gradients at step t, both evaluated at the adapter state of the full model. Let $\mathcal{F}_t := R_{X,t} \otimes R_{G,t}$ be the Kronecker-factored Fisher built from the full-model statistics at step t, and $\|V\|^2_{\mathcal{F}_t^{-1}} := \text{vec}(V)^\top \mathcal{F}_t^{-1} \text{vec}(V)$. Then, for a small learning rate η,*

$$\left\|\Delta U_T^{\text{full}} - \Delta U_T^{\text{emu}}\right\|_{\text{F}} \le \eta\, C_T \sum_{t=1}^{T} \left\|\Delta G_{\text{LoRA},t}\right\|_{\mathcal{F}_t^{-1}} + O(\eta^2), \tag{7}$$

where $C_T := \|A_0^\top A_0\|_2 \max_{t\le T} \|S_{X,t}\|_2 \|S_{G,t}\|_2$.

Proof. Let G_t denote the merged gradient G_{LoRA} at t training step during downstream finetuning, computed on the model's own trajectory. For SGD on (B, A) with $B_0 = 0$, (Hao et al., 2024, Theorem 2.1) show that the iterates keep the form

$$A_t = A_0 + \eta\, A_0 f_A(t), \qquad B_t = \eta\, f_B(t) A_0^\top, \tag{8}$$

where $f_A(t) \in \mathbb{R}^{n\times n}$ and $f_B(t) \in \mathbb{R}^{m\times n}$ are matrix-valued functions of the gradients observed up to step t: $f_B(t)$ accumulates the negative gradients, corrected by the change of A, and $f_A(t)$ tracks the change of A relative to A_0. Starting from $f_A(0) = f_B(0) = 0$, they follow the recursion

$$f_A(t) = f_A(t-1) - \eta\, f_B(t-1)^\top G_t, \qquad f_B(t) = f_B(t-1) - G_t\big(I + \eta f_A(t-1)^\top\big),$$

with the index shifted to start at $t = 1$. For bounded gradients and a fixed horizon T, $f_B(t) = O(1)$, so $f_A(t) = O(\eta)$ and $f_B(T) = -\sum_{t=1}^{T} G_t + O(\eta)$. Therefore

$$\Delta U_T = B_T A_T = \eta f_B(T) A_0^\top A_0\big(I + \eta f_A(T)\big) = -\eta \sum_{t=1}^{T} G_t A_0^\top A_0 + O(\eta^2). \tag{9}$$

The full model and the emulator share $B_0 = 0$, A_0, and the minibatch sequence. Applying Eq. 9 to both and subtracting,

$$\Delta U_T^{\text{full}} - \Delta U_T^{\text{emu}} = -\eta \sum_{t=1}^{T} \big(G_t^{\text{full}} - \tilde{G}_t^{\text{emu}}\big) A_0^\top A_0 + O(\eta^2), \tag{10}$$

where $\tilde{G}_t^{\text{emu}}$ is the emulator gradient on its own trajectory. By Eq. 8, both trajectories stay within $O(\eta)$ of the shared initialization. Hence, for an emulator gradient that is locally Lipschitz in the adapter parameters, evaluating it on the full-model trajectory instead changes each summand by $O(\eta)$. This replaces $G_t^{\text{full}} - \tilde{G}_t^{\text{emu}}$ by $\Delta G_{\text{LoRA},t}$ at the cost of $O(\eta^2)$, and taking the Frobenius norm gives

$$\left\|\Delta U_T^{\text{full}} - \Delta U_T^{\text{emu}}\right\|_{\text{F}} \leq \eta \left\|A_0^\top A_0\right\|_2 \sum_{t=1}^{T} \left\|\Delta G_{\text{LoRA},t}\right\|_{\text{F}} + O(\eta^2). \tag{11}$$

With the ridge terms, $\mathcal{F}_t = R_{X,t} \otimes R_{G,t} \succ 0$. By the Rayleigh quotient, $\|V\|_{\text{F}}^2 \leq \lambda_{\max}(\mathcal{F}_t) \|V\|_{\mathcal{F}_t^{-1}}^2$ for any V. Since the eigenvalues of a Kronecker product are the pairwise products of those of its factors,

$$\lambda_{\max}(\mathcal{F}_t) = \lambda_{\max}(R_{X,t}) \lambda_{\max}(R_{G,t}) = \|S_{X,t}\|_2^2 \|S_{G,t}\|_2^2. \tag{12}$$

Applying this to each summand of Eq. 11 and using $\|A_0^\top A_0\|_2 \|S_{X,t}\|_2 \|S_{G,t}\|_2 \leq C_T$ for all $t \leq T$, we obtain

$$\left\|\Delta U_T^{\text{full}} - \Delta U_T^{\text{emu}}\right\|_{\text{F}} \leq \eta\, C_T \sum_{t=1}^{T} \left\|\Delta G_{\text{LoRA},t}\right\|_{\mathcal{F}_t^{-1}} + O(\eta^2).$$

□

A.2 Main Theorem and Proof

The following observation is what makes the analysis tractable: although LoRA has two factors, their gradients are two fixed linear images of a single matrix, so controlling that matrix controls both.

Lemma 1. *For a layer $Y = WX + BAX$, the gradient with respect to the merged update BA is $G_{\text{LoRA}} = G_{\text{out}} X^\top$, and*

$$\frac{\partial \mathcal{L}}{\partial B} = G_{\text{LoRA}} A^\top, \qquad \frac{\partial \mathcal{L}}{\partial A} = B^\top G_{\text{LoRA}}. \tag{13}$$

Consequently, at a common adapter state (A, B),

$$\Delta\left(\frac{\partial \mathcal{L}}{\partial B}\right) = \Delta G_{\text{LoRA}} A^\top, \qquad \Delta\left(\frac{\partial \mathcal{L}}{\partial A}\right) = B^\top \Delta G_{\text{LoRA}}. \tag{14}$$

Proof. By the chain rule

$$\frac{\partial \mathcal{L}}{\partial (BA)} = G_{\text{out}} X^\top = G_{\text{LoRA}}. \tag{15}$$

Using this merged gradient, the gradient with respect to B is

$$\frac{\partial \mathcal{L}}{\partial B} = G_{\text{out}} (AX)^\top = G_{\text{out}} X^\top A^\top = G_{\text{LoRA}} A^\top. \tag{16}$$

Similarly, the gradient with respect to A is

$$\frac{\partial \mathcal{L}}{\partial A} = B^\top G_{\text{out}} X^\top = B^\top G_{\text{LoRA}}. \tag{17}$$

Because the full model and the emulator share the same initialization of (A, B), subtracting their gradients gives

$$\Delta\left(\frac{\partial \mathcal{L}}{\partial B}\right) = \Delta G_{\text{LoRA}} A^\top, \qquad \Delta\left(\frac{\partial \mathcal{L}}{\partial A}\right) = B^\top \Delta G_{\text{LoRA}}, \tag{18}$$

which proves Eq. 14. □

Theorem 1 (restated). *Let $\Delta W := W_{\text{full}} - W_{\text{emu}}$, the difference between the original model weights and the constructed emulator. Under the linearized Kronecker-factored curvature model, let $\widetilde{\Delta G}_{\text{LoRA}}$ denote the resulting first-order approximation to the true LoRA-gradient gap ΔG_{LoRA}. Then:*

$$\big\|\widetilde{\Delta G}_{\text{LoRA}}\big\|_{\mathcal{F}^{-1}} \;=\; \big\|\, S_G^\top \,\Delta W\, S_X \,\big\|_{\text{F}} \tag{19}$$

Proof. Since both models receive the same input X, share the same initial adapter state (A, B), and use the same output-to-loss map Φ,

$$\Delta Y := Y_{\text{full}} - Y_{\text{emu}} = \Delta W X. \tag{20}$$

Thus,

$$G_{\text{out}}^{\text{emu}} = \nabla\Phi(Y_{\text{emu}}) = \nabla\Phi(Y_{\text{full}} - \Delta Y). \tag{21}$$

Therefore, linearizing $\nabla\Phi$ around Y_{full} gives

$$\operatorname{vec}(\Delta G_{\text{out}}) = \operatorname{vec}\big(\nabla\Phi(Y_{\text{full}}) - \nabla\Phi(Y_{\text{full}} - \Delta Y)\big) \approx \nabla^2\Phi(Y_{\text{full}})\operatorname{vec}(\Delta Y). \tag{22}$$

Neglecting cross-token blocks and assuming a common per-token curvature R_G, we have $\nabla^2\Phi(Y_{\text{full}}) \approx I_N \otimes R_G$. Hence

$$\operatorname{vec}(\Delta G_{\text{out}}) \approx (I_N \otimes R_G)\operatorname{vec}(\Delta Y) = \operatorname{vec}(R_G \Delta Y), \quad \text{i.e.,} \quad \Delta G_{\text{out}} \approx R_G \Delta W X. \tag{23}$$

By Lemma 1, $\Delta G_{\text{LoRA}} = \Delta G_{\text{out}} X^\top$, so the first-order approximation is

$$\Delta G_{\text{LoRA}} := R_G \Delta W X X^\top = R_G \Delta W R_X. \tag{24}$$

With the layer Jacobian $J = \partial \operatorname{vec}(Y)/\partial \operatorname{vec}(W) = X^\top \otimes I_m$, the Fisher is

$$\mathcal{F} = J^\top (I_N \otimes R_G) J = (X \otimes I_m)(I_N \otimes R_G)(X^\top \otimes I_m) = R_X \otimes R_G, \tag{25}$$

which recovers the K-FAC form. Using $\mathcal{F}^{-1} = R_X^{-1} \otimes R_G^{-1}$ and the identity $\operatorname{vec}(Z)^\top (P \otimes Q)\operatorname{vec}(Z) = \operatorname{Tr}(Z^\top Q Z P)$ for symmetric P and Q, substituting Eq. 24 yields

$$\begin{aligned}
\|\Delta G_{\text{LoRA}}\|_{\mathcal{F}^{-1}}^2 &= \operatorname{Tr}\big(\Delta G_{\text{LoRA}}^\top \, R_G^{-1} \, \Delta G_{\text{LoRA}} \, R_X^{-1}\big) \\
&= \operatorname{Tr}\Big(\big(R_G \Delta W R_X\big)^\top \, R_G^{-1} \, \big(R_G \Delta W R_X\big) \, R_X^{-1}\Big) \\
&= \operatorname{Tr}\Big(R_X \, \Delta W^\top \underbrace{R_G \cdot R_G^{-1} \cdot R_G}_{=R_G} \, \Delta W \, \underbrace{R_X \, R_X^{-1}}_{=I}\Big) \\
&= \operatorname{Tr}\big(\Delta W^\top R_G \, \Delta W \, R_X\big)\,.
\end{aligned} \tag{26}$$

Substituting $R_G = S_G S_G^\top$ and $R_X = S_X S_X^\top$ (e.g., $S = Q\Lambda^{1/2}$ from the eigendecomposition) and using the cyclicity of the trace,

$$\begin{aligned}
\operatorname{Tr}\big(\Delta W^\top R_G \, \Delta W \, R_X\big) &= \operatorname{Tr}\big(\Delta W^\top S_G S_G^\top \, \Delta W \, S_X S_X^\top\big) \\
&= \operatorname{Tr}\Big(\big(S_G^\top \Delta W S_X\big)^\top \big(S_G^\top \Delta W S_X\big)\Big) \;=\; \big\|S_G^\top \, \Delta W \, S_X\big\|_{\text{F}}^2\,.
\end{aligned} \tag{27}$$

Taking square roots of both nonnegative sides yields Eq. 19. □

A.3 Proof of Corollary 1

Proof. We assume $R_X, R_G \succ 0$, so that S_X and S_G are invertible. By Theorem 1, and since squaring preserves minimizers, the training-aware emulator solves

$$\min_{W_{\text{emu}} \in \mathcal{C}_r} \left\| S_G^\top (W_{\text{full}} - W_{\text{emu}}) S_X \right\|_F^2. \tag{28}$$

Consider the change of variables $Z := S_G^\top W_{\text{emu}} S_X$, which gives

$$\left\| S_G^\top (W_{\text{full}} - W_{\text{emu}}) S_X \right\|_F^2 = \left\| \widetilde{W}_{\text{full}} - Z \right\|_F^2. \tag{29}$$

Since multiplication by invertible matrices preserves rank, $\operatorname{rank}(Z) = \operatorname{rank}(W_{\text{emu}})$. Hence $W_{\text{emu}} \mapsto Z$ is a bijection from $\mathcal{C}_r$ onto itself, and the problem is equivalent to

$$\min_{Z \in \mathcal{C}_r} \left\| \widetilde{W}_{\text{full}} - Z \right\|_F^2. \tag{30}$$

By the Eckart–Young–Mirsky theorem, this is minimized by $Z^\star = U_r \Sigma_r V_r^\top$, attaining the value $\sum_{i>r} \sigma_i^2$. The minimizer is unique if $\sigma_r > \sigma_{r+1}$. Inverting the change of variables yields

$$W_{\text{emu}}^\star = S_G^{-\top} Z^\star S_X^{-1} = S_G^{-\top} U_r \Sigma_r V_r^\top S_X^{-1}. \tag{31}$$

Finally, writing $\Sigma_r = \Sigma_r^{1/2} \Sigma_r^{1/2}$ gives

$$W_{\text{emu}}^\star = \left(S_G^{-\top} U_r \Sigma_r^{1/2} \right) \left(\Sigma_r^{1/2} V_r^\top S_X^{-1} \right) = W_u W_v, \tag{32}$$

with $W_u \in \mathbb{R}^{m \times r}$ and $W_v \in \mathbb{R}^{r \times n}$. □

B Experimental Setup

Common setup. For all vision–language experiments we use InternVL2.5-8B as the original model. Unless stated otherwise, the emulator is built by low-rank SVD compression of every linear layer in the vision encoder and the language model, with a rank ratio of 0.4 (8B→4B). For the more aggressive 8B→2B setting the ratio is 0.2. The MLP projector and the down-projection of the last LLM layer are left uncompressed. Calibration statistics are computed from 64 samples. For EMLoC the emulator uses activation-aware (output-preserving) SVD. Ours uses the combined two-sided SVD, which whitens both the input activations (R_X) and the output gradients (R_G). Damping is applied only on the gradient side, with R_X undamped. LoRA (r=8, α=8) is attached to all attention and FFN projections of the LLM. All experiments run on a single NVIDIA RTX 3090 (24 GB) or RTX A6000 (48 GB). Please refer to the detailed setups for the emulator construction and downstream finetuning in Table 5.

Benchmarks. For DocVQA and InfoVQA, we train on the official training splits (39,463 and 23,946 QA pairs) and evaluate on the validation splits (5,349 and 2,801). For PMC-VQA we train on 176,948 samples and report multiple-choice accuracy, mapping each generated answer to the most string-similar option. For WebSRC we train on 307,315 samples and report accuracy under a deterministic, ground-truth-aware matching rule. The rule uses normalized exact match, yes/no and numeric/price extraction, word-boundary containment for short answers, and token-level F1 ≥ 0.8 for longer answers. In all benchmarks, calibration samples are drawn from the training split of the target task.

Personalization (OPPU). We build on OPPU with Llama-2-7B and evaluate Movie Tagging (LaMP-2M) and News Categorization (LaMP-2N) without retrieval or profile augmentation (k=0). A task LoRA is first trained and merged into the full backbone, after which each user's personal LoRA is trained on a server-constructed emulator shared across all users. The emulator compresses

Table 5: **Hyperparameters for Training.** Detailed hyperparameter configurations for emulator construction and downstream LoRA fine-tuning on vision-language (InternVL2.5-8B) and diffusion (FLUX.1-dev) models.

	VLM (InternVL2.5-8B)	Diffusion (FLUX.1-dev)
Compressed modules	ViT + LLM linear layers	Transformer blocks
Rank ratio	0.4 (4B) / 0.2 (2B)	0.45
# Calibration samples	64	all instance images ($\leq$16)
LoRA rank / α	8 / 8	4 / 4
Optimizer	AdamW	Prodigy
Learning rate	4×10^{-5} (cosine, 3% warmup)	1.0 (constant)
Weight decay	0.01	–
Batch size (effective)	$2 \times 4 = 8$	$1 \times 4 = 4$
Training steps	500	500
Resolution	448 tiles (max 6 + thumbnail)	512×512
Decoding / sampling	greedy, 500 tokens	50 steps, CFG 3.5

the seven linear layers in every decoder block with a rank ratio of 0.6 and is calibrated using 128 label-balanced examples from non-target users; no target-user data is used.

For TALE, we apply a single adaptive layer-wise regularization table shared across the evaluated tasks, selected from lambda pool $(0, 0.005, 0.01, 0.02, 0.05, 0.1, 0.25, 0.5, 1.0)$ across movie tagging and news categorization tasks. The same calibration data is used for EMLoC. We train the task LoRA for 3 epochs on the full model and each personal LoRA for 2 epochs on the emulator, using AdamW with learning rate 10^{-4}, weight decay 0.01, 10% warmup, and gradient clipping at 0.3. We evaluate the transferred personal LoRA on the full backbone. We use a fixed random subset of 20 users per task and report accuracy and F1 using the official LaMP evaluator.

Task customization (VQA-Med). We treat the four question categories of ImageCLEF VQA-Med-2019 (Modality, Plane, Organ, Abnormality) as separate customization tasks. Each task has about 3,200 training samples (3,192 for Abnormality) and is evaluated on its 500-sample official validation set. To simulate a server that has no access to the user's data, the emulator is calibrated only on the *other three* tasks (e.g., Plane, Organ and Abnormality for the Modality task; 64 samples), while LoRA is trained on the target task's own data. EMLoC uses the same cross-task calibration, and Ours uses fixed damping λ=1.0. We compare against zero-shot InternVL2.5-8B and LoRA fine-tuning of the full 8B model. Following the official protocol, we report strict accuracy (exact token match after lowercasing and removing punctuation) and BLEU. Because most answers are only one or two words long, the BLEU n-gram order is set to $\min(4, |\text{GT}|)$.

Diffusion (DreamBooth). We use FLUX.1-dev and compress only its transformer blocks, with a rank ratio of 0.45. The emulator is calibrated on the subject's instance images under the same flow-matching objective used for fine-tuning. LoRA (r=4, α=4) is applied to the attention and feed-forward layers of the transformer. We train at 512×512 resolution with batch size 1 and 4 gradient-accumulation steps, using the Prodigy optimizer (learning rate 1.0, constant schedule) for 500 steps. Each subject is paired with 10 prompts (5 shared backgrounds, 3 class-specific, and 2 attribute prompts), and each prompt is sampled with 4 seeds, for 320 images in total. Images are sampled with 50 denoising steps and a guidance scale of 3.5 at 512×512. We report DINO (ViT-S/16), CLIP-I and CLIP-T (ViT-B/32); CLIP-T is computed on prompts with the identifier token removed.

C Computational Analysis

Training-aware emulator construction introduces a modest one-time computational cost in exchange for more faithfully reproducing the full model's training signals. Table 6 compares the computational and memory requirements of TALE with full-model LoRA fine-tuning and EMLoC on ChartQA. As TALE collects output-gradient statistics to construct a training-aware emulator, its one-time construction cost is higher than that of EMLoC (16,445 vs. 5,861 TFLOPs). However, both methods use the same compressed emulator during adaptation and therefore incur identical fine-tuning cost: 225 TFLOPs per step, 12.8 seconds per step, and 112,575 TFLOPs over 500 steps. Consequently,

Table 6: **Compute and memory footprint.** We measure resource usage on ChartQA using InternVL2.5-8B and a 25% compression budget for the emulators, on a single NVIDIA RTX 3090 GPU. We report emulator construction and 500-step LoRA fine-tuning. TALE wall-clock time with † is estimates with pinned CPU buffers, since the measured construction time is dominated by CPU–GPU data transfer.

	Emulator Construction		Fine-Tuning				Peak GPU Memory (GB)	Overall	
Method	Total Compute (TFLOPs)	Wall-clock Time (h)	Compute / Step (TFLOPs)	Total Compute (TFLOPs)	Time / Step (sec)	Wall-clock Time (hrs)		Total Compute (TFLOPs)	Wall-clock Time (h)
LoRA	–	–	747	373,395	19.9	2.77	23.4	373,395	2.77
EMLoC	5,861	0.30	225	112,575	12.8	1.78	20.0	118,436	2.08
TALE	16,445	2.20 (0.50†)	225	112,575	12.8	1.78	20.5	129,020	3.98 (2.28†)

TALE requires 129,020 TFLOPs for construction and fine-tuning combined, only an 8.9% increase over EMLoC's 118,436 TFLOPs, while remaining 65.4% lower than full-model LoRA fine-tuning.

The measured TALE construction time is 2.20 hours, but this measurement is dominated by CPU–GPU data transfer during statistics collection rather than GPU computation. With pinned CPU staging buffers, we estimate that construction takes approximately 0.50 hours, resulting in a total wall-clock time of 2.28 hours for construction and 500-step fine-tuning. TALE also maintains a memory footprint comparable to EMLoC, increasing peak GPU memory by only 0.5 GB (20.5 GB versus 20.0 GB). This one-time construction cost is further amortized in our server-side deployment setting, where a single emulator is constructed centrally and shared across many users or downstream tasks. Thus, users perform only the fine-tuning stage on-device and do not incur the construction overhead locally. Overall, TALE achieves substantially lower resource use than full-model fine-tuning while retaining a low memory footprint, enabling effective approximation of full-model training signals within an inference-level budget.

Algorithm 1 Adaptive Selection Strategy for Layer-wise Regularization

Input: Weights $\{W_\ell\}_{\ell=1}^{L}$, target ranks $\{r_\ell\}_{\ell=1}^{L}$, datasets $\mathcal{D}$, candidate pool Λ, fit blocks $\{\mathcal{F}_{d,b}\}_{b=1}^{B}$, and held-out blocks $\{\mathcal{V}_d\}$
Output: Layer-wise regularization table $\{\lambda_\ell^\star\}_{\ell=1}^{L}$
foreach *dataset* $d \in \mathcal{D}$ **do**
  Collect fit statistics from $\{\mathcal{F}_{d,b}\}_{b=1}^{B}$ and validation statistics from $\mathcal{V}_d$ **foreach** *layer* ℓ **do**
    foreach $\lambda \in \Lambda$ **do**
      $\{W_{\ell,\lambda}^{(b)}\}_{b=1}^{B} \leftarrow \mathrm{TrainingAwareSVD}(W_\ell, r_\ell, \lambda, \mathrm{FitStats}_d)$
      $s_{d,\ell}(\lambda) \leftarrow \mathrm{EvaluateObjective}(\{W_{\ell,\lambda}^{(b)}\}_{b=1}^{B}, \mathrm{ValStats}_d)$
      Store $(\lambda, s_{d,\ell}(\lambda))$
    $\bar{s}_{d,\ell}(\lambda) \leftarrow s_{d,\ell}(\lambda) / \min_{\lambda' \in \Lambda} s_{d,\ell}(\lambda')$
foreach *layer* ℓ **do**
  $\lambda_\ell^\star \leftarrow \arg\min_{\lambda \in \Lambda} \sum_{d \in \mathcal{D}} \bar{s}_{d,\ell}(\lambda)$
return $\{\lambda_\ell^\star\}_{\ell=1}^{L}$

D Objective-based Adaptive Regularization

TALE applies an objective-based adaptive regularization strategy independently to each layer to handle the varying stabilization needs across the network. Algorithm 1 selects the optimal coefficient $\lambda_\ell^\star \in \Lambda = \{0, 0.01, 0.1, 0.25, 0.5, 0.75, 1.0\}$ that minimizes the training-aware objective.

We determine these layer-wise coefficients through cross-validation on seven VQA datasets (ChartQA, DocVQA, InfoVQA, TextVQA, PMC-VQA, WebSRC, and WC-VQA). For each candidate coefficient and layer, the emulator is constructed on two disjoint fit batches (64 samples each) and evaluated on a held-out validation batch (128 samples). We normalize each dataset's objective curve by its minimum and aggregate them, selecting the coefficient with the lowest overall score. This yields a single fixed regularization table for the InternVL2.5-8B Chen et al. (2024) emulator, which we seamlessly reuse across all downstream tasks and quantization settings without further tuning.

Table 7: **Layer-wise Regularization Patterns.** Breakdown of the selected adaptive regularization coefficients (λ) by module type and block depth. The distribution highlights the varying stabilization requirements across the vision and language components.

		#	0	0.01	0.1	0.25	0.5	0.75	1.0	Mean
(a) By module type										
Vision	`attn.qkv`	24	0	2	18	4	0	0	0	0.118
	`attn.proj`	24	1	1	2	11	9	0	0	0.311
	`mlp.fc1`	24	0	1	5	9	3	5	1	0.375
	`mlp.fc2`	24	0	1	10	6	7	0	0	0.250
Language	`attn.wqkv`	32	8	14	10	0	0	0	0	0.036
	`attn.wo`	32	0	1	2	3	4	9	13	0.710
	`ffn.w1`	32	0	2	3	2	11	2	12	0.619
	`ffn.w3`	32	1	3	3	6	4	2	13	0.573
	`ffn.w2`	31	0	0	4	7	10	8	2	0.489
(b) By depth (block index)										
Vision	0–5	24	1	4	13	6	0	0	0	0.118
	6–11	24	0	1	12	11	0	0	0	0.165
	12–17	24	0	0	5	9	9	1	0	0.333
	18–23	24	0	0	5	4	10	4	1	0.438
Language	0–7	40	7	1	0	0	1	5	26	0.756
	8–15	40	1	7	0	0	8	10	14	0.639
	16–23	40	0	6	2	9	17	6	0	0.388
	24–31	39	1	6	20	9	3	0	0	0.149

Analysis of layer-wise coefficients. Table 7 shows that the selected coefficients vary systematically across both module types and depth. In the vision encoder, the mean selected λ increases from 0.118 in the shallowest blocks to 0.438 in the deepest blocks; the language model exhibits the opposite trend, decreasing from 0.756 to 0.149. Module-specific patterns are also clear: language `attn.wqkv` layers consistently select little regularization, whereas shallow `attn.wo`, `ffn.w1`, and `ffn.w3` layers favor stronger damping. These structured differences motivate layer-wise rather than globally fixed regularization.

Table 8: **Per-Dataset λ Selection and Generalization.** Histograms of optimal λ values for individual datasets and the transferability metrics (correlation and LOO excess loss) of the general shared table.

	Per-dataset $\lambda^\star$ histogram							Agreement		Excess loss (%)	
Dataset	0	0.01	0.1	0.25	0.5	0.75	1.0	ρ	±1	LOO	Const.
ChartQA	14	41	66	36	44	47	7	0.92	0.95	0.84	5.33
DocVQA	10	29	37	39	23	24	93	0.88	0.85	0.97	5.94
InfoVQA	11	21	45	36	30	19	93	0.85	0.80	0.71	5.44
TextVQA	8	49	44	30	10	18	96	0.64	0.54	1.75	3.12
PMC-VQA	15	24	63	79	24	17	33	0.94	0.97	1.42	5.88
WebSRC	11	19	39	38	21	7	120	0.77	0.74	1.25	7.01
WC-VQA	14	33	84	52	37	18	17	0.88	0.85	1.85	5.41
General	10	25	57	48	48	26	41	–	–	–	–
Mean								0.84	0.81	**1.25**	5.45

Analysis of cross-dataset generalization. Table 8 evaluates the general table on a dataset that was excluded during construction. This leave-one-out evaluation incurs only 1.25% average excess relative loss, compared with 5.45% for the best dataset-specific constant. The general table also remains strongly aligned with the per-dataset selections ($\rho = 0.84$ and 81% agreement within one pool step on average), indicating that the beneficial variation is primarily layer-wise rather than task-specific.