Sensitivity-Aware Adapter Placement for Efficient On-Device Fine-Tuning of Transformer Encoders

Anonymous ACL submission

Abstract

Fine-tuning encoder-based Transformers in memory-constrained settings (e.g., edge devices or 8–16 GB GPUs) is often limited by peak VRAM rather than wall-clock time. We propose *Sensitivity-Aware Adapter Placement* (SAAP), a parameter-efficient fine-tuning method that selectively instantiates low-rank adapters only in task-sensitive modules. SAAP identifies adapter locations using an activation-weighted squared-gradient score ($a \cdot g^2$) in a single probing process, introducing a modest one-time overhead (100–300 seconds) without architectural changes. Across DistilBERT, BERT-base, and RoBERTa-base on IMDb, AG News, Yelp Polarity, and TweetEval:Hate, SAAP updates far fewer parameters than standard LoRA while maintaining comparable accuracy. On BERT-base (IMDb), SAAP reduces trainable parameters from 1.20M to 0.03M ($> 97\%$) and lowers peak training VRAM from approximately 3.5 GB to 0.9 GB. Overall, SAAP improves accuracy-per-parameter trade-offs and provides a transparent, drop-in solution for memory-efficient fine-tuning.

1 Introduction

While Large Language Models dominate cloud computing, edge environments rely on compact encoders such as BERT and RoBERTa for privacy and offline capability. Fine-tuning these models on consumer-grade hardware is restricted by GPU memory, a bottleneck that standard Parameter-Efficient Fine-Tuning (PEFT) methods only partially address. Specifically, baselines like Low-Rank Adaptation (LoRA; Hu et al., 2022) apply adapters uniformly across the network, consuming scarce memory on modules that contribute little to task performance. We propose **Sensitivity-Aware Adapter Placement** (SAAP) to enable fine-tuning on strictly memory-constrained devices. Unlike adaptive rank methods such as AdaLoRA (Zhang et al., 2023), which distribute rank across existing adapters, SAAP uses a data-driven probing pass based on activation-weighted squared gradients ($a \cdot g^2$) to decide whether to instantiate an adapter at all. This results in structural sparsity specifically suited for on-device learning.

The impact on resource efficiency is substantial. When fine-tuning BERT-base on IMDb, SAAP reduces trainable parameters from 1.20M to 0.03M and lowers peak training memory from 3.5 GB to just 0.9 GB. This allows standard encoders to be fine-tuned on hardware with less than 2 GB VRAM while maintaining accuracy comparable to full fine-tuning. We frame efficiency strictly as training-time VRAM reduction, positioning SAAP as a critical enabler for personalised adaptation on the edge.

2 Related Work

Parameter-efficient fine-tuning (PEFT) reduces the barrier to adapting transformer models by updating only a small subset of parameters (Xu et al., 2023). While early methods like adapter modules (Houlsby et al., 2019) and prefix tuning (Li and Liang, 2021) introduced this paradigm, Low-Rank Adaptation (LoRA; Hu et al., 2022) has become the standard for edge applications due to its zero inference latency. However, standard LoRA applies adapters uniformly, and even extensions like QLoRA (Dettmers et al., 2023), which address storage via quantisation, do not inherently solve this structural redundancy. Recent approaches attempt to mitigate this dynamically: AdaLoRA (Zhang et al., 2023) reallocates rank based on singular value importance, and Mixture-of-Experts strategies like MoLA and AlphaLoRA (Gao et al., 2025,?; Qing et al., 2024) introduce routing mechanisms to improve expressivity. Yet, these methods often add architectural complexity prohibitive for strictly memory-bound devices. In contrast, SAAP determines adapter placement determinis-

tically prior to training, ensuring a minimal, predictable memory footprint.

Methodologically, our approach draws upon sensitivity analysis and network pruning. Techniques such as attention head importance (Michel et al., 2019), saliency maps (Simonyan et al., 2014), influence functions (Koh and Liang, 2017), DeepLIFT (Shrikumar et al., 2017), and first-order Taylor approximations (Molchanov et al., 2019; Sanh et al., 2020) have traditionally been used to remove redundant weights from dense networks. SAAP repurposes these signals for additive learning by employing a lightweight, activation-weighted squared-gradient probe ($a \cdot g^2$) to identify high-sensitivity submodules before fine-tuning begins. This strategy allows us to instantiate adapters only where necessary, treating structural sparsity as a primary tool for enabling on-device learning.

3 Methodology

We propose **Sensitivity-Aware Adapter Placement (SAAP)**, a framework for allocating parameter-efficient adapters (LoRA) only to those neural modules that contribute most strongly to task-specific loss reduction. To maintain computational efficiency, we restrict the candidate search space $\mathcal{M}$ to the linear projections commonly targeted by standard LoRA, namely the query, key, value, and output/feed-forward dense layers. Layer normalisation parameters and embeddings are excluded, as they are rarely adapted in practice and contribute minimally to adapter-based fine-tuning. To identify high-impact modules within $\mathcal{M}$, we introduce the GRAD-WEIGHTED MUL (GWM) heuristic, which prioritises representations that are both sensitive to training signals and actively utilised by the network. Let $A \in \mathbb{R}^{d_{\text{out}}}$ denote the output activation of a module and $G \in \mathbb{R}^{d_{\text{out}}}$ the gradient of the loss $\mathcal{L}$ with respect to A. We compute a scalar importance score for each module m by accumulating an activation-weighted squared-gradient signal over a class-balanced mini-batch $\mathcal{B}$:

$$S_{\text{GWM}}(m) = \sum_{x \in \mathcal{B}} \sum_{i} \left(|G_i^{(x)}|^2 \cdot |A_i^{(x)}| \right). \quad (1)$$

Squaring the gradient amplifies high-sensitivity features by penalising flat loss regions, while weighting by activation magnitude acts as a gating mechanism that suppresses gradients flowing through inactive neurons (e.g., those suppressed by ReLU). Raw sensitivity scores can be noisy and biased by network depth, so we apply a robust, multi-stage selection procedure (full algorithm in Appendix A). First, to mitigate depth-dependent gradient drift, we apply Min–Max normalisation to $S(m)$ within each transformer block, ensuring that selection is driven by relative intra-layer importance rather than global magnitude effects. Second, to capture features relevant to all labels, we compute normalised rankings independently for each class $c \in \mathcal{C}$. For each class-specific ranking, we apply the *Kneedle* algorithm (Satop"a"a et al., 2011) to determine an adaptive cutoff k_c. The final set of adapter locations is then obtained by taking the union of the top-ranked modules across classes,

$$\mathcal{M}_{\text{final}} = \bigcup_{c \in \mathcal{C}} \text{Top}_{k_c}(\mathcal{M}),$$

allowing SAAP to dynamically adjust the adapter budget based on the knee of the sensitivity curve rather than an arbitrary sparsity constraint.

4 Experiments

We evaluate SAAP on DistilBERT, BERT, and RoBERTa across four text classification benchmarks: IMDb, AG News, Yelp Polarity, and TweetEval (Hate). We compare performance against **Standard LoRA** (targeting query, value, and dense modules), **AdaLoRA** (adaptive rank allocation), and **Full Fine-Tuning**.

4.1 Experimental Setup

Baselines. To strictly isolate the benefit of our sensitivity-based placement from parameter count, we introduce a **Budget-Matched Random** baseline. Unlike generic random selection, this baseline restricts its search space to the exact number of modules (K) automatically discovered by SAAP for that specific seed, ensuring a fair comparison of placement strategies at identical sparsity levels.

SAAP Instantiation. We deploy the framework using the GRAD-WEIGHTED MUL metric (§3.2) and the robust selection pipeline (§3.3). For the stability protocol, we aggregate scores over $N = 20$ bootstrap subsampling passes. To further filter transient noise, we apply a frequency threshold, retaining only those modules selected in at least 99% of the bootstrap passes before the final union selection. The adaptive budget k_c is determined dynamically via Kneedle without manual sparsity constraints.

Implementation Details. All experiments use PyTorch (Paszke et al., 2019) and Hugging Face Transformers. We enforce reproducibility via a dual-seed strategy: a fixed seed (42) for model initialisation and a variable run-seed for data sampling and stability selection.

- **Training:** We use an effective batch size of 32 (`batch=8`, `accum=4`), learning rate $5e{-}4$, and weight decay 0.01. Training is conducted for 4 epochs using a split of 5,000 training samples and 1,000 test samples.
- **Adapters:** LoRA is configured with rank $r = 8$, $\alpha = 16$, and dropout 0.05.
- **AdaLoRA:** We initialise with rank $r = 12$ decaying to a target $r = 8$, with pruning occurring between steps 50 (t_{init}) and 250 (t_{final}) to accommodate short training durations.

5 Results and Analysis

Table 1 summarises the performance and efficiency of SAAP across architectures and tasks. Overall, SAAP achieves a favourable efficiency–performance trade-off: while Full Fine-Tuning (Full FT) attains marginally higher accuracy in some settings, it does so at the cost of 10–100$\times$ more trainable parameters and substantially higher computational and energy overhead. Compared to Standard LoRA and AdaLoRA, SAAP remains competitive in accuracy while dramatically reducing parameter count and memory usage. These gains are driven by structural sparsity rather than rank compression alone. For example, on **BERT-base (IMDb)**, SAAP requires only $\approx$ 0.03M trainable parameters compared to 1.19M for Standard LoRA ($>$ 97% reduction), resulting in a peak VRAM usage of 915 MB versus 3543 MB for LoRA. Unlike AdaLoRA, which dynamically masks parameters but still instantiates adapters across layers, SAAP entirely avoids allocating adapters in low-sensitivity modules, directly translating sparsity into memory savings.

Across baselines, SAAP matches or slightly trails Standard LoRA in accuracy while consistently dominating efficiency metrics. Relative to AdaLoRA, SAAP is competitive or superior (e.g., $+1.5\%$ accuracy on DistilBERT/Yelp) while using substantially less memory. SAAP also consistently outperforms a **Budget-Matched Random** baseline (observed at $\approx$ 1–3% lower accuracy), confirming that the proposed $\mathcal{S} = \sum |\nabla W|^2 \cdot |A|$ metric identifies information-dense submodules rather than benefiting from sparsity alone. On the challenging **TweetEval:Hate** task, SAAP ties for the highest accuracy (0.555), suggesting that highly targeted updates may act as an implicit regulariser on small or noisy datasets.

Notably, SAAP induces markedly different adapter budgets across encoder families, reflecting architecture-specific redundancy. On **BERT-base**, SAAP uses as little as $\sim$ 0.08M trainable parameters (about 6% of Standard LoRA), whereas **DistilBERT** and **RoBERTa** retain $\sim$ 0.71M (87%) and $\sim$ 0.63M (35%), respectively. This behaviour is consistent with prior understanding: BERT-base is substantially over-parameterised, enabling aggressive sparsification, while DistilBERT has already undergone compression via distillation and therefore requires denser adaptation. RoBERTa's stronger pre-training similarly reduces excess capacity, leading SAAP to retain more parameters than in BERT but still far fewer than uniform LoRA. The identical parameter counts observed across five runs for DistilBERT and RoBERTa ($\sigma = 0$) further indicate that SAAP's sensitivity estimates are stable and architecture-driven.

5.1 Qualitative and Ablation Analysis

Beyond efficiency, we examine whether the modules selected by the proposed S_{GWM} metric align with known properties of Transformer layer specialisation. Across architectures, SAAP consistently bypasses lower layers (0–2), which are known to encode general syntactic and surface features, and concentrates adaptation in upper-layer `output.dense` projections for semantically demanding tasks. This pattern mirrors the rediscovery of a classical NLP pipeline reported by Tenney et al. (2019) and aligns with findings that higher layers encode task-specific semantics (Kovaleva et al., 2019). In this sense, SAAP functions as an interpretable structural probe rather than a random sparsification strategy.

We further validate the robustness of the sensitivity signal through targeted ablations. Comparing the proposed Grad-Squared Activation score ($a \cdot g^2$) against single-signal alternatives (e.g., $|a|$, $|g|$) and other gradient–activation combinations shows that composite metrics generally outperform isolated signals, with $a \cdot g^2$ yielding the best balance of mean accuracy and variance. Finally, we analyse sensitivity to probing batch size and observe a clear

Model	Dataset	Method	Test Accuracy	Test Loss	Peak GPU Mem (MB)	Trainable Params
BERT base	AG News	Full FT	$0.905_{\pm 0.000}$	$0.282_{\pm 0.000}$	$2844_{\pm 320.7}$	$109.5M_{\pm 0}$
		LoRA†	$0.912_{\pm 0.002}$	$0.267_{\pm 0.004}$	$1613_{\pm 107.2}$	$1.20M_{\pm 0}$
		AdaLoRA	$0.889_{\pm 0.002}$	$0.325_{\pm 0.003}$	$2262_{\pm 379.1}$	$1.79M_{\pm 0}$
		Budget-Matched Random	$0.894_{\pm 0.010}$	$0.320_{\pm 0.016}$	$1066_{\pm 328.8}$	$0.14M_{\pm 22.6k}$
		SAAP	$0.891_{\pm 0.005}$	$0.322_{\pm 0.015}$	$724_{\pm 133.4}$	**0.075M**$_{\pm 11K}$
BERT base	IMDB	Full FT	$0.917_{\pm 0.000}$	$0.265_{\pm 0.000}$	$4268_{\pm 8.5}$	$109.5M_{\pm 0}$
		LoRA†	$0.917_{\pm 0.004}$	$0.242_{\pm 0.010}$	$3533_{\pm 32.8}$	$1.19M_{\pm 0}$
		AdaLoRA	$0.889_{\pm 0.002}$	$0.276_{\pm 0.002}$	$3575_{\pm 7.3}$	$1.79M_{\pm 0}$
		Budget-Matched Random	$0.897_{\pm 0.009}$	$0.257_{\pm 0.012}$	$1999_{\pm 212.6}$	$0.14M_{\pm 22.6k}$
		SAAP	$0.886_{\pm 0.002}$	$0.280_{\pm 0.002}$	$904_{\pm 26.8}$	**0.026M**$_{\pm 0}$
Distil BERT	Yelp Polarity	Full FT	$0.943_{\pm 0.000}$	$0.199_{\pm 0.000}$	$3609_{\pm 3.5}$	$67.0M_{\pm 0}$
		LoRA†	$0.933_{\pm 0.002}$	$0.189_{\pm 0.003}$	$2614_{\pm 13.6}$	$0.81M_{\pm 0}$
		AdaLoRA	$0.912_{\pm 0.003}$	$0.217_{\pm 0.002}$	$2629_{\pm 36.2}$	$0.92M_{\pm 0}$
		Budget-Matched Random	$0.925_{\pm 0.007}$	$0.192_{\pm 0.018}$	$2092_{\pm 237.9}$	$0.73M_{\pm 13K}$
		SAAP	$0.926_{\pm 0.002}$	$0.197_{\pm 0.003}$	$1260_{\pm 31.3}$	**0.71M**$_{\pm 0}$
RoBERTa base	TweetEval Hate	Full FT	$0.512_{\pm 0.000}$	$1.527_{\pm 0.000}$	$2523_{\pm 72.3}$	$124.6M_{\pm 0}$
		LoRA†	$0.514_{\pm 0.007}$	$1.512_{\pm 0.014}$	$1220_{\pm 101.2}$	$1.78M_{\pm 0}$
		AdaLoRA	$0.555_{\pm 0.003}$	$0.906_{\pm 0.008}$	$1317_{\pm 99.8}$	$2.38M_{\pm 0}$
		Budget-Matched Random	$0.529_{\pm 0.019}$	$1.246_{\pm 0.105}$	$890_{\pm 54.2}$	$0.73M_{\pm 13.7K}$
		SAAP	$0.555_{\pm 0.001}$	$0.712_{\pm 0.001}$	$754_{\pm 73.2}$	**0.63M**$_{\pm 0}$

Table 1: Comparison of Full Fine-Tuning, standard LoRA, AdaLoRA, Budget-Matched Random baseline, and SAAP. Results report the mean ± 95% confidence interval over 5 seeds. SAAP achieves competitive accuracy (significantly outperforming the Budget-Matched Random baseline on TweetEval) with consistently lower peak VRAM usage and fewer trainable parameters than AdaLoRA or LoRA.

knee at $N \approx 100$: increasing N from 10 to 100 significantly improves stability, while larger batches yield diminishing returns. This indicates that the gradient–activation signal converges rapidly, allowing SAAP to extract reliable sensitivity estimates with minimal computational overhead.

6 Discussion and Conclusion

Our results position SAAP as a transparent, drop-in alternative to uniform LoRA for constrained environments. We proposed SAAP to bridge the gap between large pre-trained encoders and memory-bound edge devices. Unlike methods that rely on complex routing or quantisation alone (Gao et al., 2025; Qing et al., 2024), SAAP utilises a lightweight probing phase to deterministically allocate adapters where they matter most. Our experiments show this approach dramatically lowers the entry barrier for on-device adaptation: reducing peak training memory from 3.5 GB to 0.9 GB on BERT-base without significant accuracy degradation. This structural efficiency aligns with the goals of Green AI by minimising the computational overhead of fine-tuning. While our current evaluation focused on encoder-only architectures, the orthogonality of SAAP to rank-allocation techniques suggests a promising avenue for hybrid approaches combining structural placement with rank tuning (Zhang et al., 2023). Ultimately, SAAP provides a stable, calibration-preserving solution for bringing personalised, privacy-preserving NLP to resource-constrained hardware.

7 Limitations

While the proposed Sensitivity-Aware Adapter Placement (SAAP) framework demonstrates strong parameter efficiency and competitive performance across multiple benchmarks, several limitations of the current study point to important directions for future work. First, our empirical evaluation is limited to encoder-only transformer architectures. Given the growing dominance of generative and instruction-following models, extending SAAP to autoregressive and sequence-to-sequence tasks represents a necessary and non-trivial next step. Second, all experiments in this work are conducted using "base"-scale models. As a result, the behaviour of the probing phase and the stability of the sparsity–performance trade-off at substantially larger scales remain unexplored. Third, our evalua-

tion focuses exclusively on text classification tasks. In complex reasoning or long-context modelling, the relative importance of different submodules may shift dynamically, which is not accounted for in the current formulation. Fourth, SAAP relies on a lightweight but non-trivial probing phase that uses a small, class-balanced subset of labelled data. This assumption may not hold in extreme few-shot scenarios or streaming settings. Finally, SAAP achieves parameter and training-memory sparsity, not inference latency reduction, as unselected modules remain active during the forward pass.

8 Ethical considerations

This work contributes to the broader literature on parameter-efficient fine-tuning (PEFT) with the goal of lowering the computational barriers to adapting large pretrained language models. By selectively placing adapters only in sensitivity-critical submodules, SAAP substantially reduces the number of trainable parameters relative to uniform LoRA-based approaches and lowers GPU memory requirements during training. As a result, the proposed method directly reduces energy consumption and associated carbon emissions, aligning with the objectives of Green AI. In addition to efficiency benefits, SAAP promotes greater transparency in model adaptation. This explicit selection process enables practitioners to inspect which components of a pretrained model are most influential for a given task. We also acknowledge ethical considerations related to dataset choice and potential bias. One of the evaluated benchmarks, TweetEval:Hate, contains subjective annotations. We recommend that practitioners conduct fairness and bias evaluations when deploying SAAP-adapted models in real-world content moderation systems. Finally, we acknowledge the use of large language models (ChatGPT and Gemini) to assist with refining the clarity and professional tone of the writing in this paper.

A Full Selection Procedure

For completeness, we outline the stability-aware selection algorithm referenced in Section 3:

1. **Filtering:** Restrict candidates $\mathcal{M}$ to standard linear projections (Query, Value, Dense).

2. **Sampling:** For each class c, sample a balanced mini-batch from the training set.

3. **Scoring:** Run forward and backward passes to collect a and g. Compute $S^{(c)}_{\text{GWM}}(m) = \sum |a \odot g^2|$.

4. **Normalisation:** Group modules by layer index (e.g., `layer.11`). Apply Min-Max normalisation within each group to scale scores to $[0, 1]$.

5. **Thresholding:** Apply the Kneedle algorithm to the sorted scores of class c to find the elbow point, selecting the top modules.

6. **Aggregation:** Form the final adapter set as the union over classes: $\mathcal{M}_{\text{selected}} = \mathcal{M}_{\text{pos}} \cup \mathcal{M}_{\text{neg}}$.

7. **Budgeting (Optional):** If a hard budget is enforced, rank the union set by aggregate score and truncate.

B Optimisation of the SAAP Probing Sample Size

To ensure the robustness and efficiency of the SAAP method employed in this study, we conducted a preliminary sensitivity analysis to determine the optimal probing sample size. The objective was to identify a value that balances computational efficiency with statistical significance, thereby avoiding both under-sampling and unnecessary resource expenditure. We evaluated the SAAP method across a range of variable sample sizes (e.g. 5,10, up to 5000), utilising the **Knee Point Detection method** (Kneedle algorithm) to rigorously identify the optimal cutoff point. This statistical approach detects the point of maximum curvature in the performance curve, representing the point of diminishing returns where increasing the sample value yields negligible improvement relative to the cost.

As illustrated in Figure 1, the performance curve exhibits a sharp initial decrease in evaluation loss followed by a plateau. Mathematical analysis of the curvature confirmed that the inflection point occurs at exactly 100. Consequently, a value of 100 was adopted as the standard for all SAAP methodologies reported in the main text.

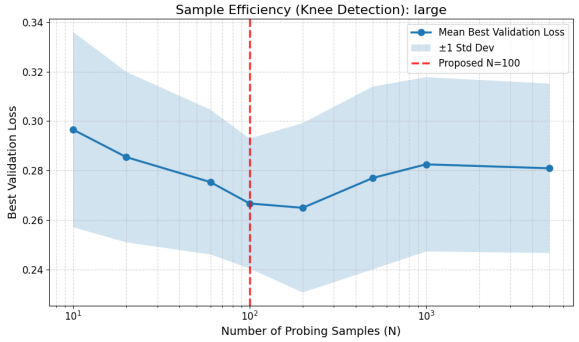

Figure 1: Optimisation of the SAAP method using Knee Point analysis. The plot demonstrates the relationship between sample size and method performance. The vertical indicator confirms the statistically determined knee point at a value of 100, which was selected as the optimal parameter.

C Selection of Importance Heuristic

To identify the most effective metric for the SAAP method, we evaluated a broad spectrum of importance heuristics. We tested formulations ranging from isolated components, specifically raw Gradient-only and Activation-only metrics, to hybrid multiplicative combinations, such as Gradient $\times$ Activation and (Gradient$\times$ Activation)^4.

As demonstrated in Figure 2, heuristics that integrated both gradient and activation information consistently outperformed those relying on a single metric. Upon further analysis of the hybrid methods, the **Gradient**$^2\times$ **Activation** formulation proved to be the most robust. It exhibited superior stability across different samples while yielding the highest performance results. Consequently, this heuristic was selected as the core metric for the SAAP method presented in this work.

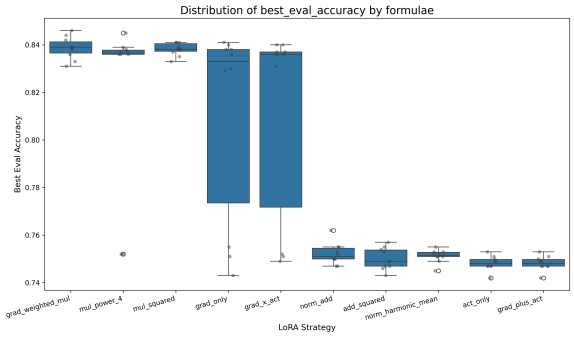

Figure 2: Performance of importance heuristics. Hybrid methods utilising gradient and activation signals outperform single-metric approaches. The **Gradient**$^2\times$ **Activation** heuristic (labeled `grad weighted mul`) achieves the optimal balance of performance and stability.

References

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 10088–10115.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and V. S. Subrahmanian. 2025. Mola: Moe lora with layer-wise expert allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5097–5112. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, pages 2790–2799.

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

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1885–1894.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. 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)*, pages 4365–4374.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Pavlo Molchanov, Arun Mallya, Jan Kautz, Insung Ihm, and Xia Shen. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11264–11272.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances*

in Neural Information Processing Systems (NeurIPS), volume 32.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. 2024. AlphaLoRA: Assigning LoRA experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 20378–20389.

Ville Satop"a"a, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a 'kneedle' in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171. IEEE.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3145–3153.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ian Tenney, Dipanjan Das, and Slav Petrov. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Qingru Zhang, Min Chen, Alexander Bukharin, Penn He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.