NOT ALL ATTENTION HEADS HELP: SPECTRAL ATTENTION HEAD SELECTION FOR LOW-RANK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) enables efficient adaptation of pretrained large language models by parameterizing a weight matrix as $\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}$, where only the low-rank factors $\boldsymbol{A}$ and $\boldsymbol{B}$ are optimized. Existing LoRA variants mainly improve the initialization, or optimization process of these low-rank factors, but rarely examine how LoRA is optimized when applied to multi-head attention. An attention projection $\boldsymbol{W}_0$ contains head-specific blocks, while LoRA applies one low-rank update to the entire projection. The factor $\boldsymbol{B}$ can be divided into head-specific blocks, whereas $\boldsymbol{A}$ is shared and updated by the sum of gradient contributions from all heads. The squared norm of this aggregated gradient contains cross-head terms, so the effect of each head depends on how its gradient combines with the others. This cross-head coupling motivates us to investigate selective aggregation for updating $\boldsymbol{A}$. In this paper, we propose a spectral head selection method that selects which heads contribute to the update of $\boldsymbol{A}$, while keeping all $\boldsymbol{B}$ trainable. Specifically, we first derive a reference subspace that preserves the maximum dense gradient energy across heads, and use spectral clustering to select an initial head set based on how much of its gradient lies in this subspace. To maintain effective selection throughout training, we further develop a lightweight adaptive mechanism that refines the selected head set without repeatedly performing costly spectral clustering. Our method introduces no additional parameters or inference overhead and can be integrated with various LoRA variants. Experiments on NLU and NLG tasks across multiple models, together with ablation studies, consistently demonstrate the effectiveness of our method.

1 INTRODUCTION

Parameter-efficient fine-tuning (PEFT) (Hu et al., 2022; Liu et al., 2022; Li & Liang, 2021; Liu et al., 2024b) has become a standard paradigm for adapting large pretrained models under limited computation and storage. Among PEFT methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is widely used. Given a pretrained weight matrix $\boldsymbol{W}_0$, LoRA represents its adaptation as $\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}$, and optimizes only the low-rank factors $\boldsymbol{A}$ and $\boldsymbol{B}$, substantially reducing the number of trainable parameters. In Transformer models, LoRA is commonly applied to the projection matrices of multi-head attention. Its effectiveness therefore depends on how these factors are optimized.

Existing LoRA variants mainly improve low-rank adaptation through rank allocation, initialization, and factor design (Zhang et al., 2023b; Meng et al., 2024; Liu et al., 2024a). These methods focus on the low-rank factors themselves, while how LoRA combines update signals from different heads when applied to attention projection and whether the original function is effective have received little attention. Prior work has shown that different attention heads learn different functions (Voita et al., 2019; Chen et al., 2022; Gui & Xiao, 2023). This makes it important to examine how these head-wise signals jointly update the LoRA factors.

To make this explicit, a multi-head attention projection is implemented as one matrix and can be divided into head-specific blocks. In full fine-tuning, each head-specific block is updated by its own gradient, without sharing parameters across heads. This is different in LoRA. When applying LoRA to the projection matrix, the adaptation can be denoted as $\Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}$, where $\boldsymbol{B}$ can be divided into

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.

head-specific blocks $\boldsymbol{B}_h$, where h denotes the h-th attention head.while $\boldsymbol{A}$ is shared by all heads. Each $\boldsymbol{B}_h$ is still updated by the gradient of its own head, but $\boldsymbol{A}$ is updated by the sum of gradients from all heads: $\nabla_{\boldsymbol{A}}\mathcal{L} = \sum_{h=1}^{H} \boldsymbol{B}_h^\top \boldsymbol{G}_h$, where $\boldsymbol{G}_h$ denotes the corresponding block of the dense gradient and H is the number of heads. Thus, LoRA makes different heads share $\boldsymbol{A}$, which does not happen in full fine-tuning.

The effect of all-head update depends on how the head gradients combine. Let $\boldsymbol{g}_A = \sum_{h=1}^{H} \boldsymbol{g}_h$, where $\boldsymbol{g}_h = \boldsymbol{B}_h^\top \boldsymbol{G}_h$. The resulting gradient energy for $\boldsymbol{A}$ is

$$\|\boldsymbol{g}_A\|_F^2 = \sum_{h=1}^{H} \|\boldsymbol{g}_h\|_F^2 + 2 \sum_{i<j} \langle \boldsymbol{g}_i, \boldsymbol{g}_j \rangle . \tag{1}$$

The first term sums the squared norms of individual head contributions, while the cross terms describe how these contributions reinforce or offset one another. Consequently, the energy depends not only on individual head contributions but also on how they combine.

This decomposition reveals cross-head coupling in the gradient of A, motivating us to investigate improving adaptation via selectively aggregating head contributions instead of unconditional accumulation. We therefore propose a head selection strategy that modifies the gradient supplied to A, while retaining the standard updates of all B_h. We first derive a rank-r reference subspace that preserves the maximum dense gradient energy across all heads, and then use spectral clustering to select an initial head set whose summed gradient has high energy in this subspace. As gradients change during training and continuously performing the procedure above may be expensive, we periodically refine the selected set with a lightweight adaptive selection. Our method adds no trainable parameters and can be used with different LoRA variants.

Experiments on NLU and NLG tasks across multiple models show that our method improves LoRA and its variants. Ablation studies verify the contribution of both the initial spectral clustering and adaptive selection stages. Further analyses support our design choices and meanwhile show the effectiveness of our component and low training overhead.

2 Related Work

LoRA and its variants. LoRA represents the update to a frozen pretrained weight with two trainable low-rank factors (Hu et al., 2022). Existing variants improve its low-rank adaptation from several perspectives. PiSSA improves *initialization* by initializing the low-rank factors with the principal singular components of the pretrained weight, providing a more informative starting subspace than standard random initialization (Meng et al., 2024); related methods similarly explore gradient- or decomposition-aware initialization (Wang et al., 2024b; Yang et al., 2024; Wang et al., 2025; Zhang et al., 2025; He et al., 2025; Ding et al., 2026). AdaLoRA focuses on *adaptive rank allocation*, dynamically distributing a limited rank budget across weight matrices according to their importance (Zhang et al., 2023b), alongside other rank-adaptive approaches such as DyLoRA, SoRA and IncreLoRA (Valipour et al., 2023; Ding et al., 2023; Zhang et al., 2023a). DoRA instead modifies the *weight reparameterization* by decomposing the pretrained weight into magnitude and directional components and applying LoRA to the directional update (Liu et al., 2024a), with subsequent methods further extending this decomposition-based design (Wang et al., 2024a; Nasiri & Garraghan, 2025). Despite these advances, existing LoRA variants pay little attention to what happens when LoRA is applied to a multi-head attention projection. In this setting, $\boldsymbol{A}$ is shared and updated by gradient contributions from all heads and the gradient may be not effective. Our method focuses on how to better update this shared factor and can be combined with existing LoRA variants.

Studies on attention heads. Previous studies have shown that different attention heads play different roles in Transformer models. Clark et al. (2019b) and Kovaleva et al. (2019) analyzed attention patterns and revealed that different heads capture distinct types of information, while several works (Voita et al., 2019; Michel et al., 2019) showed that only a subset of heads are critical and others can be selectively pruned. These findings have motivated head-aware adaptation methods. Chen et al. (2022) and Gui & Xiao (2023) exploited head-level information to improve parameter-efficient adaptation, while Fu et al. (2026) studied how to maintain diverse attention behaviors during training. These studies demonstrate the importance of attention head structures in large language models. However, they mainly focus on head importance, selection, or functional analysis, while how to better applying LoRA to attention head structures has less been explored. We aim to improve LoRA in

such situation, i.e., selecting which head gradients contribute to the update of the shared factor $\boldsymbol{A}$. We further discuss the relationship between head importance and our method in Appendix C.4.

3 METHOD

Considering the phenomenon of applying LoRA to multi-head attention, our goal is to improve adaptation by selecting the head gradients supplied to the shared LoRA factor $\boldsymbol{A}$. We first decompose its gradient into headwise contributions, and then construct a reference subspace from the dense gradient and use projected gradient energy to select an initial head set. During training, we refine this set using a subspace-retention criterion and update it with a lightweight local search. Selection changes only the gradient supplied to $\boldsymbol{A}$, all $\boldsymbol{B}_h$ remain trainable.

3.1 PRELIMINARY

Headwise optimization in multi-head attention. We first describe the query projection in multi-head attention, the same discussion can be applied to the key and value projections. Given an input representation $\boldsymbol{X} \in \mathbb{R}^{N\times n}$, where n is the input dimension, the query projection is typically implemented using a single weight matrix

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}_1^\top & \cdots & \boldsymbol{W}_H^\top\end{bmatrix}^\top, \qquad \boldsymbol{W}_h \in \mathbb{R}^{d_h\times n}, \tag{2}$$

where H is the number of attention heads and $\boldsymbol{W}_h$ corresponds to the h-th attention head. The projected queries can therefore be written as $\boldsymbol{Q}_h = \boldsymbol{X}\boldsymbol{W}_h^\top$. Although the headwise weight matrices are concatenated for efficient implementation, each block $\boldsymbol{W}_h$ is used independently by its corresponding attention head during the forward pass and receives its own gradient signal in the backward pass. Let $\mathcal{L}$ denotes the training loss, the dense gradient of the full projection matrix is the concatenation of headwise gradients, and each head block is updated independently:

$$\boldsymbol{G} = \nabla_{\boldsymbol{W}}\mathcal{L} = \begin{bmatrix}\boldsymbol{G}_1^\top & \cdots & \boldsymbol{G}_H^\top\end{bmatrix}^\top, \qquad \boldsymbol{W}_h \leftarrow \boldsymbol{W}_h - \eta\boldsymbol{G}_h. \tag{3}$$

Thus, although the head blocks are stored in one matrix, full fine-tuning does not force them to share parameters and they can be updated independently.

Cross-head sharing introduced by LoRA. LoRA keeps the pretrained projection matrix $\boldsymbol{W}$ frozen and parameterizes its update function via two low-rank factor $\boldsymbol{A}$ and $\boldsymbol{B}$ as

$$\Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}, \qquad \boldsymbol{A} \in \mathbb{R}^{r\times n}, \qquad \boldsymbol{B} \in \mathbb{R}^{(H\cdot d_h)\times r}, \tag{4}$$

where r is the LoRA rank. Given the dense gradient $\boldsymbol{G}$, the gradients of the two LoRA factors are $\nabla_{\boldsymbol{B}}\mathcal{L} = \boldsymbol{G}\boldsymbol{A}^\top$, and $\nabla_{\boldsymbol{A}}\mathcal{L} = \boldsymbol{B}^\top\boldsymbol{G}$. Further consider the multi-head attention structure, the gradient of $\boldsymbol{B}$ is $\nabla_{\boldsymbol{B}}\mathcal{L} = \boldsymbol{G}\boldsymbol{A}^\top = \begin{bmatrix}\boldsymbol{A}\boldsymbol{G}_1^\top \cdots \boldsymbol{A}\boldsymbol{G}_H^\top\end{bmatrix}^\top$, $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{B}_H^\top\end{bmatrix}^\top$, where $\boldsymbol{B}_h \in \mathbb{R}^{d_h\times r}$ represents the head-wise factor and its gradient $\boldsymbol{G}_h\boldsymbol{A}^\top$ can also be split head-wise from the entire matrix; while the gradient of $\boldsymbol{A}$ is

$$\nabla_{\boldsymbol{A}}\mathcal{L} = \boldsymbol{B}^\top\boldsymbol{G} = \sum\nolimits_{h=1}^{H}\boldsymbol{B}_h^\top\boldsymbol{G}_h. \tag{5}$$

Thus, each $\boldsymbol{B}_h$ receives a head-specific gradient, whereas the shared factor $\boldsymbol{A}$ receives contributions from all heads. Defining the gradient of head h to the shared-factor gradient as $\boldsymbol{g}_h = \boldsymbol{B}_h^\top\boldsymbol{G}_h$, the gradient used to update $\boldsymbol{A}$ becomes $\boldsymbol{g}_A = \sum_{h=1}^{H}\boldsymbol{g}_h$. Unlike the independent block updates in Equation 3, the update of $\boldsymbol{A}$ combines the contributions of all heads. Its squared norm can be written as $\|\boldsymbol{g}_A\|_F^2 = \sum_{h=1}^{H}\|\boldsymbol{g}_h\|_F^2 + 2\sum_{i<j}\langle\boldsymbol{g}_i, \boldsymbol{g}_j\rangle$. These cross terms characterize interactions among head contributions to the shared gradient, indicating that the energy of the aggregated gradient depends not only on the strength of individual head contributions but also on how they combine. We therefore investigate selective aggregation as an alternative update strategy and further evaluate its effectiveness on downstream adaptation.

For a selected set $\mathbb{S} \subseteq \mathbb{H} = \{1, \ldots, H\}$, we define $\boldsymbol{g}_{\mathbb{S}} = \sum_{h\in\mathbb{S}}\boldsymbol{g}_h$, and use $\boldsymbol{g}_{\mathbb{S}}$ to update $\boldsymbol{A}$. Importantly, this selection acts only on the gradient supplied to the shared factor $\boldsymbol{A}$. All head-specific factors $\boldsymbol{B}_h$, including those associated with heads outside $\mathbb{S}$, remain trainable and are updated normally. Our problem is therefore how to find a head set whose summed gradient provides a better update for $\boldsymbol{A}$. To do so, we first identify the gradient directions that the row space of $\boldsymbol{A}$ should preserve, and then compare candidate head sets against this reference.

3.2 REFERENCE-GUIDED SPECTRAL CLUSTERING

We first consider how to evaluate the quality of $\boldsymbol{A}$. The row space of $\boldsymbol{A}$ determines the input directions represented by the shared factor, its orthogonal projector is $\boldsymbol{P} = \boldsymbol{A}^\top \left(\boldsymbol{A}\boldsymbol{A}^\top\right)^\dagger \boldsymbol{A}$, where $\dagger$ denotes the Moore–Penrose pseudoinverse. Following the subspace perspective on low-rank adaptation(Ding et al., 2026; Hao et al., 2024; Zhang & Pilanci, 2024), we use dense-gradient energy retention as a criterion to measure the subspace quality:

$$\Phi(\boldsymbol{P}) = \|\boldsymbol{G}\boldsymbol{P}\|_F^2 = \sum\nolimits_{h=1}^{H} \|\boldsymbol{G}_h\boldsymbol{P}\|_F^2 \,. \tag{6}$$

Here, $\boldsymbol{G}$ represents the dense gradient at the current training step. A larger $\Phi(\boldsymbol{P})$ means that the subspace retains more energy from this gradient. Based on the criterion, we first construct a reference subspace that maximizes Equation 6, and then use it to evaluate aggregated head gradients for initializing the selected head set.

Optimal shared gradient subspace. Following Equation 6, we first find the rank-r subspace that preserves the most current dense gradient energy across all heads. It is given by

$$\boldsymbol{P}^\star = \underset{\boldsymbol{P}=\boldsymbol{P}^\top,\ \boldsymbol{P}^2=\boldsymbol{P},\ \mathrm{tr}(\boldsymbol{P})=r}{\arg\max} \Phi(\boldsymbol{P}) = \Pi_r(\boldsymbol{M}), \tag{7}$$

where $\boldsymbol{M} = \sum_{h=1}^{H} \boldsymbol{G}_h^\top \boldsymbol{G}_h$ and $\Pi_r(\boldsymbol{M})$ denotes the projector onto the eigenvectors associated with the r largest eigenvalues of the positive-semidefinite matrix $\boldsymbol{M}$. The derivation is in Appendix A.1. Thus, $\boldsymbol{P}^\star$ gives a reference for evaluating head contributions. Importantly, $\boldsymbol{P}^\star$ is constructed from the sum of individual gradient energies $\boldsymbol{G}_h^\top \boldsymbol{G}_h$, rather than from a signed aggregation of headwise gradients in the normal update of $\boldsymbol{A}$.

Given the reference subspace $\boldsymbol{P}^\star$, we define an initialization criterion for a candidate head set $\mathbb{S}$ using the energy of its aggregated gradient projected onto the reference subspace:

$$J(\mathbb{S}) = \|\boldsymbol{g}_{\mathbb{S}}\boldsymbol{P}^\star\|_F^2 \,, \qquad \text{with} \quad \boldsymbol{g}_{\mathbb{S}} = \sum\nolimits_{h\in\mathbb{S}} \boldsymbol{g}_h, \tag{8}$$

The criterion $J(\mathbb{S})$ measures how much of the aggregated gradient from the selected heads lies within the reference subspace. It is therefore used as an initialization criterion for head selection. Note that $J(\mathbb{S})$ and $\Phi(\boldsymbol{P})$ characterize different aspects of the optimization process: $\Phi(\boldsymbol{P})$ evaluates the gradient energy retained by a subspace, while $J(\mathbb{S})$ evaluates the projected energy of a gradient aggregate relative to a reference subspace.

Signed spectral clustering and block search. Directly maximizing Equation 8 requires evaluating all $2^H - 1$ nonempty head subsets. To reduce this search space, we exploit the pairwise terms in the objective. Denote $\boldsymbol{z}_h = \boldsymbol{g}_h\boldsymbol{P}^\star$, then expanding J gives

$$J(\mathbb{S}) = \left\|\sum\nolimits_{h\in\mathbb{S}} \boldsymbol{z}_h\right\|_F^2 = \sum\nolimits_{h\in\mathbb{S}} \|\boldsymbol{z}_h\|_F^2 + \sum\nolimits_{i,j\in\mathbb{S},i<j} 2\langle \boldsymbol{z}_i, \boldsymbol{z}_j\rangle. \tag{9}$$

The interaction between heads i and j is exactly given by the pairwise cross term of the above equation, denoting $w_{ij} = 2\langle \boldsymbol{z}_i, \boldsymbol{z}_j\rangle$. The signs of w_{ij} indicate whether two projected head gradients reinforce or offset each other in the reference subspace. We further perform specatral clustring (Ng et al., 2001), specifically, we construct the symmetric interaction matrix $\boldsymbol{W}_s \in \mathbb{R}^{H\times H}$ with $(\boldsymbol{W}_s)_{ij} = w_{ij}$ and zero diagonal. The normalized signed Laplacian is

$$\boldsymbol{L}_s = \boldsymbol{I} - \boldsymbol{D}_s^{-1/2}\boldsymbol{W}_s\boldsymbol{D}_s^{-1/2}, \qquad (\boldsymbol{D}_s)_{ii} = \sum\nolimits_j |w_{ij}|. \tag{10}$$

The eigenvectors associated with the K smallest eigenvalues of $\boldsymbol{L}_s$ provide a spectral embedding of the heads. Applying K-means to this embedding partitions the heads into K blocks $\{\mathbb{C}_k\}_{k=1}^K$. As heads within each block exhibit similar relations, they can be treated as a joint unit during selection. We therefore employ Equation 8 and search over unions of these blocks:

$$\mathbb{S}_0 = \underset{\mathcal{I}\subseteq\{1,\dots,K\},\ \mathcal{I}\neq\varnothing}{\arg\max} J\left(\bigcup\nolimits_{k\in\mathcal{I}} \mathbb{C}_k\right). \tag{11}$$

This reduces the candidate space from $2^H - 1$ head subsets to $2^K - 1$ block combinations. The resulting $\mathbb{S}_0$ serves as the initial active head set for the subsequent adaptive selection stage. Details of the signed spectral clustering procedure are provided in Appendix B.1.

Algorithm 1 Reference-guided head selection and adaptive updates

Require: Warm-up length t_c; adaptive selection interval T; number of clusters K; total training steps $T_{\max}$
Ensure: Trained LoRA factors $\boldsymbol{A}$ and $\{\boldsymbol{B}_h\}_{h=1}^{H}$

$\mathbb{H} \leftarrow \{1, \dots, H\}$ ▷ set of attention heads
Run LoRA for t_c warm-up steps and collect $\{\boldsymbol{G}_h, \boldsymbol{g}_h\}_{h=1}^{H}$
Compute the reference projector $\boldsymbol{P}^\star$ via Eq. 7 ▷ initial selection
Compute $\{\boldsymbol{z}_h\}_{h=1}^{H}$ and construct the signed Laplacian $\boldsymbol{L}_s$ via Eq. 10
Partition $\mathbb{H}$ into K clusters $\{\mathbb{C}_k\}_{k=1}^{K}$ and initialize the active set $\mathbb{S}_0$ via Eq. 11
$\mathbb{S} \leftarrow \mathbb{S}_0$
for $t \leftarrow t_c + 1, \dots, T_{\max}$ **do**
  if $(t - t_c) \bmod T = 0$ **then** ▷ adaptive selection
    Refine $\mathbb{S}$ via Eqs. 13 and 14
  end if
  Update $\boldsymbol{A}$ and $\{\boldsymbol{B}_h\}_{h=1}^{H}$ via Eq. 15
end for
return $\boldsymbol{A}$ and $\{\boldsymbol{B}_h\}_{h=1}^{H}$

3.3 Structure-Preserving Adaptive Selection

As the dense gradient $\boldsymbol{G}$ changes throughout training, the reference subspace and the preferred head set may also change. However, repeatedly recomputing $\boldsymbol{P}^\star$ and rerunning spectral clustering would be expensive. This raises an question: *how can we efficiently evaluate and update the selected head set during training?* We address this in two steps. First, we use an SGD-style virtual update to evaluate candidate head sets. Second, our empirical analysis shows that the selected set changes gradually during training (see Section 4.4). We therefore restrict the search to a small neighborhood of the current set and periodically refine it through a lightweight add–remove procedure.

Local candidate evaluation. Recall that the quality of a rank-r subspace is measured by Equation 6, during adaptive selection, we evaluate candidate sets by the dense gradient energy retained after an SGD-style virtual update. For a candidate set $\mathbb{S}$, we define $\boldsymbol{A}_{\mathbb{S}}^{+} = \boldsymbol{A} - \eta_A \boldsymbol{g}_{\mathbb{S}}$, where η_A is the learning rate. This virtual update is used only for candidate evaluation and does not use optimizer state. We use the resulting gradient energy retention as a criterion for comparing candidate head sets. The row space induced by $\boldsymbol{A}_{\mathbb{S}}^{+}$ is characterized by $\boldsymbol{P}_{\mathbb{S}}^{+} = (\boldsymbol{A}_{\mathbb{S}}^{+})^\top \left[\boldsymbol{A}_{\mathbb{S}}^{+}(\boldsymbol{A}_{\mathbb{S}}^{+})^\top\right]^\dagger \boldsymbol{A}_{\mathbb{S}}^{+}$. Therefore, we evaluate the quality of a candidate set by $\Phi(\boldsymbol{P}_{\mathbb{S}}^{+})$. Expanding this criterion gives

$$\Phi(\boldsymbol{P}_{\mathbb{S}}^{+}) = \sum\nolimits_{h=1}^{H} \left\|\boldsymbol{G}_h \boldsymbol{P}_{\mathbb{S}}^{+}\right\|_F^2 = \sum\nolimits_{h=1}^{H} \operatorname{Tr}\left(\boldsymbol{g}_{B_h}^{+}(\mathbb{S}) \left[\boldsymbol{A}_{\mathbb{S}}^{+}(\boldsymbol{A}_{\mathbb{S}}^{+})^\top\right]^\dagger \boldsymbol{g}_{B_h}^{+}(\mathbb{S})^\top\right), \tag{12}$$

where $\boldsymbol{g}_{B_h}^{+}(\mathbb{S}) = \boldsymbol{G}_h(\boldsymbol{A}_{\mathbb{S}}^{+})^\top$. The expanded form shows that the criterion can be computed using only low-rank quantities of size $d_h \times r$ and $r \times r$, without explicitly forming the dense gradient matrices $\boldsymbol{G}_h$. Details are provided in Appendix B.2.

Lightweight add–remove refinement. Motivated by the gradual changes in the selected set, at each adaptive selection step, we search only a small neighborhood around the current active set $\mathbb{S}_t$. For each head i in the complement $\mathbb{H} \setminus \mathbb{S}_t$, where $\mathbb{H}$ denotes the set of all attention heads, we construct an insertion candidate $\mathbb{S}_t \cup \{i\}$ by adding that head to the active set. We also include the unchanged set $\mathbb{S}_t$ and select the candidate with the highest virtual-update criterion score:

$$\mathbb{N}_t^{+} = \{\mathbb{S}_t\} \cup \{\mathbb{S}_t \cup i : i \notin \mathbb{S}_t\}, \qquad \mathbb{S}_t^{+} = \arg\max\nolimits_{\mathbb{S} \in \mathbb{N}_t^{+}} \Phi(\boldsymbol{P}_{\mathbb{S}}^{+}). \tag{13}$$

Including $\mathbb{S}_t$ itself ensures that a new head is inserted only when it does not decrease the criterion score $\Phi(\boldsymbol{P}_{\mathbb{S}}^{+})$. After that, we consider keeping $\mathbb{S}_t^{+}$ unchanged or removing one of its members and obtain the active set for subsequent training as

$$\mathbb{N}_t^{-} = \{\mathbb{S}_t^{+}\} \cup \left\{\mathbb{S}_t^{+} \setminus j : j \in \mathbb{S}_t^{+}, |\mathbb{S}_t^{+}| > 1\right\}, \qquad \mathbb{S}_{t+1} = \arg\max\nolimits_{\mathbb{S} \in \mathbb{N}_t^{-}} \Phi(\boldsymbol{P}_{\mathbb{S}}^{+}). \tag{14}$$

Together, the two stages allow the active set to remain unchanged, add or remove one head, or replace an active head with a previously inactive one. In this way, each refinement searches a

local neighborhood using the virtual-update criterion in Equation 12. This method can absolutely be extended to enumerating all possible sets. However, this would cause the complexity to grow exponentially. Such local updates are well suited to the gradual evolution of the preferred head structure observed during training, allowing the selected set to adapt continuously without repeatedly reconstructing it through clustering.

For any current active set $\mathbb{S}$, we modify the gradient supplied to the shared factor $\boldsymbol{A}$ by routing only the gradients from selected heads, while keeping all head-specific factors $\boldsymbol{B}_h$ fully trainable. Denoting the routed gradient by $\widetilde{\nabla}$, the gradients supplied to the optimizer are

$$\widetilde{\nabla}_{\boldsymbol{A}}\mathcal{L} = \sum\nolimits_{h\in\mathbb{S}} \boldsymbol{B}_h^\top \boldsymbol{G}_h, \qquad \nabla_{\boldsymbol{B}_h}\mathcal{L} = \boldsymbol{G}_h \boldsymbol{A}^\top, \quad h = 1, \ldots, H. \tag{15}$$

The complete procedure is summarized in Algorithm 1.

4 Experiments

We organize our experiments around four questions. **(Q1)** Does the proposed head selection strategy improve LoRA and representative LoRA variants while keeping the number of trainable parameters unchanged? **(Q2)** How well does the method generalize across different settings, including different tasks, model backbones, and LoRA ranks? **(Q3)** Which components of the proposed framework contribute to the performance, and what empirical evidence supports our design choices? **(Q4)** What additional computational cost does the proposed method introduce in terms of training time and GPU memory compared with the original methods?

4.1 Experimental Setup

Detailed datasets for tasks. For natural language understanding, we evaluate on following tasks from GLUE (Wang et al., 2019): CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), QQP[1], MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE(Wang et al., 2019). We follow the standard GLUE metrics to evaluate the performance. For natural language generation, we evaluate on mathematical reasoning, code generation, and commonsense reasoning benchmarks. Specifically, we use GSM8K (Cobbe et al., 2021) for math, HumanEval for code, and commonsense benchmarks (Hu et al., 2023), including BoolQ, PIQA, SIQA, WinoGrande, OBQA, ARC-C, and ARC-E. Models are fine-tuned on MetaMathQA (Yu et al., 2024), Code-Feedback (Zheng et al., 2024), and Commonsense-170K (Hu et al., 2023), respectively.

Backbones and baselines. We use RoBERTa (Liu et al., 2019) for GLUE, Llama-3.1-8B (Dubey et al., 2024) and Mistral-7B-v0.1 (Jiang et al., 2023) for generation tasks. We evaluate standard LoRA (Hu et al., 2022) together with three representative variants: PiSSA (Meng et al., 2024), which modifies adapter initialization; AdaLoRA (Zhang et al., 2023b), which dynamically reallocates the rank budget; and DoRA (Liu et al., 2024a), which decomposes weight updates into magnitude and direction. Our methed can be integrated with various LoRA variants, so for each base method, we compare the original formulation with its counterpart augmented by our head selection strategy. Moreover, no additional trainable parameters are introduced.

Implementation. All methods are implemented using HuggingFace Transformers (Wolf et al., 2020) and optimized with AdamW (Loshchilov & Hutter, 2019). Following Hu et al. (2022), LoRA adapters are applied to the query, key and value projections in every attention layer. Each configuration is evaluated over five independent random seeds. Additional implementation and hyperparameter details are provided in Appendix C.

4.2 Main Results

Result analysis on NLU tasks. Table 1 compares each base method with its head-selection variant on GLUE, where "SEL" denotes our method, and the LoRA rank is set to 8. Employing our head selection method improves the corresponding base method across the evaluated datasets, showing

[1] data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Table 1: Test accuracy(%) of RoBERTa with LoRA variants (with/without selection) on GLUE benchmark datasets (rank 8).

Method	SEL	CoLA	SST-2	QQP	MNLI	QNLI	RTE
LoRA	w/o	$58.81_{\pm 0.35}$	$93.80_{\pm 0.37}$	$87.51_{\pm 0.59}$	$85.47_{\pm 0.77}$	$91.19_{\pm 0.68}$	$71.36_{\pm 0.68}$
	w/	$63.68_{\pm 0.80}$ (+4.87)	$95.07_{\pm 0.16}$ (+1.27)	$89.61_{\pm 0.16}$ (+2.10)	$87.35_{\pm 0.23}$ (+1.88)	$92.85_{\pm 0.04}$ (+1.66)	$74.01_{\pm 3.57}$ (+2.65)
PiSSA	w/o	$59.31_{\pm 0.62}$	$94.14_{\pm 0.12}$	$88.17_{\pm 0.70}$	$86.49_{\pm 0.49}$	$91.67_{\pm 0.53}$	$72.68_{\pm 0.45}$
	w/	$61.07_{\pm 0.35}$ (+1.76)	$94.72_{\pm 0.16}$ (+0.58)	$89.83_{\pm 0.06}$ (+1.66)	$87.45_{\pm 0.01}$ (+0.96)	$92.62_{\pm 0.03}$ (+0.95)	$74.91_{\pm 1.28}$ (+2.23)
AdaLoRA	w/o	$58.92_{\pm 0.64}$	$93.92_{\pm 0.08}$	$88.19_{\pm 0.17}$	$86.26_{\pm 0.04}$	$91.83_{\pm 0.07}$	$71.84_{\pm 1.02}$
	w/	$63.56_{\pm 1.40}$ (+4.64)	$94.72_{\pm 0.16}$ (+0.80)	$89.98_{\pm 0.04}$ (+1.79)	$87.35_{\pm 0.01}$ (+1.09)	$92.81_{\pm 0.08}$ (+0.98)	$74.73_{\pm 0.03}$ (+2.89)
DoRA	w/o	$59.07_{\pm 0.69}$	$94.16_{\pm 0.16}$	$88.04_{\pm 0.19}$	$85.97_{\pm 0.13}$	$91.59_{\pm 0.01}$	$71.30_{\pm 0.77}$
	w/	$63.52_{\pm 0.32}$ (+4.45)	$94.67_{\pm 0.24}$ (+0.51)	$89.77_{\pm 0.12}$ (+1.73)	$87.60_{\pm 0.08}$ (+1.63)	$92.79_{\pm 0.18}$ (+1.20)	$76.53_{\pm 0.51}$ (+5.23)

Table 2: Test score(%) of Llama-3.1-8B and Mistral-7B-v0.1 with LoRA variants (with/without selection) on NLG datasets (rank 8).

Method	SEL	Llama-3.1-8B		Mistral-7B-v0.1	
		GSM8K	HumanEval	GSM8K	HumanEval
LoRA	w/o	$70.41_{\pm 0.99}$	$45.12_{\pm 1.80}$	$63.58_{\pm 0.41}$	$39.63_{\pm 0.64}$
	w/	$71.49_{\pm 0.87}$ (+1.08)	$46.75_{\pm 1.25}$ (+1.63)	$65.81_{\pm 0.08}$ (+2.23)	$40.85_{\pm 1.28}$ (+1.22)
PiSSA	w/o	$71.07_{\pm 0.87}$	$45.52_{\pm 0.59}$	$62.04_{\pm 1.57}$	$39.63_{\pm 0.51}$
	w/	$72.25_{\pm 0.40}$ (+1.18)	$47.15_{\pm 2.45}$ (+1.63)	$64.63_{\pm 0.80}$ (+2.59)	$41.46_{\pm 0.19}$ (+1.83)
AdaLoRA	w/o	$67.77_{\pm 0.30}$	$42.31_{\pm 0.73}$	$64.92_{\pm 0.32}$	$39.02_{\pm 0.67}$
	w/	$72.35_{\pm 0.10}$ (+4.58)	$44.92_{\pm 0.29}$ (+2.61)	$65.48_{\pm 0.62}$ (+0.56)	$43.29_{\pm 0.98}$ (+4.27)
DoRA	w/o	$69.70_{\pm 0.64}$	$43.70_{\pm 0.57}$	$63.48_{\pm 0.42}$	$38.41_{\pm 0.34}$
	w/	$72.07_{\pm 0.87}$ (+2.37)	$47.97_{\pm 1.25}$ (+4.27)	$64.85_{\pm 0.36}$ (+1.37)	$40.24_{\pm 0.60}$ (+1.83)

that its benefit extends to different LoRA variants. The gains are particularly noticeable on smaller datasets. For example, the average improvement is nearly 4 points on CoLA and nearly 3 points on RTE. The larger gains on them suggest that how $\boldsymbol{A}$ is updated can be particularly important when training data are limited. Standard LoRA always forms this update by aggregating gradients from all attention heads, whereas our method selects the head gradients used to update $\boldsymbol{A}$ according to the reference subspace and adaptively adjusts the selected set during training. These results indicate that the default all-head aggregation is not always the most effective choice, and selective aggregation for $\boldsymbol{A}$ can improve downstream performance. An additional comparison with LoRA+(Hayou et al., 2024) shows that our method retains an advantage over the factor-specific learning-rate adjustments evaluated in Appendix C.2.

Robustness to Rank. Results for RoBERTa at varying ranks $(4, 8, 16)$ show that these LoRA variants with head selection achieve better performance, highlighting the robustness of head selection. Detailed results and analysis is in Appendix C.1.2.

Result analysis on NLG tasks. Table 2 extends the comparison to GSM8K and HumanEval benchmarks with Llama-3.1-8B and Mistral-7B-v0.1, where the LoRA rank is 8. These tasks test whether the head selection mechanism transfers beyond sentence-level understanding and beyond a single model family. The table still compares the matched base and selection pairs. The results show that LoRA and its variants with head selection outperform their default implement across the datasets, which indicate that the mechanism is not tied to one dataset or model. Results and analysis on commonsense benchmarks are in Appendix C.3.2.

4.3 Ablation Study

The ablation studies in Table 3 examine the roles of spectral clustering initialization and adaptive refinement, and other variants. The specific variants are as follows. **i**. **Standard LoRA**: conventional LoRA (Hu et al., 2022), in which the gradients from all attention heads jointly update the shared factor. **ii**. **Ours**: our full method, which combines spectral clustering initialization with lightweight adaptive refinement during training. Owing to its low cost, the selected set can be refined frequently,

Table 3: Ablation results of RoBERTa on GLUE benchmark datasets(rank 8).

Method	Head Set		GLUE					
	Initialization	Update	CoLA	SST-2	QQP	MNLI	QNLI	RTE
Standard LoRA	All heads	Joint update	58.81	93.80	87.51	85.47	91.19	71.36
Static Selection	Clustering	Fixed	58.34	94.26	88.12	85.89	92.24	71.48
Random Selection	Random	Adaptive refinement	58.81	94.04	88.36	86.05	92.28	71.84
Periodic Re-clustering	Clustering	Re-clustering	63.42	94.49	89.53	86.87	92.75	72.92
Ours	Clustering	Adaptive refinement	**63.68**	**95.07**	**89.61**	**87.35**	**92.85**	**74.01**

and we therefore use a short interval of $T = 50$ steps. **iii**. **Static Selection**: performs the one-time spectral clustering and selection, after which the selected head set remains fixed. **iv**. **Random Selection**: randomly selects the initial active head set with the same size as the clustering-based selection, and then applies the same adaptive refinement procedure and update interval ($T = 50$) as our full method. **v**. **Periodic Re-clustering**: periodically re-clusters the attention heads and reconstructs the selected set accordingly. Since repeated clustering introduces non-negligible computational overhead, we adopt a relatively long update interval of $T = 200$ steps.

Several conclusions can be drawn from the results. **i**. Periodic head-selection variants generally outperform the all-head update baseline, suggesting that periodically updating head selection can improve downstream performance over all-head aggregation in these settings. **ii**. Our full method outperforms Random Selection across the evaluated datasets. Since both variants use the same adaptive refinement procedure and interval, this comparison supports the value of spectral initialization beyond just randomly matching the initial number of selected heads. **iii**. Our full method also outperforms Static Selection, indicating that refining the selected set during training provides benefits beyond the initial selection alone. Together, these comparisons support combining spectral initialization with adaptive refinement. **iv**. Our method achieves a modest but consistent improvement over Periodic Re-clustering under the evaluated update schedules. This supports lightweight local refinement as an effective alternative to repeated clustering.

4.4 Analysis of Adaptive Head Selection

We analyze how the selected head set changes during training and whether the adaptive procedure can follow these changes at low cost. We also examine how different head-selection strategies affect the gradient information retained by the shared LoRA subspace. All experiments are conducted on MNLI using the same training configuration as the main experiments.

Local stability of the selected set. To study how the selected set changes over time, we periodically rerun the spectral-clustering-based global selection and denote the resulting set at step t by $\mathbb{S}_t$. We measure the similarity between sets separated by Δ training steps using the Jaccard index:

$$\mathrm{Sim}_\Delta(t) = \mathrm{Jac}\left(\mathbb{S}_t, \mathbb{S}_{t+\Delta}\right) = |\mathbb{S}_t \cap \mathbb{S}_{t+\Delta}| / |\mathbb{S}_t \cup \mathbb{S}_{t+\Delta}| \tag{16}$$

where $\mathrm{Jac}(\cdot,\cdot)$ denotes the Jaccard similarity and $\Delta \in \{25, 50, 100\}$. As shown in Fig. 1(a), sets separated by 25 or 50 steps remain more similar than those separated by 100 steps. This suggests that the selected head structure retains overlap over short intervals, while larger changes emerge over longer intervals. The relatively high similarity at both 25 and 50 steps motivates using a moderate refinement interval rather than adjusting the selected set at every training step, and provides empirical support for our choice of $T = 50$. Meanwhile, the lower similarity at 100 steps motivates refining the selected set throughout training rather than keeping it fixed.

Tracking changes during training. We next examine whether the proposed Structure-Preserving Adaptive Selection can follow the changes observed above. After performing spectral clustering at an early point t_0, we initialize both a fixed baseline and our adaptive method with the same selected set $\mathbb{S}_{t_0}$. The fixed baseline keeps this set unchanged for the remainder of training, whereas our method refines it every 50 steps using the add–remove procedure in Sec. 3.3, denoting Fixed and Adaptive respectively. For analysis only, we continue to run the global selection procedure at each

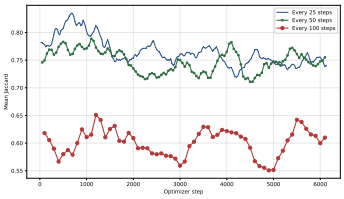
evaluation point and use its output $\mathbb{S}_t$ as a reference. We evaluate the tracking quality of each method by $\mathrm{Jac}\left(\mathbb{S}_t^m, \mathbb{S}_t\right)$, where $m \in \{\mathrm{Fixed}, \mathrm{Adaptive}\}$.

Fig. 1(b) shows that the fixed set remains moderately similar to the reference, consistent with the observation above, but not high. In contrast, the adaptive set stays closer to the periodically recomputed reference than the fixed set, indicating that adaptive selection can effectively follows changes in the selected heads without using periodic spectral clustering to make its updates.

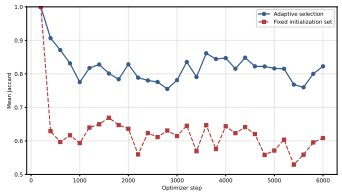
Gradient-energy retention. Finally, we examine how the choice of heads used to update $\boldsymbol{A}$ affects its row space. We initialize the row space of $\boldsymbol{A}$ with the optimal rank-r subspace in Equation 7, so all methods start from the same point. We then compare three methods over this period: updating $\boldsymbol{A}$ with all heads, using a fixed head set obtained from the initial clustering, and adaptively refining this set throughout training. We compute the projector $\boldsymbol{P}$ onto the current row space of $\boldsymbol{A}$ and report the fraction of dense gradient energy retained by this subspace:

$$\mathrm{Capture}(\boldsymbol{P}) = \Phi(\boldsymbol{P}) / \sum_{h=1}^{H} \|\boldsymbol{G}_h\|_F^2, \tag{17}$$

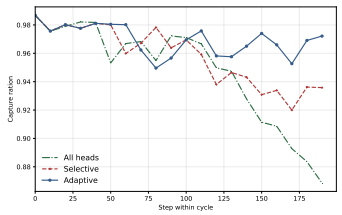
where $\Phi(\boldsymbol{P})$ is evaluated using Equation 6. As shown in Fig. 1(c), the capture ratio of the all-head update drops markedly in the later part of the interval. The fixed selected set retains relatively more energy near the end, while adaptive selection maintains higher gradient energy retention toward the end of the interval, consistent with the criterion used to guide refinement. This analysis examines the subspace property targeted by our criterion, while the downstream experiments evaluate the practical benefit of the resulting selection strategy. Together, these results support the empirical effectiveness of our design.



(a) Temporal similarity.



(b) Tracking similarity.



(c) Captured gradient energy.

Figure 1: Analysis of adaptive head selection.

4.5 Training efficiency.

We compare the training cost of our method with standard LoRA in Table 4. Both methods are trained for one epoch on 100K examples using 8 NVIDIA GeForce RTX 5090 GPUs.

Both methods use the same 4.72M trainable parameters because head selection only changes the gradient used to update $\boldsymbol{A}$ and introduces no additional trainable parameters. Over the one-epoch run, our method takes about 81 minutes, compared with 78 minutes for LoRA, an increase of only 3.8%. During regular training, peak memory increases only slightly. The largest memory usage occurs during the one-time initial selection which temporarily stores the gradient statistics needed to construct the reference subspace and perform spectral clustering, and its memory is released after initialization. The periodic adaptive refinement requires only a small amount of extra GPU memory compared with standard training. These results show that the adaptive procedure remains lightweight, while the more costly global selection is performed only once.

Table 4: Training cost on Llama-3.1-8B. Peak GPU memory (GB) is reported per GPU.

Method	Params	Memory (GB)			Time
		Train	Init.	Refine	
LoRA	4.72M	17.79	–	–	78min
Ours	4.72M	18.01	20.60	18.57	81min

5 Conclusion

In this work, we study how the shared LoRA factor $\boldsymbol{A}$ is updated in multi-head attention and show that selective aggregation can improve downstream performance over all-head aggregation empirically. Our method derives a reference subspace, selects an initial head set via spectral clustering, and efficiently refines it during training. The method adds no trainable parameters and can be applied to various LoRA variants. Experiments on NLU and NLG tasks across multiple models, together with ablation studies, consistently demonstrate its effectiveness.

AI USE STATEMENT

Generative AI tools were used to assist in the writing of proofs, help modify code to implement methods, assist with translation. They were not used for Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We have reviewed all AI-assisted work, including the proof part of the manuscript and the authenticity of the references. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The 34th AAAI Conference on Artificial Intelligenc (AAAI'20)*, pp. 7432–7439, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, and et al. Evaluating large language models trained on code. *arxiv:2107.03374*, 2021.

Yifan Chen, Devamanyu Hazarika, Mahdi Namazifar, Yang Liu, Di Jin, and Dilek Hakkani-Tur. Empowering parameter-efficient transfer learning by recognizing the kernel structure in self-attention. In *Findings of the Association for Computational Linguistics (NAACL'22)*, pp. 1375–1388, July 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NACCL'19)*, pp. 2924–2936, 2019a.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arxiv:2110.14168*, 2021.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*, pp. 4133–4145, 2023.

Shu Ding, Yang Peng, Hangan Zhou, Xinyu Lu, Shangwei Chen, Junhua Huang, Mingxuan Yuan, and Wei Wang. Towards understanding the dynamics of low-rank adaptation. In *Proceedings of the 43th International Conference on Machine Learning (ICML'26)*, 2026.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and et al. The llama 3 herd of models. *arxiv:2407.21783*, 2024.

Zizhuo Fu, Wenxuan Zeng, Runsheng Wang, and Meng Li. Attention sink forges native moe in attention layers: Sink-aware training to address head collapse. In *Proceedings of the 43th International Conference on Machine Learning (ICML'26)*, 2026.

Anchun Gui and Han Xiao. HiFi: High-information attention heads hold for parameter-efficient model adaptation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL'23)*, pp. 8521–8537, 2023.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, 2024.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, 2024.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, and Lei Chen. Gora: Gradient-driven adaptive low rank adaptation. In *Advances in Neural Information Processing Systems 38 (NeurIPS'25)*, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*, pp. 5254–5276, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv:2310.06825*, 2023.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*, pp. 4365–4374, 2019.

Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto W. De Luca, and Sahin Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM'10)*, pp. 559–570, 2010.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP'21)*, pp. 4582–4597, 2021.

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 *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, 2024a.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL'22)*, pp. 61–68, 2022.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *AI Open*, 5:208–215, 2024b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, 2024.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS'19)*, pp. 14014–14024, 2019.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*, pp. 2381–2391, 2018.

Hamid Nasiri and Peter Garraghan. Edora: Efficient weight-decomposed low-rank adaptation via singular value decomposition. *arXiv:2501.12067*, 2025.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 (NIPS'01)*, pp. 849–856, 2001.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, pp. 2383–2392, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pp. 8732–8740, 2020.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*, pp. 4463–4473, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, pp. 1631–1642, 2013.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL'23)*, pp. 3266–3279, 2023.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*, pp. 5797–5808, 2019.

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 *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'25)*, pp. 4823–4836, 2025.

Qiushi Wang, Yuchen Fan, Junwei Bao, Hongfei Jiang, and Yang Song. Bora: Bi-dimensional weight-decomposed low-rank adaptation. *arXiv:2412.06441*, 2024a.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, 2024b.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*, pp. 1112–1122, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP'20)*, pp. 38–45, 2020.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*, 2024.

Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned lora for fine-tuning foundation models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, 2024.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv:2308.12043*, 2023a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*, 2023b.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. LoRA-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. In *Proceedings of the 42th International Conference on Machine Learning (ICML'25)*, 2025.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics (Findings of ACL'24)*, pp. 12834–12859, 2024.

A Detailed derivations and proofs

A.1 Derivation of the optimal shared gradient subspace

Let $\boldsymbol{P}$ be a rank-r orthogonal projector, i.e., $\boldsymbol{P}^\top = \boldsymbol{P}$, $\boldsymbol{P}^2 = \boldsymbol{P}$, and $\mathrm{tr}(\boldsymbol{P}) = r$. For each head,

$$\|\boldsymbol{G}_h \boldsymbol{P}\|_F^2 = \mathrm{tr}\left(\boldsymbol{P}\boldsymbol{G}_h^\top \boldsymbol{G}_h \boldsymbol{P}\right) = \mathrm{tr}\left(\boldsymbol{P}\boldsymbol{G}_h^\top \boldsymbol{G}_h\right),$$

where the second equality follows from $\boldsymbol{P}^2 = \boldsymbol{P}$ and the cyclic property of the trace. Therefore,

$$\Phi(\boldsymbol{P}) = \sum_{h=1}^{H} \|\boldsymbol{G}_h \boldsymbol{P}\|_F^2 = \mathrm{tr}\left(\boldsymbol{P} \sum_{h=1}^{H} \boldsymbol{G}_h^\top \boldsymbol{G}_h\right) = \mathrm{tr}(\boldsymbol{P}\boldsymbol{M}),$$

where $\boldsymbol{M} = \sum_{h=1}^{H} \boldsymbol{G}_h^\top \boldsymbol{G}_h$. The matrix $\boldsymbol{M}$ is positive semidefinite. Let

$$\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \qquad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), \qquad \lambda_1 \geq \cdots \geq \lambda_d \geq 0, \tag{18}$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d]$ is orthogonal. Writing the projector as $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^\top$, with $\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_r$, yields

$$\mathrm{tr}(\boldsymbol{P}\boldsymbol{M}) = \mathrm{tr}(\boldsymbol{Q}^\top \boldsymbol{M}\boldsymbol{Q}) \leq \sum_{i=1}^{r} \lambda_i, \tag{19}$$

where the inequality is the Ky Fan maximum principle. Equality is attained by $\boldsymbol{Q}^\star = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$. Therefore, an optimal projector is

$$\boldsymbol{P}^\star = \boldsymbol{Q}^\star \boldsymbol{Q}^{\star\top} = \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^\top = \Pi_r(\boldsymbol{M}). \tag{20}$$

Here $\Pi_r(\boldsymbol{M})$ denotes any projector onto a principal r-dimensional eigenspace. Finally,

$$\sum_{h=1}^{H} \|\boldsymbol{G}_h \boldsymbol{P}^\star\|_F^2 = \mathrm{tr}(\boldsymbol{P}^\star \boldsymbol{M}) = \sum_{i=1}^{r} \lambda_i, \tag{21}$$

so $\boldsymbol{P}^\star$ captures the maximum total gradient energy.

B Additional Method Details

B.1 Details of signed spectral clustering

This section details the signed spectral clustering procedure used to obtain the head blocks in Section 3.2. The procedure follows normalized spectral clustering (Ng et al., 2001), with the ordinary affinity graph replaced by a signed graph whose absolute edge weights define node degrees, as in signed spectral graph analysis (Kunegis et al., 2010).

Signed head-interaction graph. For every head h, we first project its shared-factor gradient onto the reference subspace,

$$\boldsymbol{z}_h = \boldsymbol{g}_{A,h} \boldsymbol{P}^\star. \tag{22}$$

We regard each head as a node and define the weight of the edge between heads i and j by

$$w_{ij} = \begin{cases} 2\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle, & i \neq j, \\ 0, & i = j. \end{cases} \tag{23}$$

Let $\boldsymbol{W}_s = [w_{ij}]_{i,j=1}^{H}$ and define the absolute degree of node i as

$$d_i = \sum_{j=1}^{H} |w_{ij}|, \qquad \boldsymbol{D}_s = \mathrm{diag}(d_1, \ldots, d_H). \tag{24}$$

Using absolute rather than signed degrees prevents positive and negative edges from canceling in the normalization. The symmetric normalized signed Laplacian is

$$\boldsymbol{L}_s = \boldsymbol{I} - \boldsymbol{D}_s^{-1/2}\boldsymbol{W}_s\boldsymbol{D}_s^{-1/2}. \tag{25}$$

Under nonzero projected gradients, the dense interaction graph generically has $d_i > 0$ for every head. If a head has exactly zero degree, we exclude it from the eigendecomposition and assign it afterward to the currently smallest block.

The construction in equation 25 is positive semidefinite. In particular, for any $\boldsymbol{x} \in \mathbb{R}^H$,

$$\boldsymbol{x}^\top \boldsymbol{L}_s \boldsymbol{x} = \frac{1}{2}\sum_{i,j} |w_{ij}| \left(\frac{x_i}{\sqrt{d_i}} - \operatorname{sgn}(w_{ij}) \frac{x_j}{\sqrt{d_j}} \right)^2 \geq 0, \tag{26}$$

Consequently, minimizing the spectral relaxation places nodes connected by positive edges nearby and nodes connected by negative edges in opposing directions.

Spectral embedding and discretization. Let $\lambda_1 \leq \cdots \leq \lambda_H$ be the eigenvalues of $\boldsymbol{L}_s$, and let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ be orthonormal eigenvectors associated with its K smallest eigenvalues. We form

$$\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K] \in \mathbb{R}^{H \times K}, \qquad \widehat{\boldsymbol{U}}_{h,:} = \frac{\boldsymbol{U}_{h,:}}{\max(\|\boldsymbol{U}_{h,:}\|_2, \epsilon)}, \tag{27}$$

where ϵ is a small numerical constant. Unlike unsigned spectral clustering, the smallest eigenvector is not discarded: a signed graph need not have the constant vector as a trivial eigenvector. We then apply K-means to the H row vectors of $\widehat{\boldsymbol{U}}$. This yields a partition $\{\mathbb{C}_k\}_{k=1}^K$ of the heads.

Exact search over block unions. Clustering only defines the units of the reduced search; the final choice is still made using the original objective J. This search can be evaluated without repeatedly summing individual head gradients. For each block, cache

$$\boldsymbol{q}_k = \sum_{h \in \mathbb{C}_k} \boldsymbol{z}_h, \qquad \boldsymbol{B}_{k\ell} = \langle \boldsymbol{q}_k, \boldsymbol{q}_\ell \rangle. \tag{28}$$

For a nonzero binary vector $\boldsymbol{b} \in \{0,1\}^K \setminus \{\boldsymbol{0}\}$, let $\mathbb{S}(\boldsymbol{b}) = \bigcup_{k:b_k=1} \mathbb{C}_k$. Its score is exactly

$$J(\mathbb{S}(\boldsymbol{b})) = \left\| \sum_{k=1}^K b_k \boldsymbol{q}_k \right\|_F^2 = \boldsymbol{b}^\top \boldsymbol{B} \boldsymbol{b}. \tag{29}$$

We enumerate the $2^K - 1$ nonempty binary vectors and set

$$\boldsymbol{b}^\star = \arg\max_{\boldsymbol{b} \in \{0,1\}^K \setminus \{\boldsymbol{0}\}} \boldsymbol{b}^\top \boldsymbol{B} \boldsymbol{b}, \qquad \mathbb{S}_0 = \mathbb{S}(\boldsymbol{b}^\star). \tag{30}$$

When several unions attain the same score up to numerical precision, we choose the union containing fewer heads and then break any remaining tie lexicographically. This affects only reproducibility and not the objective.

B.2 Efficient computation of the adaptive selection score

A direct evaluation of equation 12 appears to require the full weight gradients $\{\boldsymbol{G}_h\}_{h=1}^H$. We show that the exact score can be computed from LoRA matrices.

Low-rank reformulation. Flatten the batch and sequence dimensions, and let $\boldsymbol{X} \in \mathbb{R}^{N \times n}$ denote the input activations and $\boldsymbol{D}_h \in \mathbb{R}^{N \times d_h}$ the output-gradient signal of head h. Then $\boldsymbol{G}_h = \boldsymbol{D}_h^\top \boldsymbol{X}$, but this matrix need not be materialized. For a candidate set $\mathbb{S}$, define

$$\boldsymbol{A}_{\mathbb{S}}^+ = \boldsymbol{A} - \eta_A \boldsymbol{g}_{\mathbb{S}}, \qquad \boldsymbol{C}_{\mathbb{S}} = \boldsymbol{A}_{\mathbb{S}}^+ (\boldsymbol{A}_{\mathbb{S}}^+)^\top, \tag{31}$$

where $\boldsymbol{g}_{\mathbb{S}} = \sum_{k \in \mathbb{S}} \boldsymbol{g}_h$. The projector after this virtual update is $\boldsymbol{P}_{\mathbb{S}}^+ = (\boldsymbol{A}_{\mathbb{S}}^+)^\top \boldsymbol{C}_{\mathbb{S}}^\dagger \boldsymbol{A}_{\mathbb{S}}^+$.

Concatenate the backward signals as $\boldsymbol{D} = [\boldsymbol{D}_1, \ldots, \boldsymbol{D}_H]$ and define

$$\boldsymbol{U}_{\mathbb{S}} = \boldsymbol{D}^\top \big(\boldsymbol{X}(\boldsymbol{A}_{\mathbb{S}}^+)^\top\big) \in \mathbb{R}^{d\times r}, \qquad d = \sum_{h=1}^{H} d_h. \tag{32}$$

Its h-th row block is exactly $\boldsymbol{g}_{B_h}^+(\mathbb{S}) = \boldsymbol{G}_h(\boldsymbol{A}_{\mathbb{S}}^+)^\top$. Using the Moore–Penrose identity $\boldsymbol{C}_{\mathbb{S}}^\dagger \boldsymbol{C}_{\mathbb{S}} \boldsymbol{C}_{\mathbb{S}}^\dagger = \boldsymbol{C}_{\mathbb{S}}^\dagger$, we obtain

$$\begin{aligned}\Phi(\boldsymbol{P}_{\mathbb{S}}^+) &= \left\| \boldsymbol{U}_{\mathbb{S}} \boldsymbol{C}_{\mathbb{S}}^\dagger \boldsymbol{A}_{\mathbb{S}}^+ \right\|_F^2 \\ &= \mathrm{Tr}\Big(\boldsymbol{C}_{\mathbb{S}}^\dagger \boldsymbol{U}_{\mathbb{S}}^\top \boldsymbol{U}_{\mathbb{S}} \Big).\end{aligned} \tag{33}$$

Thus, equation 33 is an exact algebraic evaluation of the virtual-update score defined in Equation 12. It uses only a $d \times r$ matrix and an $r \times r$ matrix. The identity also holds when $\boldsymbol{A}_{\mathbb{S}}^+$ is rank deficient, because $(\boldsymbol{A}_{\mathbb{S}}^+)^\top \boldsymbol{C}_{\mathbb{S}}^\dagger \boldsymbol{A}_{\mathbb{S}}^+$ is the orthogonal projector onto its row space. Since $\boldsymbol{C}_{\mathbb{S}}$ is only $r \times r$, its exact pseudoinverse can be computed in $O(r^3)$ time.

Reuse across insertion and pruning candidates. Let $\mathbb{S}_t$ be the current active set and cache

$$\bar{\boldsymbol{A}} = \boldsymbol{A} - \eta_A \boldsymbol{g}_{\mathbb{S}_t}, \qquad \bar{\boldsymbol{C}} = \bar{\boldsymbol{A}}\bar{\boldsymbol{A}}^\top, \qquad \bar{\boldsymbol{U}} = \boldsymbol{D}^\top \big(\boldsymbol{X}\bar{\boldsymbol{A}}^\top\big). \tag{34}$$

For each head that may be inserted or removed, define the low-rank correction

$$\boldsymbol{R}_h = \boldsymbol{D}^\top (\boldsymbol{X}\boldsymbol{g}_h^\top) \in \mathbb{R}^{d\times r}. \tag{35}$$

These corrections can be computed during the same backward pass and cached for the adaptive selection. In particular, if $\boldsymbol{U}_0 = \boldsymbol{D}^\top \big(\boldsymbol{X}\boldsymbol{A}^\top\big)$ is the stacked gradient of the LoRA $\boldsymbol{B}$ factors, then

$$\bar{\boldsymbol{U}} = \boldsymbol{U}_0 - \eta_A \sum_{h\in\mathbb{S}_t} \boldsymbol{R}_h. \tag{36}$$

For an insertion candidate $i \notin \mathbb{S}_t$,

$$\boldsymbol{A}_i = \bar{\boldsymbol{A}} - \eta_A \boldsymbol{g}_i, \qquad \boldsymbol{U}_i = \bar{\boldsymbol{U}} - \eta_A \boldsymbol{R}_i, \tag{37}$$

$$\boldsymbol{C}_i = \bar{\boldsymbol{C}} - \eta_A \left(\bar{\boldsymbol{A}}\boldsymbol{g}_i^\top + \boldsymbol{g}_i \bar{\boldsymbol{A}}^\top\right) + \eta_A^2 \boldsymbol{g}_i \boldsymbol{g}_i^\top. \tag{38}$$

After selecting $\mathbb{S}_t^+$ in the insertion stage, denote its cached quantities by $(\boldsymbol{A}_+, \boldsymbol{C}_+, \boldsymbol{U}_+)$. Removing any $j \in \mathbb{S}_t^+$ reverses the corresponding virtual update:

$$\boldsymbol{A}_{-j} = \boldsymbol{A}_+ + \eta_A \boldsymbol{g}_j, \qquad \boldsymbol{U}_{-j} = \boldsymbol{U}_+ + \eta_A \boldsymbol{R}_j, \tag{39}$$

$$\boldsymbol{C}_{-j} = \boldsymbol{C}_+ + \eta_A \left(\boldsymbol{A}_+ \boldsymbol{g}_j^\top + \boldsymbol{g}_j \boldsymbol{A}_+^\top\right) + \eta_A^2 \boldsymbol{g}_j \boldsymbol{g}_j^\top. \tag{40}$$

Both stages therefore reuse the same low-rank corrections and require no additional forward or backward pass. The per-head shared-factor contribution is available without forming $\boldsymbol{G}_h$:

$$\boldsymbol{g}_h = \boldsymbol{B}_h^\top \boldsymbol{G}_h = (\boldsymbol{D}_h \boldsymbol{B}_h)^\top \boldsymbol{X}. \tag{41}$$

Complexity. A direct implementation first materializes the full gradient $\boldsymbol{G} = \boldsymbol{D}^\top \boldsymbol{X} \in \mathbb{R}^{d\times n}$, which costs $O(Ndn)$ time and $O(dn)$ additional memory. For each candidate, forming $\boldsymbol{G}(\boldsymbol{A}_{\mathbb{S}}^+)^\top$ costs $O(dnr)$; forming and factorizing $\boldsymbol{C}_{\mathbb{S}}$ and evaluating the trace add $O((d+n)r^2 + r^3)$. Directly evaluating M candidates therefore costs

$$O\big(Ndn + M\left[dnr + (d+n)r^2 + r^3\right]\big) \tag{42}$$

time, with $O(dn)$ storage for the full gradient (in addition to the low-rank candidate quantities).

In our implementation, the per-head quantities $\boldsymbol{g}_h$ and the low-rank corrections in equation 35 are accumulated without materializing $\boldsymbol{G}$ and are reused by all candidate evaluations. Once these quantities are available, updating $\boldsymbol{C}_{\mathbb{S}}$ costs $O(nr^2)$ per candidate, updating $\boldsymbol{U}_{\mathbb{S}}$ costs $O(dr)$, forming $\boldsymbol{U}_{\mathbb{S}}^\top \boldsymbol{U}_{\mathbb{S}}$ costs $O(dr^2)$, and computing the exact pseudoinverse of $\boldsymbol{C}_{\mathbb{S}}$ costs $O(r^3)$. The time complexity is therefore

$$O\big((d+n)r^2 + r^3\big) \tag{43}$$

per candidate. Across the local add–remove refinement stages there are at most $H + 1$ distinct candidate sets: the unchanged set and duplicated candidates are scored once and then reused. Hence, one complete adaptive selection costs

$$O\big(H\left[(d+n)r^2 + r^3\right]\big), \tag{44}$$

which is linear in the number of heads and operates entirely on low-rank matrices.

For completeness, constructing all $\boldsymbol{g}_h$ and $\boldsymbol{R}_h$ is not free even when it is fused with the backward pass. A conservative implementation that forms the required contractions separately takes $O(HN(n+d)r)$ time. Including this candidate-independent preparation gives the following upper bound for a complete adaptive selection:

$$O\big(HN(n+d)r + H\left[(d+n)r^2 + r^3\right]\big). \tag{45}$$

Excluding the activations and backward signals already retained for ordinary training, caching all $\{\boldsymbol{g}_h, \boldsymbol{R}_h\}_{h=1}^{H}$ requires $O(Hr(n+d))$ additional memory. In particular, the method avoids the $O(dn)$ full-gradient buffer. Since $r \ll \min(d,n)$, it replaces full-rank candidate projections with low-rank matrix computations; when $Hr \ll \min(d,n)$, the complete cache is also smaller than the full-gradient buffer.

C Implementation details and Additional experiments

C.1 Natural Language Understanding Tasks

C.1.1 Setup

For natural language understanding tasks, we evaluate these methods on several GLUE benchmark datasets (Wang et al., 2019), *i.e.*, CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), QQP[2], MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE (Wang et al., 2019).

We use the RoBERTa (Liu et al., 2019) model as the backbone and report the best result with the learning rate $\eta \in \{5\times10^{-5}, 1\times10^{-4}, 2\times10^{-4}, 5\times10^{-4}\}$. We apply the LoRA adapters to the query, key, and value projections in every attention layer. The batch size is 32, the number of epochs is 20 for smaller datasets(CoLA and RTE) and 5 for others, the gradient accumulation step is 1, the scaling factor is twice the rank, and the maximum sequence length is 128. The number of clusers K for initial head seletion is 4. The warm-up length t_c is 10% of the total training steps, adaptive selection interval T is 50 steps. All the experiments are conducted on a platform with 8 NVIDIA GeForce RTX 5090 GPUs.

C.1.2 Robustness to Rank.

We further evaluate the robustness of the proposed head selection strategy under different LoRA ranks. Specifically, we conduct experiments on RoBERTa with $r \in \{4, 8, 16\}$ while keeping the remaining training configurations unchanged. As shown in Table 5 and 6, incorporating head selection improves the performance of different LoRA variants across the evaluated ranks. These results indicate that the effectiveness of head selection is not tied to a particular rank setting. Instead, selectively controlling the headwise gradients to the shared low-rank subspace provides an effective optimization mechanism across different adapter capacities.

C.2 Comparison with Factor-Specific Learning Rates

Head selection changes both the direction and magnitude of the gradient supplied to the shared factor $\boldsymbol{A}$. To examine whether comparable gains can be obtained through a simpler adjustment of the relative learning rates of the two LoRA factors, we compare our method with LoRA+(Hayou et al., 2024), which uses different learning rates for A and B. We conduct this comparison on GLUE using RoBERTa at rank 8. For LoRA+, we set the learning-rate ratio $\lambda = \eta_B/\eta_A = 16$, the same setting following(Hayou et al., 2024).

[2]data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Table 5: Test accuracy(%) of RoBERTa with LoRA variants (with/without selection) on GLUE benchmark datasets (rank 4).

Method	SEL	CoLA	SST-2	QQP	MNLI	QNLI	RTE
LoRA	w/o	$56.38_{\pm 0.54}$	$93.69_{\pm 0.23}$	$87.37_{\pm 0.19}$	$85.26_{\pm 0.07}$	$91.24_{\pm 0.20}$	$69.31_{\pm 0.72}$
	w/	$61.80_{\pm 0.73}$ (+5.42)	$94.42_{\pm 0.24}$ (+0.73)	$89.24_{\pm 0.12}$ (+1.87)	$86.91_{\pm 0.09}$ (+1.65)	$92.52_{\pm 0.17}$ (+1.28)	$74.25_{\pm 1.67}$ (+4.94)
PiSSA	w/o	$60.25_{\pm 2.74}$	$93.88_{\pm 0.33}$	$87.97_{\pm 0.02}$	$85.38_{\pm 0.10}$	$91.67_{\pm 0.05}$	$69.43_{\pm 1.78}$
	w/	$63.05_{\pm 0.59}$ (+2.80)	$94.50_{\pm 0.23}$ (+0.62)	$89.22_{\pm 0.10}$ (+1.25)	$86.81_{\pm 0.15}$ (+1.43)	$92.28_{\pm 0.13}$ (+0.61)	$73.04_{\pm 0.75}$ (+3.61)
AdaLoRA	w/o	$58.35_{\pm 0.39}$	$93.96_{\pm 0.26}$	$87.74_{\pm 0.16}$	$85.55_{\pm 0.13}$	$91.34_{\pm 0.30}$	$69.68_{\pm 0.96}$
	w/	$62.57_{\pm 1.00}$ (+4.22)	$94.61_{\pm 0.11}$ (+0.65)	$89.34_{\pm 0.14}$ (+1.60)	$87.23_{\pm 0.41}$ (+1.68)	$92.70_{\pm 0.14}$ (+1.36)	$74.37_{\pm 0.96}$ (+4.69)
DoRA	w/o	$55.61_{\pm 0.92}$	$93.50_{\pm 0.18}$	$87.52_{\pm 0.09}$	$85.44_{\pm 0.14}$	$91.14_{\pm 0.41}$	$70.52_{\pm 1.04}$
	w/	$61.12_{\pm 1.90}$ (+5.51)	$94.38_{\pm 0.41}$ (+0.88)	$89.33_{\pm 0.07}$ (+1.81)	$86.94_{\pm 0.15}$ (+1.50)	$92.51_{\pm 0.26}$ (+1.37)	$72.56_{\pm 0.01}$ (+2.04)

Table 6: Test accuracy(%) of RoBERTa with LoRA variants (with/without selection) on GLUE benchmark datasets (rank 16).

Method	SEL	CoLA	SST-2	QQP	MNLI	QNLI	RTE
LoRA	w/o	$61.07_{\pm 0.62}$	$94.04_{\pm 0.20}$	$88.68_{\pm 0.14}$	$86.51_{\pm 0.07}$	$92.04_{\pm 0.06}$	$72.68_{\pm 1.46}$
	w/	$62.61_{\pm 1.57}$ (+1.54)	$94.69_{\pm 0.18}$ (+0.65)	$90.33_{\pm 0.05}$ (+1.65)	$87.57_{\pm 0.16}$ (+1.06)	$93.06_{\pm 0.20}$ (+1.02)	$74.85_{\pm 1.50}$ (+2.17)
PiSSA	w/o	$60.45_{\pm 0.98}$	$94.38_{\pm 0.30}$	$89.24_{\pm 0.15}$	$86.49_{\pm 0.27}$	$92.42_{\pm 0.32}$	$74.97_{\pm 1.04}$
	w/	$62.03_{\pm 1.19}$ (+1.58)	$94.50_{\pm 0.20}$ (+0.12)	$90.30_{\pm 0.06}$ (+1.06)	$87.33_{\pm 0.11}$ (+0.84)	$92.75_{\pm 0.24}$ (+0.33)	$75.69_{\pm 1.27}$ (+0.72)
AdaLoRA	w/o	$61.59_{\pm 0.87}$	$93.96_{\pm 0.35}$	$88.81_{\pm 0.01}$	$86.44_{\pm 0.05}$	$92.31_{\pm 0.20}$	$72.32_{\pm 0.75}$
	w/	$62.83_{\pm 2.18}$ (+1.24)	$94.53_{\pm 0.43}$ (+0.57)	$90.26_{\pm 0.04}$ (+1.45)	$87.61_{\pm 0.15}$ (+1.17)	$92.90_{\pm 0.14}$ (+0.59)	$73.04_{\pm 2.35}$ (+0.72)
DoRA	w/o	$57.67_{\pm 8.26}$	$94.19_{\pm 0.18}$	$88.69_{\pm 0.11}$	$86.44_{\pm 0.15}$	$92.13_{\pm 0.20}$	$72.68_{\pm 1.27}$
	w/	$60.75_{\pm 0.62}$ (+3.08)	$95.03_{\pm 0.18}$ (+0.84)	$90.32_{\pm 0.01}$ (+1.63)	$87.53_{\pm 0.19}$ (+1.09)	$92.90_{\pm 0.15}$ (+0.77)	$74.73_{\pm 2.20}$ (+2.05)

As shown in Table 7, LoRA+ improves over standard LoRA, indicating that adjusting the relative learning rates is beneficial in these settings. Importantly, however, our method outperforms LoRA+ across the datasets. These results suggest that the gains from head selection are not fully reproduced by the factor-specific learning-rate adjustments evaluated here, providing additional empirical support for selective gradient aggregation.

C.3 Natural Language Generation Tasks

C.3.1 Setup

For natural language generation tasks, we evaluate these methods on GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021) and commonsense reasoning benchmarks(Hu et al., 2023), including math, code and commonsense tasks. For the GSM8K task, we train the model on a 100k subset of the MetaMathQA dataset (Yu et al., 2024) and report the accuracy on GSM8K. For the HumanEval task, we train the model on a 100k subset of the Code-Feedback dataset (Zheng et al., 2024) and report the PASS@1 metric on HumanEval. For the commonsense reasoning benchmark, we train the model on Commonsense-170k(Hu et al., 2023) and report the score on BoolQ(Clark et al., 2019a), PIQA(Bisk et al., 2020), SIQA(Sap et al., 2019), WinoGrande(Sakaguchi et al., 2020), OBQA(Mihaylov et al., 2018), ARC-c and ARC-e(Clark et al., 2018).

We use the Llama-3.1-8B model (Dubey et al., 2024) and the Mistral-7B-v0.1 (Jiang et al., 2023) model as the backbone. For both models, we use the FP16/BF16 mixed-precision strategy to reduce the GPU memory consumption. All the experiments are conducted on a platform with 8 NVIDIA GeForce RTX 5090 GPUs. We adopt the AdamW optimizer (Loshchilov & Hutter, 2019), and report the best result with the learning rate $\eta \in \{1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$. We apply the LoRA adapters to the query, key, and value projections in every attention layer. The batch size is 1, the number of epochs is 1 for MetaMathQA and Code-Feedback dataset, 3 for commonse-170k, the gradient accumulation step is 8, the scaling factor is twice the rank, and the maximum sequence length is 1024. The number of clusers K for initial head seletion is 4. The warm-up length t_c is 10% of the total training steps, adaptive selection interval T is 50 steps.

Table 7: Comparison with LoRA+ on the GLUE benchmark.

Method	CoLA	SST-2	QQP	MNLI	QNLI	RTE
LoRA	58.81	93.80	87.51	85.47	91.19	71.36
LoRA+	61.82	94.15	89.19	86.66	92.24	73.29
Ours	**63.68**	**95.07**	**89.61**	**87.35**	**92.85**	**74.01**

Table 8: Accuracy (%) of Llama-3.1-8B with different LoRA variants, with and without our selection method, on commonsense reasoning benchmarks (rank 8).

Method	SEL	BoolQ	PIQA	SIQA	WinoGrande	ARC-e	ARC-c	OBQA
LoRA	w/o	72.14	86.02	80.09	86.42	92.42	79.35	84.60
	w/	72.38 (+0.24)	87.29 (+1.27)	81.83 (+1.74)	88.24 (+1.82)	92.51 (+0.09)	81.23 (+1.88)	87.00 (+2.40)
PiSSA	w/o	66.97	85.80	78.71	79.79	89.06	75.26	79.80
	w/	72.26 (+5.29)	86.67 (+0.87)	82.55 (+3.84)	88.24 (+8.45)	91.79 (+2.73)	81.65 (+6.39)	89.20 (+9.40)
AdaLoRA	w/o	65.44	87.10	75.18	86.74	89.98	77.13	84.00
	w/	67.61 (+2.17)	87.54 (+0.44)	82.91 (+7.73)	87.21 (+0.47)	91.88 (+1.90)	81.66 (+4.53)	87.20 (+3.20)
DoRA	w/o	72.96	86.34	79.84	87.45	91.16	79.27	85.60
	w/	73.00 (+0.04)	86.56 (+0.22)	81.21 (+1.37)	88.56 (+1.11)	92.17 (+1.01)	82.25 (+2.98)	85.60 (+0.00)

C.3.2 CommonSense Task

As shown in Table 8 and 9, our selection strategy generally improves or maintains the performance of different LoRA variants on Llama-3.1-8B and Mistral-7B-v0.1 across the evaluated commonsense reasoning benchmarks. These results suggest that the selection mechanism is not tied to a specific LoRA formulation, but can be effectively integrated with different strategies. Overall, the improvements across methods and tasks further demonstrate the generality of controlling headwise contributions when optimizing the shared low-rank subspace.

C.4 Relation to Attention-Heads Importance

We further discuss the relation between our work and head importance (Voita et al., 2019). We first examine whether the importance of individual attention heads changes substantially throughout training. We measure head importance by the increase in validation loss after removing the learned update associated with each head. As shown in Fig. 2, redder colors indicate greater importance of an attention head, whereas bluer colors indicate lower importance. Accordingly, it can be observed that at the beginning of training, the importance of attention heads is not yet evident; however, they become markedly differentiated in importance early in training. In particular, heads that become relatively influential after the initial training stage tend to remain so during later optimization, consistent with (Michel et al., 2019). Based on the observation, we assume that the functional roles of

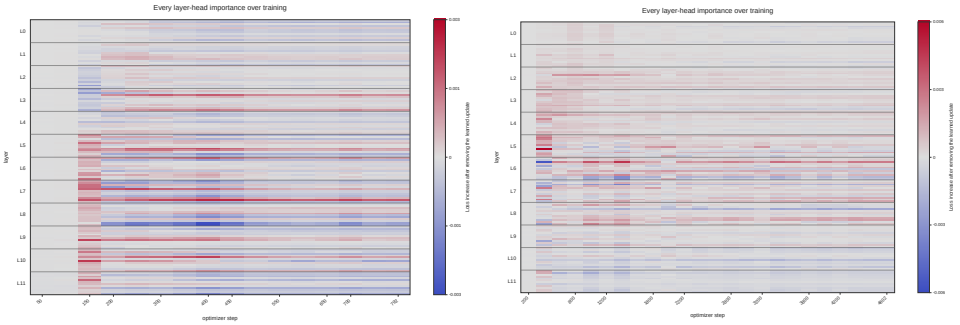
Table 9: Accuracy (%) of Mistral-7B-v0.1 with different LoRA variants, with and without our selection method, on commonsense reasoning benchmarks (rank 8).

Method	SEL	BoolQ	PIQA	SIQA	WinoGrande	ARC-e	ARC-c	OBQA
LoRA	w/o	72.38	84.82	81.11	87.37	91.37	80.63	86.40
	w/	73.79 (+1.41)	87.29 (+2.47)	81.83 (+0.72)	88.24 (+0.87)	92.51 (+1.14)	81.23 (+0.60)	87.00 (+0.60)
PiSSA	w/o	71.13	83.79	79.48	85.40	89.90	77.47	85.80
	w/	72.23 (+1.10)	84.06 (+0.27)	81.17 (+1.69)	86.82 (+1.42)	91.41 (+1.51)	77.73 (+0.26)	86.00 (+0.20)
AdaLoRA	w/o	63.02	80.63	79.07	88.24	92.17	80.12	87.80
	w/	66.88 (+3.86)	80.63 (+0.00)	81.93 (+2.86)	88.63 (+0.39)	92.21 (+0.04)	81.14 (+1.02)	88.60 (+0.80)
DoRA	w/o	71.89	84.82	81.12	87.37	90.74	80.63	86.40
	w/	73.79 (+1.90)	85.47 (+0.65)	81.88 (+0.76)	87.61 (+0.24)	91.37 (+0.63)	81.06 (+0.43)	88.60 (+2.20)

Table 10: Comparison with importance-based head selection on the GLUE benchmark.

Method	CoLA	SST-2	QQP	MNLI	QNLI	RTE
LoRA	58.81	93.80	87.51	85.47	91.19	71.36
Importance Top-50%	58.76	93.92	88.06	85.67	91.27	69.31
Ours	**63.68**	**95.07**	**89.61**	**87.35**	**92.85**	**74.01**

different attention heads may become relatively established early in training, offering an interesting perspective on our empirical finding that the head structure changes only gradually during training.



(a) Evolution of head importance in SST-2 (b) Evolution of head importance in MNLI

Figure 2: Evolution of head importance.

Importantly, stable head importance does not imply that the most important heads are necessarily the optimal ones for updating the shared LoRA factor. To examine this distinction, we construct an importance-based baseline: the top-50% most important heads are allowed to contribute to the update of the shared factor and we evaluate the top-50% most important heads every $T = 50$ steps throughout the remaining optimization, the same interval as our adaptive selection. We then evaluate this strategy on GLUE under the same experimental settings as standard LoRA (Hu et al., 2022) and our method in Table 10.

Interestingly, the importance-based baseline outperforms standard LoRA on several datasets, suggesting that restricting the update to a subset of informative heads can promote the optimization. However, the importance-based strategy still performs worse than our method. This indicates that individual head importance alone may be insufficient for identifying the most suitable heads for shared-factor optimization. Head importance measures how much an individual head contributes to the learned model, but does not account for whether its gradient is compatible with those of other heads when they jointly update the shared low-rank subspace. In contrast, our method selects heads according to their joint contribution to the shared subspace. Therefore, although important heads can provide useful update signals, the individually most important heads are not necessarily the most effective group for jointly optimizing the shared LoRA factor.