ABM-LoRA: Activation Boundary Matching for Fast Convergence in Low-Rank Adaptation*

Dongha Lee*[1] Jinhee Park*[1,2] Minjun Kim[1] Junseok Kwon†[1]
[1]Chung-Ang University [2]Korea Electronics Technology Institute (KETI)
ia06073@cau.ac.kr iv4084em@cau.ac.kr ekdh635@cau.ac.kr jskwon@cau.ac.kr
*Equal contribution †Corresponding author

Abstract

We propose Activation Boundary Matching for Low-Rank Adaptation (ABM-LoRA), a principled initialization strategy that substantially accelerates the convergence of low-rank adapters. While LoRA offers high parameter efficiency, its random initialization restricts gradient updates to a mismatched tangent space, causing significant information loss and hindering early convergence. Our ABM-LoRA addresses this by aligning the adapter's activation boundaries with those of the pretrained model before downstream training, thereby maximizing the projection of full-parameter gradients into the adapter subspace. This alignment sharply reduces information loss at initialization, yields a lower starting loss, and accelerates convergence. We demonstrate ABM-LoRA's effectiveness across diverse architectures and tasks: language understanding (T5-Base on GLUE), dialogue generation (LLaMA2-7B on WizardLM), and vision recognition (ViT-B/16 on VTAB-1K). On VTAB-1K, it achieves the highest accuracy among all methods, with strong gains on structured reasoning tasks requiring geometric understanding.

1. Introduction

With the advent of large-scale pretrained transformers such as BERT [6], GPT-3 [2], T5 [24], and Vision Transformer (ViT) [7], foundation models have emerged as the dominant paradigm for transfer learning across both language and vision domains. Their ability to capture rich, general-purpose representations allows downstream models to achieve state-of-the-art results on diverse benchmarks, ranging from GLUE [30] to ImageNet [28]. However, fully fine-tuning all parameters of these multi-billion-parameter models remains prohibitively expensive in terms of both memory and computation. To address this, Low-Rank Adaptation (LoRA) [16] freezes the pretrained weights $W_0 \in \mathbb{R}^{d\times k}$ and injects a pair of low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ (with rank r). This reduces the number of trainable parameters from $\mathcal{O}(dk)$ required by full fine-tuning to $\mathcal{O}(r(d+k))$. With libraries like PEFT, LoRA can be easily applied across modalities, from language to vision transformers, and has proven effective on GLUE [30], SQuAD [25], and VTAB-1K [36] benchmarks.

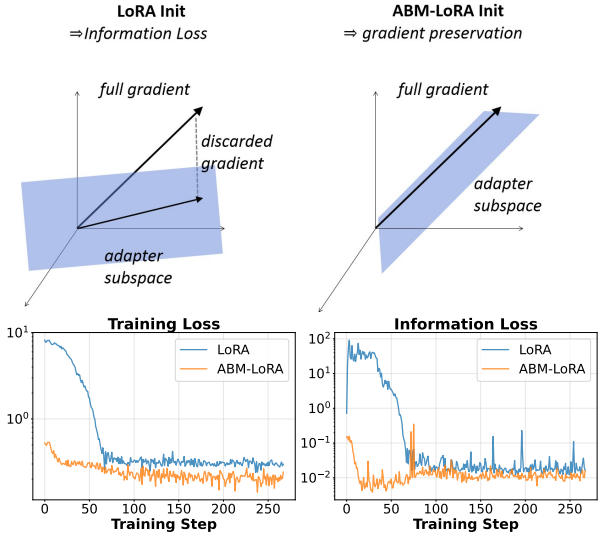


Figure 1. **Top**: Conceptual illustration showing how a poorly aligned adapter subspace (left) discards a significant portion of the gradient signal, whereas our ABM initialization (right) aligns the subspace to fully preserve the gradient direction. **Bottom**: Training and information loss comparison between LoRA and our ABM-LoRA on T5 (GLUE benchmark), highlighting faster convergence and reduced early-stage information loss with ABM-LoRA. Information loss is quantified as the squared Frobenius norm of the components of the full gradient that are discarded when projected onto the adapter's initial tangent space. See Section 3 for theoretical analysis.

Despite its efficiency, LoRA often suffers from slow early convergence: the training loss under LoRA tends to decrease much more gradually than in full-parameter fine-tuning, requiring significantly more steps to reach comparable performance. Prior works (*e.g.* LoRA-GA [32] and other adapter-initialization methods [4, 5, 31]) recognize the importance of initialization and propose various

*arXiv Preprint. [Code]

enhancements, including scale [19] or gradient-matching tweaks [32]. However, these approaches do not sufficiently analyze why random initialization leads to markedly different convergence behaviors. In particular, they overlook how random initialization can immediately discard important gradient components, and how misaligned activation boundaries can further exacerbate gradient loss.

In this paper, we argue that this slow early convergence arises from *initialization-induced information loss*, as shown in the top panel of Fig.A.2. Specifically, a randomly initialized adapter spans only a low-rank subspace, causing any component of the true gradient that lies outside this subspace to be irretrievably discarded at the very first update. The situation becomes more problematic in networks with point-wise nonlinearities (*e.g.* ReLU [1], GeLU [14]). Because LoRA applies a randomly initialized adapter AB before the activation layer, a neuron that would be active in the pretrained model (*i.e.* $W_0x > 0$) may become inactive after applying AB (*i.e.* $(W_0 + AB)x \leq 0$). In ReLU networks, the gradient of this neuron is completely zeroed out. Furthermore, since random initialization shifts activation boundaries differently, poor initialization can dramatically amplify the loss of total gradient information.

To address these issues, we propose **ABM-LoRA**, an Activation Boundary Matching (ABM) initialization for LoRA. Before task-specific fine-tuning, we align the adapter's activations with those of the frozen pretrained model on a representative input batch. This targeted alignment step restores all gradient directions that would otherwise be lost, without altering the downstream fine-tuning procedure. As a result, ABM initialization enables standard LoRA fine-tuning to begin with a lower training loss, converge in significantly fewer steps, and achieve higher final accuracy. The bottom of Fig.A.2 shows these improvements on a T5 model on the GLUE benchmark. The left panel shows training loss, where ABM-LoRA starts with much lower values. The right panel shows the information-loss metric, confirming that ABM sharply reduces discarded gradient information early in training.

Our analysis identifies a *missing piece in vision adaptation*, showing that the same activation-boundary misalignment observed in language models also appears in vision transformers. When applied to ViT backbones on the VTAB-1k benchmark, ABM-LoRA consistently accelerates convergence and improves final accuracy across diverse visual tasks. This finding is significant because, despite the recent surge of parameter-efficient fine-tuning (PEFT) methods for vision transformers (AdaptFormer [3] and Visual-PEFT [10]), most prior works primarily focus on architectural efficiency, rank allocation, or feature re-weighting.

Initialization, however, remains largely overlooked in the vision community, although it plays a crucial role in optimization stability and convergence speed. To the best of our knowledge, ABM-LoRA provides the first systematic analysis of the initialization bottleneck for LoRA in vision transformers, revealing that the same gradient-projection and activation-boundary misalignment identified in language models also limits visual fine-tuning efficiency.

- We identify activation-boundary misalignment as a key cause of slow early-stage convergence in both language and vision-transformer fine-tuning.
- We propose a novel adapter initialization method that aligns activation masks with the pretrained model to recover lost gradient components before fine-tuning.
- We empirically validate our method across multiple pretrained backbones (T5, LLaMA2, and ViT), showing faster convergence and higher accuracy on both GLUE and VTAB-1K benchmarks.

2. Related Work

Parameter-Efficient Fine-Tuning for Vision Transformers. With the success of Vision Transformers [7], numerous parameter-efficient methods have been proposed for visual adaptation. These methods include inserting lightweight adapter modules (*e.g.* AdaptFormer [3]), adding trainable prompts to the input (*e.g.* Visual Prompt Tuning (VPT) [17]), or exploring various PEFT combinations (*e.g.* Visual-PEFT [10]). While these methods achieve strong performance through architectural or prompt-based modifications and rank allocation strategies, they overlook the role of initialization in convergence speed and final accuracy.

Our work complements these approaches by addressing the initialization bottleneck, demonstrating that proper initialization can further improve both convergence speed and final accuracy of LoRA-based visual adaptation methods.

Initialization in Low-Rank Adaptation. Adapter initialization in LoRA has been shown to significantly affect both convergence speed and final performance. LoRA-GA [32] introduced a gradient-matching correction term to stabilize alternating updates of A and B. HRP [4] and Beyond Zero Initialization [19] adjusted initialization scales or enforced orthogonality constraints to improve expressivity. ConsNoTrainLoRA [5] took a data-driven approach, computing LoRA weights in closed form based on activation statistics; however, it remains a one-off, static initialization without further adaptation during fine-tuning. AG-LoRA [31] reallocated low-rank dimensions by decomposing pretrained weights via SVD and redistributing ranks according to normalized activation magnitudes.

Despite their varied strategies, these methods all rely on precomputed information and do not incorporate the true input distribution or the nonlinear activation structure of the model (*e.g.* ReLU activation boundaries) when initializing adapters. Consequently, they overlook the root causes of step-one gradient loss and boundary-induced information loss identified in this work. While recent study [12] on ini-

tialization effects further examined the impact of initialization on LoRA dynamics, they stopped short of linking slow warm-up behavior to specific information-loss mechanisms.

Knowledge Distillation and Activation Alignment. Knowledge distillation has been a cornerstone technique in transfer learning, enabling efficient knowledge transfer from teacher to student networks. Activation Boundary Distillation (ABD) [15] was originally proposed to transfer structural knowledge from teacher to student networks by aligning their binary activation masks. Building on broader distillation methods such as hidden response transfer [26], relational feature transfer [34], and attention map distillation [35], ABD highlights the role of piecewise linear regions induced by ReLU and similar activations [22, 23].

However, these methods are tailored for training-time distillation and do not address the initialization phase of adapters. In contrast, our ABM initialization repurposes the core idea of activation mask alignment specifically for initializing LoRA adapters. From a theoretical analysis of initialization-induced information loss, we show that mismatched activation boundaries can directly discard critical gradient components at step one. ABM mitigates this by dynamically optimizing adapter weights on a representative input batch, aligning activation masks with those of the pretrained model to preserve gradient directions from the very start of fine-tuning. This dynamic, input-aware initialization is fundamentally different from static, precomputed schemes, where it explicitly targets the information-loss bottleneck in LoRA initialization.

3. Information Loss in LoRA

In LoRA, a scaled low-rank adapter is injected as:

$$\Delta = \eta AB, \tag{1}$$

where $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$ denotes the adapter parameters, and $r \ll \min(d,k)$ is the rank of the adapter. In (1), we set $\eta = \frac{\alpha}{r}$, where α is the original LoRA scaling hyperparameter. Let $W_0 \in \mathbb{R}^{d\times k}$ be a frozen, pretrained weight matrix, $f(W;x)$ denote the model's output (*e.g.* logits) on input x using weights W, and y be the corresponding label. The fine-tuned weight is then given by:

$$W = W_0 + \Delta = W_0 + \eta\, A\, B, \tag{2}$$

and the *expected loss* (or risk) of weights W over the data distribution $\mathcal{D}$ is defined by $L(W) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\ell\big(f(W;x),y\big)\big]$, where $\ell\colon \mathbb{R}^k \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is the *per-example loss* (*e.g.* cross-entropy), and $L(W)$ denotes its expectation under $\mathcal{D}$. Then, the gradient at the pretrained point can be written as

$$g = \nabla_W L(W_0) \in \mathbb{R}^{d\times k}, \tag{3}$$

where g simplifies the adapter gradient expressions and quantifies information loss by measuring the mismatch between the full and adapter-projected gradients.

First-Order Update on W. We perform one step of gradient descent on the adapter parameters (A, B) with learning rate γ, evaluated at the initial weight $W_0 + \eta\, A_0B_0$:

$$\begin{aligned} A_1 &= A_0 \;-\; \gamma\, \nabla_A L\big(W_0 + \eta\, A_0B_0\big), \\ B_1 &= B_0 \;-\; \gamma\, \nabla_B L\big(W_0 + \eta\, A_0B_0\big). \end{aligned} \tag{4}$$

By the chain rule and the fact that $W = W_0 + \Delta$, we have $\nabla_\Delta L = \nabla_W L = g$. Hence,

$$\begin{aligned} \nabla_A L &= \nabla_\Delta L\, \frac{\partial(\eta AB)}{\partial A} = \eta\, g\, B_0^\top, \\ \nabla_B L &= \nabla_\Delta L\, \frac{\partial(\eta AB)}{\partial B} = \eta\, A_0^\top\, g. \end{aligned} \tag{5}$$

Substituting the adapter gradients $\nabla_A L = \eta\, g\, B_0^\top$ and $\nabla_B L = \eta\, A_0^\top\, g$ into (4) yields

$$\begin{aligned} A_1B_1 &= \big(A_0 - \gamma\,\eta\, g\, B_0^\top\big)\,\big(B_0 - \gamma\,\eta\, A_0^\top\, g\big) \\ &= A_0B_0 - \gamma\,\eta\big(g\, B_0^\top B_0 + A_0A_0^\top g\big) + \mathcal{O}\big((\gamma\eta)^2\big). \end{aligned} \tag{6}$$

In other words, after one gradient step, the updated adapter product A_1B_1 equals the initial value $\Delta_0 = A_0B_0$ plus the term $\big(g\, B_0^\top B_0 + A_0A_0^\top g\big)$ only.

Intuitive Explanation. We can represent $\big(g\, B_0^\top B_0 + A_0A_0^\top g\big)$ as the projection of the full gradient g onto the tangent space of the initial adapter.

$$\Pi_{T_{\Delta_0}}(g) = g\, B_0^\top B_0 + A_0A_0^\top g. \tag{7}$$

Here, $\Pi_{T_{\Delta_0}}(g)$ projects the full gradient g onto the tangent space of the *initial* adapter $\Delta_0 = A_0B_0$, while $-g\, B_0^\top B_0$ projects g onto the row space of B_0 (directions adjustable by updating B) and $-A_0A_0^\top g$ projects g onto the column space of A_0 (directions adjustable by updating A). Since LoRA's update can only move within this tangent space, any component in $g - \Pi_{T_{\Delta_0}}(g)$ is discarded.

Resulting Weight Update and Information Loss. Because $W_1 = W_0 + A_1B_1$, the net change in the full weight is calculated by

$$\Delta W = W_1 - W_0 = -\,\gamma\,\eta\,\Pi_{T_{\Delta_0}}(g) + \mathcal{O}\big((\gamma\eta)^2\big), \tag{8}$$

and information loss is quantified by the Frobenius norm:

$$\mathcal{I}(A_0, B_0; g) = \|\, g - \Pi_{T_{\Delta_0}}(g)\|_F^2. \tag{9}$$

Extension to Nonlinear Activations. Practical networks interleave linear layers with pointwise nonlinearities σ (*e.g.* ReLU, GELU). We focus on a single such layer, writing $z(W;x) = Wx, \quad h(W;x) = \sigma\big(z(W;x)\big)$, where σ denotes the activation function. If the loss on sample (x,y) is given by $\ell\big(h(W;x),y\big)$, the backpropagated error into the layer is computed as $\delta(x) = \frac{\partial \ell\left(h(W_0;x),y\right)}{\partial h(W_0;x)}$. Then, at W_0, the full-parameter gradient is given by:

$$g = \nabla_W L(W_0) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\sigma'\big(z(W_0;x)\big)\, x^\top\, \delta(x)\big], \tag{10}$$

where σ' denotes the derivative of the activation function. With the LoRA adapter defined as $\Delta_0 = \eta A_0 B_0$, the corresponding gradient with respect to the adapter is

$$\nabla_\Delta L = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sigma'\big(z(W_0+\Delta_0;x)\big)\, x^\top\, \delta(x)\right]. \quad (11)$$

4. Bridging Between LoRA and ABM

Using the definitions of g in (10) and $\nabla_\Delta L$ in (11), we decompose the total information loss into two components. At initialization, LoRA's actual gradient update uses $\Pi_{T_{\Delta_0}}(\nabla_\Delta L)$ rather than the full gradient g. The total discrepancy can be expressed as:

$$\|g - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2. \quad (12)$$

By the Pythagorean theorem in the Hilbert space, this decomposes orthogonally as:

$$\begin{aligned}\|g - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2 &= \|g - \Pi_{T_{\Delta_0}}(g)\|_F^2 \\ &\quad + \|\Pi_{T_{\Delta_0}}(g) - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2.\end{aligned} \quad (13)$$

The first term, $\mathcal{I}(A_0, B_0; g) = \|g - \Pi_{T_{\Delta_0}}(g)\|_F^2$, is the *unavoidable* information loss due to the low-rank constraint of LoRA, which cannot be eliminated without increasing the rank. The second term represents the *reducible* information loss caused by activation boundary mismatch, which our ABM initialization targets.

Since projection is non-expanding, we have:

$$\|\Pi_{T_{\Delta_0}}(g) - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2 \le \|g - \nabla_\Delta L\|_F^2. \quad (14)$$

Using g in (10) and $\nabla_\Delta L$ in (11), we obtain

$$g - \nabla_\Delta L = \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\big(\sigma'(W_0x) - \sigma'((W_0+\Delta_0)x)\big)\, x^\top\, \delta(x)\Big]. \quad (15)$$

For ReLU, $\sigma'(u) = \mathbf{1}_{\{u>0\}} \in \{0,1\}$ for almost every u, so each σ' acts as an *activation mask*, indicating whether a neuron is active. If activation boundaries are aligned by the proposed ABM initialization, *i.e.*

$$\sigma'(W_0x) = \sigma'((W_0+\Delta_0)x) \quad \forall x, \quad (16)$$

then $g - \nabla_\Delta L = 0$ in (15). By (14), this makes the reducible term in (13) vanish:

$$\|\Pi_{T_{\Delta_0}}(g) - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2 = 0. \quad (17)$$

Therefore, with ABM initialization, the total information loss in (13) reduces to:

$$\|g - \Pi_{T_{\Delta_0}}(\nabla_\Delta L)\|_F^2 = \mathcal{I}(A_0, B_0; g), \quad (18)$$

which is the minimal achievable loss given rank constraint.

In conclusion, while the low-rank constraint inherently causes some information loss, the proposed ABM initialization in (16) eliminates the *additional* loss due to activation boundary mismatch, ensuring that LoRA starts from an optimal initialization point within the low-rank constraint.

5. ABM as Adapter Initialization

Activation Boundary Distillation (ABD) [15] has been originally proposed as a knowledge-distillation technique: by matching the binary activation masks of a teacher's intermediate layers, student models learn to inherit the same feature-space partitioning. In contrast, we repurpose activation boundary alignment as an *initialization* criterion for LoRA adapters. As shown in the previous section, any mismatch between the pretrained activation mask and the adapter-augmented mask introduces the *reducible information loss* (the second term in (13)). Consequently, by initializing Δ_0 such that it closely satisfies

$$\sigma'(W_0x) \approx \sigma'((W_0+\Delta_0)x), \quad \text{for samples } x \sim \mathcal{D}, \quad (19)$$

we substantially reduce this initialization-induced reducible loss prior to fine-tuning. This shift—from distillation at train time to activation-matching at init time—forms the core of our Activation Boundary Matching (ABM) method.

Stage 1: Activation Boundary-based LoRA Initialization. Let the frozen base model be denoted by W_0, and define the pretrained and fine-tuned adapters as

$$\Delta^{\mathrm{pt}} = \eta\, A^{\mathrm{pt}} B^{\mathrm{pt}}, \quad \Delta = \eta\, A\, B, \quad (20)$$

respectively. For each mini-batch $\{x_i\}_{i=1}^N$ and layer l, we define the pre-activations as

$$z_{i,l}^{\mathrm{pt}} = \big(W_{0,l} + \Delta_l^{\mathrm{pt}}\big)\, x_i, \quad z_{i,l} = \big(W_{0,l} + \Delta_l\big)\, x_i, \quad (21)$$

and the sign $\tau_{i,l} = \mathrm{sgn}\big(z_{i,l}^{\mathrm{pt}}\big) \in \{-1, 1\}$. We then minimize a weighted, squared-hinge boundary loss:

$$L_{\mathrm{ABM}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L} w_l^2 \left[\max\big(0, -\tau_{i,l}\, z_{i,l} + m\big)\right]^2, \quad (22)$$

where $m > 0$ is a margin and w_l increases with l. We compute the layer-specific weight w_l so that deeper layers receive larger emphasis: $w_l = \left(\frac{l+1}{L}\right), \quad l = 0, 1, \ldots, L-1$, where L is the total number of layers considered for boundary matching. Starting from random (A_0, B_0), we update

$$\begin{aligned} A_{t+1} &= A_t - \mu\, \nabla_A L_{\mathrm{ABM}}, \\ B_{t+1} &= B_t - \mu\, \nabla_B L_{\mathrm{ABM}}, \end{aligned} \quad (23)$$

for T steps with step size μ, where $\Delta_0 = \eta A_T B_T$. No labels are needed in this stage.

Stage 2: Task-specific Fine-Tuning. With Δ_0 initialized via the proposed ABM, we fine-tune only the LoRA adapters on the downstream task (*e.g.* classification or QA) using the standard cross-entropy loss. The base model W_0 remains frozen throughout. We observe faster convergence and, in some cases, higher final accuracy than both random and pretrained adapters, showing that ABM provides a better starting point for task-specific learning.

Table 1. **GLUE dev set accuracy on T5-Base.** *Upper block*: the results as reported in the original LoRA-GA paper [32]. *Lower block*: our reproduction of LoRA-GA using identical hyperparameters, and ABM-LoRA(GA) initialized from this reproduced LoRA-GA model. Best results in each block are highlighted in **bold**.

	MNLI	SST-2	CoLA	QNLI	MRPC	Average
Size	393k	67k	8.5k	105k	3.7k	
Reported in LoRA-GA (upper block)						
Full	86.33 ± 0.00	94.75 ± 0.21	80.70 ± 0.24	93.19 ± 0.22	84.56 ± 0.73	87.91
LoRA	85.30 ± 0.04	94.04 ± 0.11	69.35 ± 0.05	92.96 ± 0.09	68.38 ± 0.01	82.08
PiSSA	85.75 ± 0.07	94.07 ± 0.06	74.27 ± 0.39	93.15 ± 0.14	76.31 ± 0.51	84.71
rsLoRA	85.73 ± 0.10	$\mathbf{94.19 \pm 0.23}$	72.32 ± 1.12	93.12 ± 0.09	52.86 ± 2.27	79.64
LoRA+	$\mathbf{85.81 \pm 0.09}$	93.85 ± 0.24	77.53 ± 0.20	93.14 ± 0.03	74.43 ± 1.39	84.95
DoRA	85.67 ± 0.09	94.04 ± 0.53	72.04 ± 0.94	93.04 ± 0.06	68.08 ± 0.51	82.57
AdaLoRA	85.45 ± 0.11	93.69 ± 0.20	69.16 ± 0.24	91.66 ± 0.05	68.14 ± 0.28	81.62
LoRA-GA	85.70 ± 0.09	94.11 ± 0.18	$\mathbf{80.57 \pm 0.20}$	$\mathbf{93.18 \pm 0.06}$	$\mathbf{85.29 \pm 0.24}$	**87.77**
Our reproduction with official code (lower block)						
LoRA	83.78 ± 0.01	93.08 ± 0.04	69.73 ± 0.01	92.65 ± 0.04	76.88 ± 0.22	83.22
LoRA-GA	85.12 ± 0.06	$\mathbf{94.07 \pm 0.14}$	78.91 ± 0.10	92.86 ± 0.05	86.27 ± 0.28	87.45
ABM-LoRA(GA)	$\mathbf{85.33 \pm 0.01}$	93.58 ± 0.23	$\mathbf{80.73 \pm 0.15}$	$\mathbf{93.03 \pm 0.06}$	$\mathbf{86.44 \pm 0.22}$	**87.82**

6. Implementation Details

Our training pipeline was built on top of three pretrained Transformer backbones: T5-Base [24], LLaMA2-7B [29], and ViT-B/16 [7]. For each backbone, we augmented specific layers with LoRA adapters via the `PEFT` library, configuring the adapter rank r and scaling factor α, while keeping all original backbone parameters frozen.

- *T5-Base:* LoRA applied to each `DenseReluDense` sublayer in encoder and decoder ($r = 8$, $\alpha = 16$).
- *LLaMA2-7B:* LoRA adapters were added to the attention projection layers (`q_proj`, `v_proj`) in all 32 decoder layers ($r = 8$, $\alpha = 16$).
- *ViT-B/16:* We augmented the query and value projection matrices in 12 attention layers ($r = 8$, $\alpha = 16$).

In all cases, the fine-tuned model shares the same architecture but initializes all LoRA parameters to zero: A is initialized using Kaiming initialization and B is initialized to zero, ensuring $\Delta_0 = AB = 0$ at the start of training.

Model Construction and Layer Detection. To ensure correct pairing of corresponding layers between the pretrained and fine-tuned models, we employed a custom `get_layer_config()` utility that automatically extracts each layer's full path and establishes one-to-one mappings.

Hook-based Activation Extraction and Loss Computation. Forward hooks were registered on the pre-activation outputs of all target layers, and activations were stored in global dictionaries keyed by layer name. The activation-boundary matching loss was computed by (22), which applies a hinge-like penalty to sign-mismatch regions between pretrained and fine-tuned activations, with deeper layers weighted heavily to emphasize structural boundaries.

7. Experiments

Settings. We evaluated our method on three Transformer backbones (T5-Base, LLaMA2-7B, and ViT-B/16) across diverse tasks. For natural language understanding, we used the GLUE benchmark [30] with the T5-Base backbone and reported the average accuracy score. For the dialogue task, we adopted the WizardLM dataset [33], using Llama2-7B and reported task-specific accuracy metrics. Finally, to validate robustness, we conducted an ablation study across the following dimensions: 1) different pretrained backbones, 2) margin m, 3) layer-specific weights w_l in (22), 4) the subset of layers selected for activation boundary matching, and 5) the total number of matching steps.

All models were trained on 4 NVIDIA 3090 GPUs. For all GLUE tasks, we fine-tuned the LoRA-augmented T5-Base model with rank $r = 8$, scaling factor $\alpha = 16$, learning rates of 1×10^{-4} and 3×10^{-4} for the baseline and the ABM, respectively, and a maximum sequence length of 128. Optimization used AdamW with cosine learning-rate scheduling, a warm-up ratio of 0.03, a max gradient norm of 1.0, and no weight decay. Training was performed for 8 epochs on MRPC (BS 32), 1 epoch on MNLI (96), 4 epochs on CoLA (32), and 1 epoch each on SST-2 and QNLI (32). In the ABM initialization stage (Stage 1), we applied ABM to both encoder and decoder across 6 layers using a sequential weighting scheme. This stage lasted 1 epoch (100 steps) with a learning rate 3×10^{-4} and a margin m of 0.5.

We used Llama-2-7B for WizardLM dataset with LoRA rank $r = 8$ and scaling factor $\alpha = 16$, a base learning rate of 2×10^{-5}, warm-up ratio 0.03 followed by cosine decay, one epoch of training ($E = 1$), a batch size of 1 with gradient accumulation over 32 steps, and sequence length $T = 1024$. For standard LoRA, we applied a dropout of 0.1. In ABM-LoRA Stage 1, we also used dropout 0.1, performed activation-boundary matching on the last 16 of 32 decoder layers with sequential layer-wise weights (as in T5), fixed the margin at 0.5, and ran the matching for 100 steps, using a learning rate of 2×10^{-4}.

Baselines. We compared ABM-LoRA against two strong baselines: 1) *Vanilla LoRA*, which inserts LoRA adapters with Kaiming-initialized A and zero-initialized B; 2) *LoRA-GA*, which improves LoRA stability via gradient approximation as described in [32]. In LoRA-GA, the rank is dynamically adjusted during the gradient estimation process; thus, LoRA-GA operates with $r = 16$. For a fair comparison, when using LoRA-GA for initialization, we performed ABM at the baseline rank of $r = 8$.

Table 2. **GLUE dev set accuracy when initialized from a vanilla LoRA adapter**. We compare the baseline LoRA with our ABM-LoRA, initialized from the same adapter.

	MNLI	SST-2	CoLA	QNLI	MRPC	Average
LoRA	83.79	93.00	69.51	92.59	75.74	82.92
ABM-LoRA	**85.29**	**93.58**	**81.11**	**93.04**	**88.24**	**88.25**

7.1. Results on T5 with GLUE

We compared the proposed ABM-LoRA against: 1) *Full*, fine-tuning all parameters; 2) *LoRA* [16], vanilla LoRA; 3) original-structure variants—*rsLoRA* [18], *LoRA+* [13], *PiSSA* [21]; 4) structure-modified variants—*DoRA* [20], *AdaLoRA* [37]; 5) *LoRA-GA* [32], which stabilizes LoRA via gradient approximation.

We fine-tuned T5-Base on five GLUE tasks (MNLI, SST-2, CoLA, QNLI, MRPC), evaluating on each development set using accuracy and reporting the mean over three random seeds. For GLUE fine-tuning, we used prompt-based tuning: each task label was mapped to a natural language token (*e.g.* "positive", "negative"), and the normalized probability of that token served as the model's prediction.

Comparison with LoRA-GA and Baselines. Table 1 has two blocks. The *upper block* shows the results as reported in the original LoRA-GA [32] paper. The *lower block* shows our own reproduction using the official LoRA-GA code and the same hyperparameter settings. Due to GPU memory constraints, we used smaller gradient sampling configurations compared to the original work. Specifically, we set the sampled batch size to 8 for MRPC, SST2, and QNLI , and 2 for MNLI and COLA. This reduced sampling compared to the original work likely explains the performance discrepancy from the published numbers. All other settings (*e.g.* dataset splits, random seeds, learning rate schedule) are shared between our reproduction and ABM-LoRA, making the comparison in the lower block particularly fair.

In the upper block of Table 1, LoRA-GA achieves the highest accuracy on CoLA, QNLI, and MRPC—an impressive result given the small size of these datasets, showing clear improvements over vanilla LoRA. In the lower block, our ABM-LoRA (GA) continues this trend, surpassing the reproduced LoRA-GA on CoLA, QNLI, and MRPC, and achieving the highest accuracy among all LoRA-based methods. Although ABM-LoRA is not the top performer on MNLI and SST-2, it attains the highest average accuracy across the five GLUE tasks.

Robustness to Sub-Optimal Pretrained Model. ABM-LoRA(GA) was initialized from a high-performance LoRA-GA model. However, in real-world scenarios, such a strong pretrained adapter may not be readily available. To evaluate the robustness of our approach under more realistic conditions, we initialized ABM-LoRA from a standard (vanilla) LoRA adapter with lower baseline performance, as shown in Table 2. We then fine-tuned on the same GLUE tasks and compared accuracies to determine whether ABM initialization still yields significant gains. Notably, these gains closely resemble those achieved by ABM-LoRA(GA) in Table 1, and even surpass them on the CoLA and MRPC tasks, yielding a higher overall average accuracy. These results show that ABM initialization remains robust and effective even when derived from a weaker pretrained adapter.

Specifically, on the CoLA task, vanilla LoRA achieves only 69.51% accuracy, whereas our ABM-LoRA initialized from the same vanilla adapter reaches 81.11% after fine-tuning. This improvement shows that matching activation boundaries, even from a suboptimal pretrained model, offers a stronger initialization that stabilizes LoRA training and accelerates convergence.

Task Transferability. To further evaluate the generalization capability of ABM initialization, we considered scenarios in which the pretrained model and fine-tuned models are trained on different tasks. This setup simulates real-world cases where a pretrained adapter for the exact target task is unavailable, and tests whether ABM initialization from a related task can still yield meaningful performance gains.

We constructed four cross-task pairs from the GLUE benchmark: QNLI→MNLI, SST-2→CoLA, MNLI→QNLI, and CoLA→SST-2. For each pair, we first trained a vanilla LoRA adapter on task A (*e.g.* QNLI), then applied ABM initialization using that adapter to train a new LoRA on task B (*e.g.* MNLI). We compared this against standard LoRA trained from scratch on task B without ABM initialization, measuring dev-set accuracy for each configuration in Table 4. These experiments demonstrate that ABM-LoRA yields substantial improvements, even when using sub-optimal or cross-task pretrained model, and consistently approaches the performance of full fine-tuning.

Notably, when initialized from CoLA (a smaller task) and fine-tuned on SST-2 (a larger task), ABM-LoRA achieves 94.38% accuracy, the highest SST-2 result among all methods in Table 1. This result shows ABM's ability to transfer robust activation-boundary information across tasks, achieving strong performance on larger-scale tasks.

As shown in Tables 1, 2 and 4, our ABM-LoRA initialization consistently outperforms vanilla LoRA and all other baseline methods, achieving results on par with full fine-tuning. Across diverse settings, we demonstrate that ABM-LoRA delivers stable performance gains and robust generalization regardless of the quality of the pretrained adapter or the alignment between pretraining and fine-tuning tasks.

7.2. Results on ViT-B/16

Models and Datasets. We evaluated ABM-LoRA on the **ViT-B/16** backbone using the **VTAB-1K** benchmark [36] , a suite of 19 diverse vision datasets. As specified in our main Implementation Details, LoRA adapters (rank $r = 8$, scaling factor $\alpha = 16$) were applied to the query and value projection matrices in each of the 12 attention layers.

Table 3. **VTAB-1K evaluation results** with ViT-B/16 backbone on 19 diverse visual recognition tasks. Our method results are highlighted in grey. We present the best result in bold and the second best as underlined.

		Natural							Specialized				Structured									
Method	Param (M)	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dst	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Structured Mean	Overall Mean
LoRA	-	73.9	93.5	66.1	**98.7**	88.3	84.2	51.1	85.6	95.3	79.8	**72.9**	79.9	62.9	47.0	76.5	85.4	49.5	28.9	39.2	58.7	71.5
PiSSA	-	73.8	93.4	66.2	98.7	88.3	85.5	50.8	84.9	95.3	79.1	72.8	78.4	61.0	45.8	75.5	85.2	50.4	28.2	33.5	57.3	70.9
Orthogonal	-	72.9	92.3	**66.5**	98.5	**88.7**	84.1	51.2	85.5	95.8	**80.6**	72.2	80.1	62.0	46.9	78.3	83.0	49.0	27.9	36.0	57.9	71.1
Gaussian	-	73.3	**93.8**	66.2	98.4	88.6	**85.7**	50.3	86.0	**95.9**	80.2	72.6	79.5	62.3	46.5	**79.5**	84.1	49.8	29.7	38.9	58.8	71.6
ABM-LoRA (a)	-	72.6	92.8	63.2	98.1	87.7	84.1	**51.4**	85.0	95.7	78.2	71.5	81.5	63.0	47.2	77.5	86.0	**52.1**	31.2	**45.2**	**60.5**	**71.8**
ABM-LoRA (b)	-	**74.2**	92.8	65.6	97.4	87.6	84.8	51.4	85.5	95.2	77.8	72.3	81.5	63.6	**47.6**	74.5	83.4	51.8	29.1	44.4	59.5	71.6
ABM-LoRA (c)	-	72.2	93.3	62.7	98.6	87.8	84.8	49.1	**87.0**	95.5	77.9	72.2	81.4	62.8	46.8	77.5	**86.4**	51.9	31.4	43.5	60.2	71.7
ABM-LoRA (d)	-	72.7	92.8	63.4	98.3	87.5	85.3	50.5	85.2	95.6	76.8	72.6	**82.1**	**63.8**	47.2	73.7	85.7	51.4	**31.9**	44.8	60.1	71.6

Table 4. **Same-task and cross-task initialization on GLUE**. *N/A* indicates that Vanilla LoRA has no pretrained initialization.

Pretrained	**Fine-tuned**	**LoRA**	**ABM-LoRA**
MNLI	MNLI	83.79	**85.29**
QNLI	MNLI	*N/A*	**85.24**
CoLA	CoLA	69.51	**81.11**
SST-2	CoLA	*N/A*	**78.52**
QNLI	QNLI	92.59	**93.04**
MNLI	QNLI	*N/A*	**93.04**
SST-2	SST-2	93.00	**93.58**
CoLA	SST-2	*N/A*	**94.38**

Table 5. **LoRA variants comparison** (rank = 8, dropout = 0.1) under strict evaluation settings

Variant	MT-Bench	LC (%)	WR (%)
LoRA	5.89	42.16	46.27
rsLoRA	5.98	44.02	48.51
Orthogonal	5.95	45.25	49.88
Gaussian	5.98	42.78	48.63
ABM (standard A)	5.85	42.68	48.26
ABM (standard B)	5.92	**45.53**	49.51
ABM (Orthogonal A)	6.00	**45.53**	**49.94**
ABM (Orthogonal B)	**6.03**	43.92	48.51

For the ABM initialization (Stage 1), we focused on a subset of hyperparameters based on our findings from the T5 experiments. We applied ABM to the **last 6** attention layers with a fixed margin $m = 0.5$. As shown in Table 3 (grey rows), we report results for four configurations of our method, ablating the number of matching steps and the layer weighting scheme: (a) 500 steps, *uniform* weighting, (b) 500 steps, *quadratic* weighting, (c) 1000 steps, *uniform* weighting, and (d) 1000 steps, *quadratic* weighting. We compared these configurations against several strong baselines, including standard LoRA initialization ('Vanilla') [16], PiSSA [21], and LoRA with different random initializations ('Orthogonal', 'Gaussian').

Results. Table 3 presents results across all 19 VTAB-1K tasks. ABM-LoRA achieves the highest overall mean accuracy (71.8% for configuration (a)), demonstrating effectiveness across this diverse benchmark. Examining performance by task group, we observe: (1) *Natural* tasks (7 datasets): All methods perform competitively, as these tasks share similar distributions with ImageNet pretraining and require relatively small adaptation; (2) *Specialized* tasks (4 datasets): ABM-LoRA maintains strong performance, with configuration (c) achieving 87.0% on Camelyon; (3) *Structured* tasks (8 datasets): ABM-LoRA shows notable gains on several tasks requiring spatial reasoning—these tasks involve 3D geometry, counting, and spatial relationships that differ substantially from natural image classification.

Specifically on structured reasoning tasks, ABM-LoRA demonstrates substantial improvements, achieving 60.5% structured mean compared to 58.7% for vanilla LoRA (+1.8%). Notable per-task gains include: sNORB-Ele (+6.0%), sNORB-Azim (+3.0%), Clevr-Count (+2.2%), and dSpr-Ori (+2.6%). These tasks require 3D viewpoint prediction, object counting, and spatial localization—capabilities where activation boundary matching provides a stronger initialization than learning from scratch.

These results suggest that while ABM initialization yields overall gains (highest mean accuracy), its benefits are most evident in tasks involving geometric and spatial reasoning rather than uniformly across all vision tasks.

7.3. Results on LLaMA2-7B

Models and Datasets. To evaluate the generality and scalability of the proposed ABM-LoRA beyond language understanding tasks, we conducted dialogue experiments using the LLaMA2-7B [29] backbone fine-tuned on the WizardLM dataset. LLaMA2-7B is a decoder-only model that employs the SiLU (Sigmoid Linear Unit) [9] activation function instead of ReLU [1]; accordingly, during ABM initialization we align the pre-activation outputs immediately before the nonlinearity by targeting the `gate_proj` projection in each layer. Among the 32 decoder layers, we explored two layer selection strategies for ABM: (A) selecting layers 12 to 23, and (B) selecting the latter 16 layers (layers 16 to 31). In both settings, layer-wise weights were assigned following the sequential weighting scheme used in T5. The matching process was executed for 100 steps with a margin value of 0.5. Additional ablation studies on the impact of different margin values are provided in the supplementary material.

(a) Margin m (b) # Layers (c) Weighting w_ℓ (d) Matching steps (e) Scope

Figure 2. **Violin plots of accuracy distributions across ablation factors**.

Table 6. **Initialization time and memory usage** for LoRA and ABM-LoRA on GLUE dev sets.

Model	Metric	Dataset				
		MNLI	SST-2	CoLA	QNLI	MRPC
standard LoRA	Memory (GB)	2.71				
ABM-LoRA	Init Time (s)	16.95	18.04	17.30	16.65	17.05
	Memory (GB)	1.75	1.74	1.74	1.75	1.75

For evaluation on dialogue tasks, the WizardLM-fine-tuned LLaMA2-7B model was assessed on both MT-Bench [38] and AlpacaEval [11]. MT-Bench contains multi-turn conversation scenarios, and we report the first-turn response quality as rated by GPT-3.5-Turbo. AlpacaEval measures instruction-following ability on single-turn prompts, yielding both length-controlled win rates and overall win rates relative to baseline models. We used GPT-3.5-Turbo (instruction-tuned) as the annotator, comparing each model's response to reference outputs defined in `alpaca_eval.json`. Additionally, AlpacaEval reports two metrics: *Length-Controlled Win Rate* [8] and *Win Rate*. *Win Rate* denotes the percentage of prompts for which a model's response is rated superior to that of a competing model, reflecting overall response quality. *Length-Controlled Win Rate* is computed after controlling for differences in output length between the compared responses, ensuring that length does not bias the evaluation. This metric allows fairer assessment of response quality in cases where evaluation may favor longer or shorter outputs.

Results. Table 5 compares LoRA initialization methods for dialogue evaluation. When ABM is applied to vanilla LoRA, it achieves substantial improvements: +3.37pp on LC (45.53% vs. 42.16%) and +3.24pp on WR (49.51% vs. 46.27%) while achieving 5.92 on MT-Bench. When combined with orthogonal initialization, ABM further improves performance over the orthogonal baseline: +0.28pp on LC (45.53% vs. 45.25%) and +0.06pp on WR (49.94% vs. 49.88%), while achieving 6.03 on MT-Bench. Overall, ABM variants achieves the best performance across different metrics, demonstrating that ABM transfers effectively to decoder-only models and dialogue tasks.

7.4. Ablation Study

To analyze the impact of Stage 1 ABM initialization, we performed a grid-search ablation over five factors: 1) Margin m in (22): $\{0.5, 1.0, 2.0\}$, 2) Number of layers: $\{6, 12\}$[1], 3) Layer weighting w_ℓ in (22): sequential vs. uniform, 4) Matching steps: $\{50, 100\}$, and 5) Scope: encoder+decoder vs. encoder only. For this, we evaluated 48 configurations on the GLUE dev set using ABM-LoRA (T5-Base) and reported the average accuracy. As shown in Fig.2, the setup with $m = 0.5$, six deepest layers, and sequential weighting yields the highest median accuracy. Increasing steps from 50 to 100 gives modest gains, and adding both encoder and decoder layers provides a slight boost. The optimal configuration ($m = 0.5$, 6 layers, sequential w_ℓ, 100 steps, and encoder+decoder) achieves 81.98% average accuracy. Using all layers or applying a uniform weighting led to lower accuracy, indicating that focusing on a subset of layers with gradually increasing weights provides the most effective initialization.

Complexity Analysis. Table 6 reports the initialization-phase (Stage 1) cost of ABM-LoRA compared to standard LoRA. In our method, Stage 2 corresponds to the LoRA fine-tuning process, which can take from 10 minutes to 1.5 hours depending on the dataset. By contrast, Stage 1 is a lightweight initialization step performed before fine-tuning. Notably, ABM-LoRA consumes less GPU memory during this stage (1.74–1.75 GB) compared to standard LoRA (2.71 GB), and its runtime is negligible: Stage 1 completes in under 20 seconds, making its overhead negligible relative to the fine-tuning stage. *Additional results are provided in the supplementary materials.*

8. Conclusion

We identify a key bottleneck in LoRA fine-tuning: random adapter initialization projects the full gradient into a mismatched low-rank space, causing information loss and slow convergence. To overcome this, We propose ABM-LoRA, which aligns adapter activation boundaries with those of the frozen pretrained model before LoRA fine-tuning. This simple yet principled pre-alignment greatly reduces initialization-induced loss and accelerates convergence. Empirically, ABM-LoRA outperforms baselines across tasks and architectures—language understanding (T5-Base on GLUE), dialogue generation (LLaMA2-7B), and vision recognition (ViT-B/16 on VTAB-1K). Notably, it shows substantial gains on geometry-aware vision tasks, indicating effective cross-modal transfer.

[1] T5 has 12 layers in encoder and 12 layers in decoder and "6" refers to the 6 deepest layers in each.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 2, 7

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 1

[3] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, pages 16664–16678, 2022. 2

[4] Yuzhu Chen, Yingjie Wang, Shi Fu, Li Shen, Yongcheng Jing, Xinmei Tian, and Dacheng Tao. Hrp: High-rank preheating for superior lora initialization. *arXiv preprint arXiv:2502.07739*, 2025. 1, 2

[5] Debasmit Das, Hyoungwoo Park, Munawar Hayat, Seokeon Choi, Sungrack Yun, and Fatih Porikli. Consnotrainlora: Data-driven weight initialization of low-rank adapters using constraints. *arXiv preprint arXiv:2507.08044*, 2025. 1, 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 1

[7] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 5

[8] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024. 8

[9] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018. 7

[10] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024. 2

[11] Tatsunori B. Hashimoto, Yann Dubois, Balázs Galambosi, et al. Alpacaeval: An automatic evaluator for instruction-following language models. https://github.com/tatsu-lab/alpaca_eval, 2023. 8

[12] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on lora finetuning dynamics. In *NeurIPS*, 2024. 2

[13] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *ICML*, 2024. 6

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[15] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019. 3, 4

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 1, 6, 7

[17] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 2

[18] Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023. 6

[19] Shiwei Li, Xiandi Luo, Xing Tang, Haozhao Wang, Hao Chen, Weihong Luo, Yuhua Li, Xiuqiang He, and Ruixuan Li. Beyond zero initialization: Investigating the impact of non-zero initialization on lora fine-tuning dynamics. In *ICML*, 2025. 2

[20] 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 *ICML*, 2024. 6

[21] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *NeurIPS*, 2024. 6, 7

[22] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014. 3

[23] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013. 3

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020. 1, 5

[25] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *ACL*, 2016. 1

[26] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 3

[27] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004. 2

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1

[29] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Yasmine El Kalaki, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 5, 7, 3

[30] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019. 1, 5

[31] Qingchen Wang and Shengyu Shen. Activation-guided low-rank parameter adaptation for efficient model fine-tuning. *IEEE Access*, 2025. 1, 2

[32] Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024. 1, 2, 5, 6

[33] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *ICLR*, 2024. 5

[34] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017. 3

[35] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 3

[36] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 1, 6

[37] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*, 2023. 6

[38] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023. 8

ABM-LoRA: Activation Boundary Matching for Fast Convergence in Low-Rank Adaptation

Supplementary Material

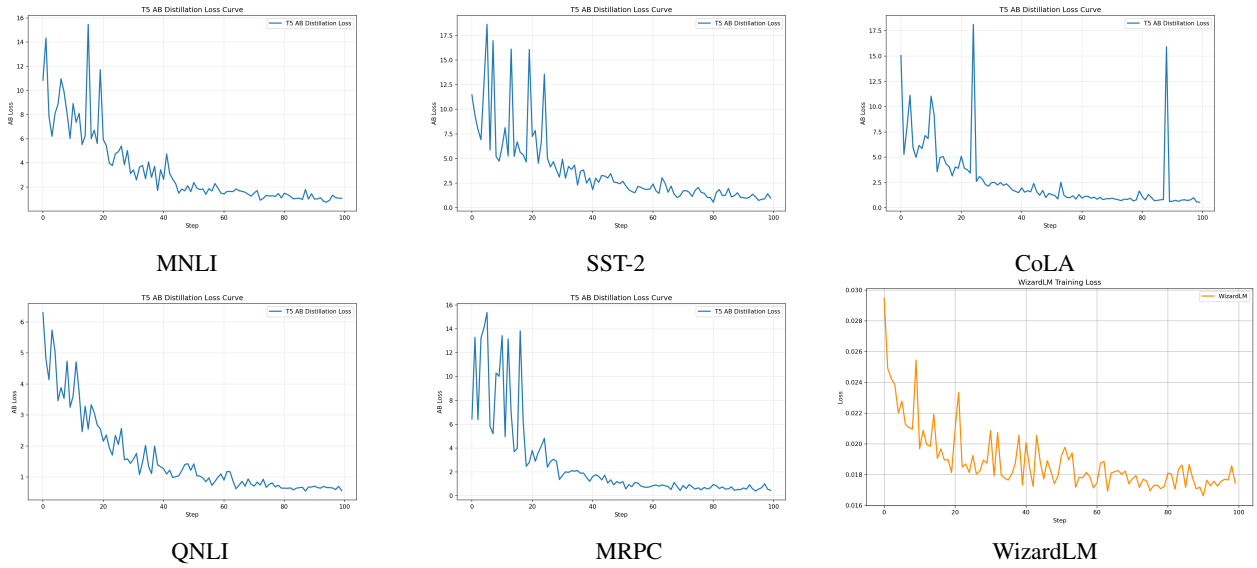


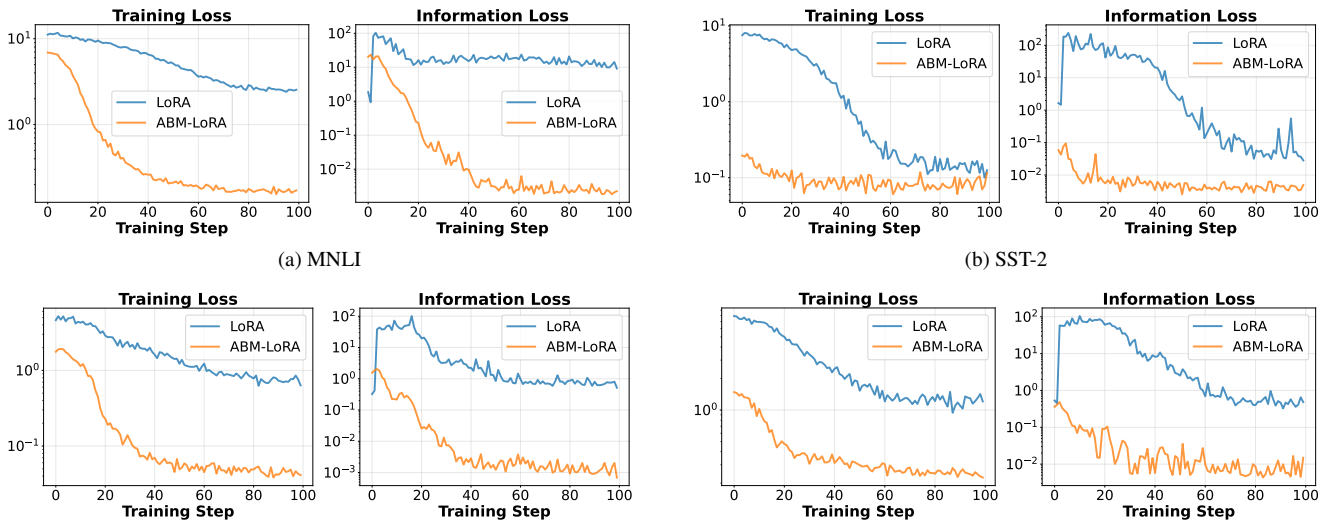
Figure A.1. **Stage 1 ABM loss evolution** on five subsets of the T5 GLUE benchmark (MNLI, SST-2, CoLA, QNLI, MRPC) and WizardLM dataset on LLaMA2-7B. In each case, the ABM loss exhibits a clear and steady downward trend throughout Stage 1, confirming that the activation boundary matching objective is being effectively optimized.

A. Stage 1 ABM Loss Dynamics

Fig. A.1 illustrates the evolution of the Stage 1 ABM loss in Eq. (22) (main paper) on five distinct subsets of the T5 GLUE benchmark and WizardLM dataset on LLaMA2-7B. In each case, the ABM loss exhibits a clear and steady downward trend throughout Stage 1, confirming that the activation boundary matching objective is being effectively optimized. Notably, CoLA shows a brief, transient upward spike around iteration 90—almost certainly a sampling-variance effect due to CoLA's relatively small size, where individual mini-batches may contain particularly "hard" examples whose activation boundaries are harder to match. However, the consistent reduction across all datasets demonstrates that our ABM initialization successfully aligns the adapter's parameters with the pretrained model's activation boundaries, enabling the adapter to learn informative gradients quickly and setting the stage for faster convergence in later fine-tuning steps.

B. Training and Information-Loss Curves on All GLUE Benchmark Sets

Fig. A.2 reproduces the bottom panel of Fig. A.2 (main text) for the remaining four GLUE dev sets (MNLI, SST-2, QNLI, MRPC). As shown in Fig. A.2, standard LoRA maintains low information loss for the first 1–3 iterations but then exhibits a sudden spike around iteration 3. This spike arises because, as updates accumulate, the true gradient rapidly becomes more complex—driven by batch-sample heterogeneity and parameter interactions—and thus no longer lies within LoRA's randomly fixed low-rank subspace. In contrast, ABM-LoRA aligns its subspace with the dominant gradient directions from the very first iteration, producing a smooth, steadily decreasing information-loss curve without any spikes. We observe this behavior consistently across all datasets. Moreover, ABM-LoRA also starts from a lower initial training loss and converges more steeply than standard LoRA. These results confirm that the benefits of ABM initialization generalize across the full GLUE benchmark.



(a) MNLI (b) SST-2

(c) QNLI (d) MRPC

Figure A.2. **Training and information-loss curves** for LoRA vs. ABM-LoRA on T5, shown for four GLUE dev sets (MNLI, SST-2, QNLI, MRPC). ABM-LoRA not only starts from a substantially lower initial training loss, but also reduces both training and information-loss more steeply than standard LoRA. These consistent performance gains across diverse GLUE tasks demonstrate that ABM initialization effectively mitigates early-stage information loss and accelerates convergence.

C. Training Loss Evolution on VTAB-1K

To demonstrate that ABM initialization extends to vision transformers, we evaluate ViT-B/16 with rank-8 adapters across all 19 tasks in the VTAB-1K benchmark. To illustrate the key advantages of our approach—faster early-stage convergence and reduced initial training loss—we present training curves for four representative tasks in Fig. A.3.

Across these examples, ABM-LoRA consistently demonstrates steeper loss reduction in the early training phase. On CIFAR-100, SUN397, and Caltech101, the improvement is particularly visible within the first 5 epochs, where ABM-LoRA achieves substantially lower training loss than standard LoRA. Even on Oxford-IIIT Pet, where convergence patterns are more similar, ABM-LoRA maintains a consistent advantage throughout training. These curves illustrate how activation boundary matching enables vision adapters to start from a better-informed initialization, leading to more efficient optimization from the very beginning of fine-tuning.

D. Handling the Non-Differentiable Hinge Term

Recalling the loss in Eq. (22) (main paper):

$$L_{\text{ABM}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L} w_l^2 \left[\max\!\big(0,\, -\tau_{i,l} z_{i,l} + m\big)\right]^2, \quad \text{(A.1)}$$

we differentiate it with respect to the finetuned adapter's activation $z_{i,l}$. Because the squared hinge term (first introduced in [27]) is non–differentiable at the hinge point $-\tau_{i,l} z_{i,l} + m = 0$, we adopt its sub-gradient. This leads to the piece-wise expression:

$$\frac{\partial L_{i,l}}{\partial z_{i,l}} = \begin{cases} -2\, w_l^2\, \tau_{i,l}\, \big(-\tau_{i,l} z_{i,l} + m\big), & \text{if } -\tau_{i,l} z_{i,l} + m > 0, \\ 0, & \text{otherwise.} \end{cases} \quad \text{(A.2)}$$

The gradient is *non-zero only when the finetuned adapter violates the margin* m:

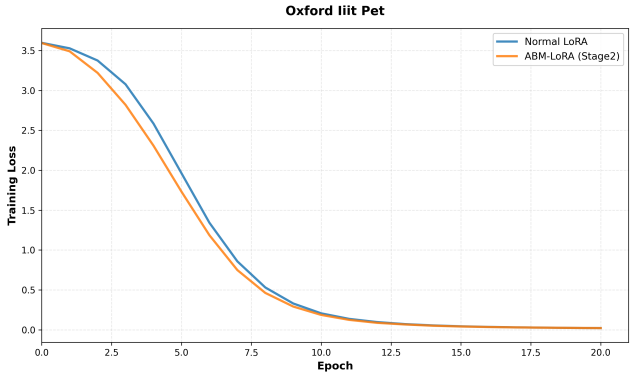
(i) if the pretrained adapter's neuron is activated ($\tau_{i,l} = +1$) and the finetuned adapter response falls below m, or

(ii) if the pretrained adapter's neuron is suppressed ($\tau_{i,l} = -1$) and the finetuned adapter response exceeds $-m$.

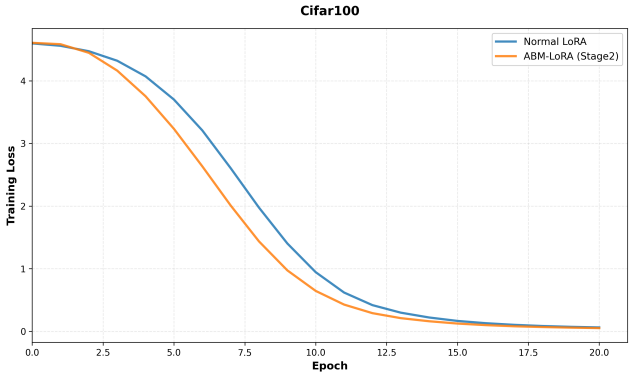
In both cases the negative sign drives $z_{i,l}$ toward the correct side of the margin, thereby lowering the loss and aligning the finetuned adapter's activation pattern with that of the pretrained adapter.

E. Ablation Study on Margin and Layer Selection for LLaMA2-7B

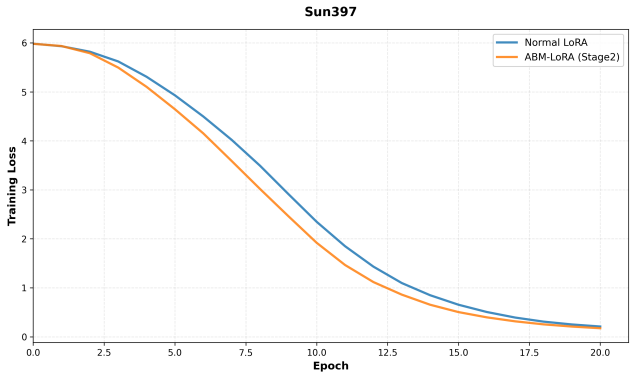
To understand the impact of key hyperparameters in ABM initialization, we conduct an ablation study on LLaMA2-7B, varying two critical factors: the margin value m and the number of layers where activation boundary matching is applied. Tab. A.1 presents results for six configurations combining two margin values ($m \in \{0.5, 1.0\}$) with three



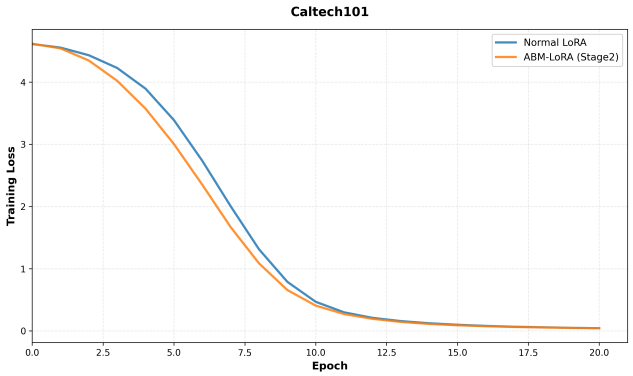
(a) Oxford-IIIT Pet



(b) CIFAR-100



(c) SUN397



(d) Caltech101

Figure A.3. **Training loss curves on VTAB-1K tasks.** We show four examples from the 19-task benchmark to illustrate ABM-LoRA's advantages in achieving lower initial training loss and faster early-stage convergence.

Table A.1. **Ablation study on margin and layer selection for LLaMA2-7B.** We compare different combinations of margin values (m) and the number of layers where ABM initialization is applied. AE = AlpacaEval; LC = length-controlled win rate; WR = win rate. Best results are highlighted in bold.

Margin	# Layers	MT-Bench	AE LC (%)	AE WR (%)
standard LoRA		5.89	42.16	46.27
0.5	12	5.85	42.68	48.26
0.5	16	**5.92**	**45.53**	**49.51**
0.5	32	5.91	44.46	49.50
1.0	12	5.84	43.44	48.82
1.0	16	5.88	43.06	49.32
1.0	32	5.85	42.40	47.27

layer selection strategies (12, 16, or 32 layers).

E.1. Experimental Setup

All experiments use LLaMA2-7B [29] with the following configuration:

- **Base Model:** `meta-llama/Llama-2-7b-hf`
- **LoRA Configuration:** Rank $r = 8$, LoRA alpha $\alpha = 16$, Dropout rate: 0.1, Task type: `CAUSAL_LM`
- **Training Configuration:** Learning rate: 2×10^{-5}, Number of epochs: 1, Maximum sequence length: 1024 tokens, Micro batch size: 1 (with gradient accumulation of 32), Learning rate scheduler: cosine decay, Warmup ratio: 0.03, Weight decay: 0.0
- **ABM Initialization (Stage 1):** 100 steps, learning rate 2×10^{-4}, margin $m \in \{0.5, 1.0\}$
- **Layer Selection:** Three strategies are evaluated:
  - 12 layers: mid-level layers (layers 11-23)
  - 16 layers: latter half (layers 16-31)
  - 32 layers: all decoder layers

E.2. Results and Analysis

Tab. A.1 summarizes the impact of margin values and layer selection strategies for ABM initialization on LLaMA2-7B (rank = 8, dropout = 0.1) under strict dialogue evaluation metrics. Across all six configurations, ABM-LoRA consistently improves over the vanilla LoRA baseline. The best performance is achieved with margin $m = 0.5$ and 16 layers, raising the MT-Bench score from 5.89 to 5.92

Table A.2. **VTAB-1K ablation study results** with ViT-B/16 backbone on 19 diverse visual recognition tasks. Layer: F6=first_6, L6=last_6, A12=all_12. Weight: U=uniform, Q=quadratic.

				Natural							Specialized				Structured								
Layer	Steps	M	W	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dst	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Overall Mean
F6	500	0.5	U	71.0	92.2	62.3	98.2	88.3	84.5	50.3	84.6	95.4	78.9	71.7	81.2	61.9	43.9	77.4	85.6	49.8	28.4	42.6	70.9
F6	500	0.5	Q	71.0	92.6	63.1	98.4	88.4	83.7	50.6	83.7	95.2	77.7	72.4	80.5	62.7	44.8	76.8	86.2	49.8	30.6	42.7	71.1
F6	500	1.0	U	71.2	92.9	63.4	98.1	87.2	84.1	50.0	85.2	95.3	76.6	72.3	79.3	62.4	46.0	76.2	85.8	49.4	30.1	42.4	70.9
F6	500	1.0	Q	71.7	93.4	62.3	98.0	88.1	84.5	50.5	85.0	95.1	78.4	71.7	80.4	62.8	45.8	77.4	85.6	50.9	30.2	42.9	71.3
F6	500	2.0	U	70.2	92.2	62.4	98.4	88.4	83.6	47.9	83.9	95.6	77.4	71.5	79.9	62.0	46.9	74.1	85.9	50.5	31.4	38.0	70.5
F6	500	2.0	Q	72.3	92.1	62.2	98.0	88.2	83.6	49.6	84.7	95.2	76.3	71.9	80.1	62.0	46.4	74.8	85.5	50.8	31.2	42.2	70.9
F6	1000	0.5	U	69.9	92.3	63.2	98.1	87.0	85.0	49.5	84.6	95.3	77.2	71.8	80.3	62.4	45.2	76.8	84.5	49.0	29.8	42.8	70.8
F6	1000	0.5	Q	70.9	92.1	63.1	98.3	86.9	84.3	49.5	85.1	95.5	77.1	71.8	80.2	62.4	45.7	75.7	85.3	48.0	28.0	42.7	70.7
F6	1000	1.0	U	71.2	92.0	61.2	97.5	87.2	85.2	50.0	84.3	95.5	76.2	73.2	78.9	62.2	45.8	75.0	85.3	48.0	31.1	41.2	70.6
F6	1000	1.0	Q	71.0	92.8	62.1	98.2	86.6	84.8	49.3	85.3	95.4	78.5	73.0	80.8	63.2	45.9	78.9	85.4	49.0	30.8	43.2	71.3
F6	1000	2.0	U	71.5	91.3	63.1	97.3	87.7	84.7	48.8	84.7	95.6	77.6	72.0	79.4	62.1	48.1	74.7	84.9	50.3	33.5	40.3	70.9
F6	1000	2.0	Q	71.3	93.0	63.8	97.8	87.9	84.6	48.5	83.8	95.6	76.9	71.5	79.3	62.9	47.5	77.1	85.4	50.2	31.6	41.6	71.1
L6	500	0.5	U	72.6	92.8	63.2	98.1	87.7	84.1	51.4	85.0	95.7	78.2	71.5	81.5	63.0	47.2	77.5	86.0	52.1	31.2	45.2	71.8
L6	500	0.5	Q	74.2	92.8	65.6	97.4	87.6	84.8	51.4	85.5	95.2	77.8	72.3	81.5	63.6	47.6	74.5	83.4	51.8	29.1	44.4	71.6
L6	500	1.0	U	72.8	93.0	63.0	98.1	88.0	85.4	50.5	85.5	95.0	78.7	71.7	81.4	62.9	47.4	75.5	86.0	51.8	29.8	43.6	71.6
L6	500	1.0	Q	73.1	93.0	64.7	98.0	87.3	85.3	50.4	84.3	95.5	78.5	70.9	81.1	64.7	47.3	76.5	84.5	49.8	29.8	44.1	71.5
L6	500	2.0	U	72.1	92.6	64.2	97.7	87.6	85.5	49.6	85.0	95.4	77.9	72.4	80.4	63.0	46.4	76.1	86.5	51.1	29.9	43.9	71.4
L6	500	2.0	Q	73.8	93.0	64.2	97.7	87.7	84.8	49.7	85.0	95.1	78.8	72.0	81.6	64.0	47.4	75.5	85.1	50.6	31.9	44.7	71.7
L6	1000	0.5	U	72.2	93.3	62.7	98.6	87.8	84.8	49.1	87.0	95.5	77.9	72.2	81.4	62.8	46.8	77.5	86.4	51.9	31.4	43.5	71.7
L6	1000	0.5	Q	72.7	92.8	63.4	98.3	87.5	85.3	50.5	85.2	95.6	76.8	72.6	82.1	63.8	47.2	73.7	85.7	51.4	31.9	44.8	71.6
L6	1000	1.0	U	71.8	92.4	63.6	97.8	87.2	84.6	49.4	85.1	95.2	77.9	72.5	81.2	62.8	47.4	74.4	86.5	50.6	29.8	42.6	71.2
L6	1000	1.0	Q	71.8	93.1	65.0	97.8	88.1	84.5	49.3	83.9	95.5	77.8	72.5	81.7	63.5	46.5	77.2	85.5	51.3	30.4	44.3	71.6
L6	1000	2.0	U	70.5	92.3	62.9	97.7	85.6	85.6	49.4	85.2	95.6	77.6	72.3	81.6	62.9	47.0	72.3	86.1	50.3	29.8	43.2	70.9
L6	1000	2.0	Q	72.4	92.7	63.4	97.2	87.4	84.6	48.7	85.7	95.5	78.4	72.7	81.4	63.3	46.8	77.8	85.9	51.1	30.3	44.0	71.5
A12	500	0.5	U	71.9	93.8	62.9	97.8	87.2	83.1	50.3	85.7	95.9	78.4	72.0	81.2	63.0	45.7	75.2	85.7	50.5	28.5	44.4	71.2
A12	500	0.5	Q	72.2	93.0	62.9	98.5	87.8	85.1	51.0	85.7	95.4	78.0	72.1	81.8	62.4	46.5	76.1	85.8	50.8	29.6	44.5	71.5
A12	500	1.0	U	72.6	91.7	63.3	98.0	86.9	86.0	50.5	85.2	95.2	77.5	72.9	81.7	62.6	47.1	76.1	86.5	51.4	29.9	43.3	71.5
A12	500	1.0	Q	72.9	91.9	62.9	98.0	86.6	84.6	50.3	84.2	95.6	78.5	72.0	82.0	63.2	46.4	77.5	85.1	50.1	30.6	43.8	71.4
A12	500	2.0	U	72.9	92.0	62.9	97.8	87.5	85.5	50.1	85.2	95.8	77.4	72.0	80.1	62.1	47.2	76.8	85.4	52.0	31.2	44.5	71.5
A12	500	2.0	Q	73.1	93.0	63.4	98.1	87.8	85.2	49.1	85.3	95.2	78.3	72.3	81.5	62.6	45.7	74.5	85.8	51.2	30.1	43.8	71.4
A12	1000	0.5	U	71.5	92.8	63.0	98.2	86.8	84.4	49.2	85.3	95.4	77.0	71.9	81.2	63.6	46.9	75.2	85.7	50.8	30.7	45.2	71.3
A12	1000	0.5	Q	72.7	93.6	63.3	98.2	87.5	84.3	49.2	85.5	95.0	78.4	72.8	81.2	63.3	46.8	78.1	86.4	52.0	30.9	43.2	71.7
A12	1000	1.0	U	71.3	92.7	61.8	97.8	86.6	85.0	49.4	84.8	95.8	77.2	72.0	80.5	63.3	46.6	75.8	86.5	50.6	30.1	44.2	71.2
A12	1000	1.0	Q	71.6	93.7	62.3	97.8	87.6	85.7	50.1	85.4	95.4	78.8	72.2	81.0	63.1	46.6	77.1	86.6	50.6	31.1	44.3	71.6
A12	1000	2.0	U	71.8	92.7	62.8	97.2	87.6	85.4	49.8	85.0	95.4	78.2	72.1	80.6	62.5	46.8	77.8	85.4	49.9	32.3	43.1	71.4
A12	1000	2.0	Q	71.5	93.2	63.4	97.4	87.5	85.8	48.8	85.8	95.6	78.7	71.5	81.3	63.1	45.9	77.2	85.8	50.4	31.2	43.8	71.5

(+0.03), boosting the length-controlled win rate (LC) from 42.16% to 45.53% (+3.37 pp), and increasing the overall win rate (WR) from 46.27% to 49.51% (+3.24 pp). This demonstrates that aligning adapters via Activation Boundary Matching yields more informative gradient updates right from the start, translating into consistent gains across all metrics.

Examining the effect of margin values, we find that smaller margin ($m = 0.5$) consistently outperforms larger margin ($m = 1.0$) across all layer configurations, suggesting that tighter boundary constraints better preserve the structural knowledge encoded in the pretrained model. Regarding layer selection, applying ABM to 16 layers (the deeper half of the model) yields the strongest results, while both mid-level layers only (12 layers, spanning 11-23) and all 32 layers show diminished gains. Using only mid-level layers may miss critical structural information in the deepest layers, whereas applying ABM to all 32 layers can overconstrain the adapter, reducing its capacity for task-specific adaptation. This finding confirms that selective layer matching—focusing on the latter half of the model—provides the optimal balance between preserving pretrained knowledge and enabling efficient task-specific fine-tuning.

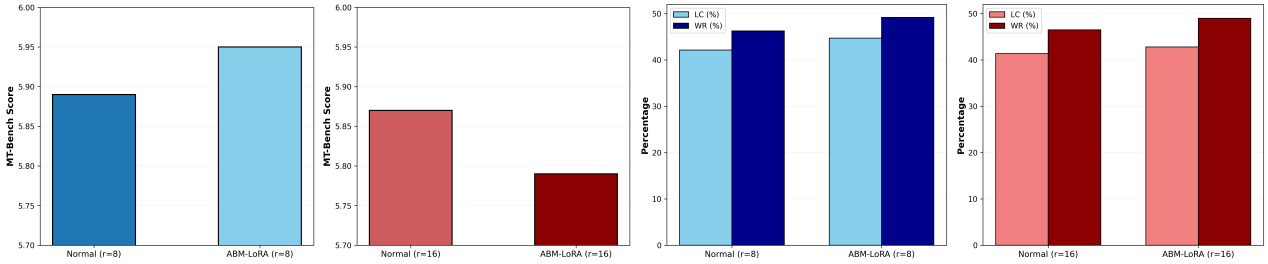
F. Ablation Study on VTAB-1K with ViT-B/16

To systematically evaluate the impact of ABM initialization hyperparameters on vision transformers, we conducted a comprehensive grid search over four key factors on the VTAB-1K benchmark with ViT-B/16: (1) layer selection strategy (first 6, last 6, or all 12 attention layers), (2) margin value $m \in \{0.5, 1.0, 2.0\}$, (3) layer weighting scheme (uniform vs. quadratic), and (4) number of matching steps (500 vs. 1000). This resulted in 36 distinct configurations, each evaluated across all 19 VTAB-1K tasks. Results are presented in Tab. A.2.

F.1. Experimental Setup

All experiments use ViT-B/16 with the following configuration:

- **Base Model:** Pre-trained ViT-B/16 on ImageNet-21k
- **LoRA Configuration:** Rank $r = 8$, LoRA alpha $\alpha = 16$, applied to query and value projections in attention layers
- **Training Configuration:** Learning rate varies by dataset (see main paper), batch size 64, cosine decay scheduler
- **ABM Initialization (Stage 1):** Steps $\in \{500, 1000\}$, learning rate 5×10^{-4}, margin $m \in \{0.5, 1.0, 2.0\}$
- **Layer Selection:** Three strategies are evaluated:



(a) Rank = 8: MT-Bench. (b) Rank = 16: MT-Bench. (c) Rank = 8: LC and WR. (d) Rank = 16: LC and WR.

Figure A.4. **Ablation study by adapter rank (r=8 vs. r=16)** on dialogue performance with dropout fixed at 0.1. (a,b) MT-Bench scores with zoomed y-axis to highlight small differences. (c,d) Length-controlled win rate (LC) and win rate (WR) percentages.

  - F6 (first_6): First 6 attention layers (layers 0-5)
  - L6 (last_6): Last 6 attention layers (layers 6-11)
  - A12 (all_12): All 12 attention layers
- **Layer Weighting:** Uniform (U) assigns equal weight to all layers; Quadratic (Q) uses $w_l = \left(\frac{l+1}{L}\right)^2$ to emphasize deeper layers

F.2. Layer Selection Strategy

Examining the overall mean accuracy across all 19 VTAB-1K tasks, we observe that L6 (last 6 layers) consistently achieves the highest performance, with the best configuration reaching 71.8%. Specifically, among the top-performing configurations:

- L6 configurations: 70.9–71.8% (7 out of top 10 configurations)
- A12 configurations: 71.2–71.7%
- F6 configurations: 70.5–71.3%

This finding aligns with our observations on LLaMA2-7B, where applying ABM to the latter half of layers (16 out of 32) yielded optimal results. The superiority of L6 suggests that deeper layers, which capture more abstract and task-specific representations, benefit most from activation boundary alignment. Applying ABM to all 12 layers (A12) shows comparable but slightly lower performance, likely due to over-constraining the adapter's capacity for task-specific adaptation in early layers. Meanwhile, focusing only on the first 6 layers (F6) misses critical structural information encoded in deeper layers, resulting in consistently lower accuracy.

F.3. Margin Value Analysis

Across all layer selection strategies, margin $m = 0.5$ consistently outperforms larger margins. For L6 configurations with 500 steps:

- $m = 0.5$: 71.6-71.8% (both U and Q weighting)
- $m = 1.0$: 71.5-71.6%
- $m = 2.0$: 71.4-71.7%

The tighter margin ($m = 0.5$) enforces stricter activation boundary alignment, better preserving the pretrained model's learned feature partitioning. Larger margins allow more deviation from the pretrained boundaries, which can reduce the effectiveness of the initialization. This trend holds across F6 and A12 configurations as well, confirming that $m = 0.5$ is the most reliable choice for vision transformers.

F.4. Impact of Matching Steps and Layer Weighting

Comparing 500 vs. 1000 matching steps, we observe marginal differences in overall performance. For the best-performing L6 configurations with $m = 0.5$:

- 500 steps: 71.6-71.8%
- 1000 steps: 71.6-71.7%

This suggests that 500 steps are sufficient for effective activation boundary alignment on VTAB-1K, and doubling the matching steps provides minimal additional benefit while increasing computational cost.

Regarding layer weighting schemes, both uniform (U) and quadratic (Q) weighting achieve competitive results, with no clear winner across all configurations. For L6 with $m = 0.5$:

- 500 steps, U: 71.8%
- 500 steps, Q: 71.6%
- 1000 steps, U: 71.7%
- 1000 steps, Q: 71.6%

While quadratic weighting emphasizes deeper layers more heavily, the uniform scheme performs equally well, likely because restricting ABM to the last 6 layers already focuses on the most important layers for adaptation.

F.5. Task Group Analysis

Breaking down performance by VTAB-1K task categories reveals where ABM initialization provides the greatest benefits:

Natural Tasks: All configurations achieve competitive performance (77.7-79.1% mean), as these tasks align closely with ImageNet pretraining and require minimal adaptation. The choice of ABM hyperparameters has limited impact here.

Specialized Tasks: L6 configurations maintain strong performance (82.3-83.1%), with configuration (c) achieving 87.0% on Camelyon. These medical and satellite imaging tasks benefit moderately from proper initialization.

Structured Tasks: This is where ABM initialization shows the most substantial gains. The best L6 configurations achieve 60.1-60.5% structured mean, compared to 58.7% for vanilla LoRA (main paper Table 3). Notable improvements among the **top-performing L6 configurations** include:

- sNORB-Ele: **43.5–45.2%** vs. 39.2% vanilla LoRA
- dSpr-Ori: **50.6–52.1%** vs. 49.5%
- Clevr-Count: **81.4–82.1%** vs. 79.9%

These structured reasoning tasks—involving 3D viewpoint prediction, spatial localization, and object counting—require geometric understanding that benefits significantly from activation boundary alignment. The pretrained model's learned feature space partitioning for spatial relationships is better preserved with ABM initialization.

F.6. Selection of Configurations for Main Paper

Based on this comprehensive ablation study, we selected four L6 configurations with $m = 0.5$ for inclusion in the main paper (Table 3):

- (a) L6, 500 steps, uniform weighting: 71.8% (highest overall)
- (b) L6, 500 steps, quadratic weighting: 71.6%
- (c) L6, 1000 steps, uniform weighting: 71.7%
- (d) L6, 1000 steps, quadratic weighting: 71.6%

These configurations represent the optimal region of the hyperparameter space, demonstrating that focusing ABM initialization on the last 6 layers with margin 0.5 consistently yields the best vision transformer adaptation performance across diverse visual recognition tasks.

G. Impact of Adapter Rank on LLaMA2-7B

In this ablation study with dropout fixed at 0.1, we compare adapter ranks of 8 and 16 across three evaluation metrics—MT-Bench score, length-controlled win rate (LC), and win rate (WR). The experiments are conducted under identical settings except for the adapter rank. As shown in Fig. A.4, the blue-toned bars correspond to the rank-8 configurations and the red-toned bars to the rank-16 configurations. For the rank-8 setup, replacing standard LoRA initialization with ABM-LoRA improves the MT-Bench score from 5.89 to 5.95, raises LC from 42.16% to 44.74%, and increases WR from 46.27% to 49.19%. At rank 16, ABM-LoRA underperforms on MT-Bench—dropping from 5.87 to 5.79, yet still boosts the AlpacaEval metrics, with LC climbing from 41.37% to 42.81% and WR from 46.52% to 49.00%. This divergence suggests that while activation boundary matching continues to enhance simple win-rate judgments, it may begin to overfit conversational patterns as rank grows, slightly compromising the broader multi-turn coherence captured by MT-Bench even as pairwise preference metrics improve. Therefore, careful adjustment of Stage1 steps and hyperparameters according to the rank size is necessary to maintain overall performance balance.