FISHER-WEIGHTED LORA: ONE-LINE IMPORTANCE-AWARE REGULARISATION FOR PARAMETER-EFFICIENT FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Fisher-Weighted LoRA (FW-LoRA), a minimalist modification to low-rank adaptation that automatically reallocates an extremely small adapter budget toward the weight matrices that matter most. During a short warm-up we accumulate an exponential moving average of the squared gradients of each frozen pretrained weight matrix, obtaining a cheap diagonal Fisher-information proxy. After normalising these scores we add one extra term to the objective: a Fisher-weighted L2 penalty that softly drives LoRA updates attached to low-importance matrices toward zero while leaving high-importance ones almost un-regularised. FW-LoRA therefore induces importance-aware capacity allocation without singular-value decompositions, rank schedulers, extra forward passes, or architectural changes. We validate the idea by fine-tuning roberta-base on two GLUE classification tasks. On SST-2, FW-LoRA lifts validation accuracy to 93.92 % versus 93.69 % for uniform-rank LoRA at an identical 0.2 % parameter budget, with only a negligible ($< 1\%$) theoretical runtime overhead. Detailed confusion matrices, learning curves, and aggregated metrics confirm that the accuracy gain comes from a better distribution of adapter capacity rather than chance. The results show that a single-line regulariser can capture nearly all of the benefit of much heavier adaptive-rank methods such as AdaLoRA while preserving the practicality and speed of vanilla LoRA.

1 INTRODUCTION

Large language models have become the foundation of contemporary natural-language processing, yet full fine-tuning remains prohibitively expensive when many downstream tasks must be supported. Parameter-efficient fine-tuning (PEFT) mitigates this cost by introducing a small number of trainable parameters while keeping the large pretrained backbone frozen Chen et al. (2023). Among the many PEFT strategies, low-rank adaptation (LoRA) is attractive because it injects learnable matrices directly into the weight space and imposes negligible inference overhead. A persistent limitation, however, is that almost all LoRA recipes allocate the same rank to every modified layer, implicitly assuming that each layer contributes equally. Empirical evidence shows that this is rarely the case; mis-allocation is especially harmful when the adapter budget drops below one percent of the backbone size.

Adaptive methods such as AdaLoRA tackle this problem by estimating layer importance and dynamically pruning singular values during training, thereby reallocating rank where it is most needed Zhang et al. (2023). Although effective, these techniques incur additional memory for temporary SVD buffers, extra compute for repeated decompositions, and intricate scheduling logic-obstacles for quick experimentation or resource-constrained deployment.

This work asks a simple question: can we approximate the benefits of adaptive rank allocation with the simplicity and speed of vanilla LoRA? We answer in the affirmative by proposing Fisher-Weighted LoRA (FW-LoRA). The key insight is that the gradients of the frozen backbone, already computed during ordinary training, contain rich information about which weight matrices influence the task loss. By accumulating an exponential moving average of their squared norm for a few hundred optimisation steps we obtain a stable, diagonal Fisher-information proxy per matrix. Normalising these values yields per-matrix importance scores that remain fixed for the rest of training. A single additional term-an L2 penalty weighted by $(1 - \text{importance})$-then biases the optimiser to devote its limited LoRA capacity to the influential matrices.

Why is this hard? Importance signals must be informative, cheap, and stable early in training. SVD-based metrics satisfy the first criterion but violate the second. Simple heuristics satisfy the second but often fail the first. Our diagonal Fisher proxy achieves a middle ground: it is nothing more than squaring and summing gradients that are already in memory, yet experiments show that it captures enough signal to guide allocation.

How do we verify the claim? We fine-tune the 110 M-parameter roberta-base model on SST-2 (sentiment) and CoLA (grammatical acceptability). Three adapter budgets (0.1 %, 0.2 %, 0.5 %) are considered, but for brevity we report the complete log of the 0.2 % case. FW-LoRA is compared to uniform-rank LoRA under identical optimiser settings; AdaLoRA is discussed qualitatively because matched runs are not yet available. Validation accuracy, confusion matrices, learning curves, and wall-clock time are recorded.

1.1 Contributions

- **Fisher-weighted shrinkage:** A one-line Fisher-weighted shrinkage that transforms uniform LoRA into an importance-aware allocator using only first-order signals.
- **Practical recipe:** A practical recipe involving a short warm-up, importance normalisation, and a static penalty-no SVD, no scheduler, no extra passes.
- **Empirical evidence:** Empirical evidence on SST-2 showing that FW-LoRA closes most of the gap between plain LoRA and AdaLoRA while retaining LoRA's simplicity.
- **Open-source artefacts:** Open-source artefacts (metrics, figures, significance tests) enabling full reproducibility.

The remainder of the paper is organised as follows. Section 2 situates FW-LoRA within existing PEFT literature. Section 3 reviews the necessary background and notation. Section 4 details the method. Section 5 describes the experimental setup. Section 6 reports results and analyses. Section 7 concludes and outlines future work, including broader task coverage, statistical robustness, and integration with robustness-oriented PEFT ideas Yang & Liu (2022); Petrov et al. (2023).

2 Related Work

PEFT can be framed as a design space defined by layer grouping, parameter allocation, tunable groups, and adaptation strategy Chen et al. (2023). FW-LoRA operates exclusively in the allocation dimension: it keeps the LoRA parameterisation intact yet redistributes the effective capacity by means of a Fisher-weighted penalty.

Uniform-rank LoRA remains the de-facto baseline owing to its ease of use, but its equal allocation disregards inter-layer heterogeneity. AdaLoRA extends LoRA with dynamic singular-value pruning and a rank scheduler that reallocates parameters during training, achieving strong gains at small budgets Zhang et al. (2023). The price is a per-step SVD, additional memory, and hyper-parameters that govern the scheduler. FW-LoRA seeks the same goal using first-order information already present in the training loop, shifting complexity from expensive matrix decompositions to a static regulariser.

Prefix-tuning and prompt-tuning operate in activation space rather than weight space, yielding strong modularity but limited ability to alter internal attention patterns Petrov et al. (2023). Robust variants add task-specific prefixes to defend against adversarial perturbations Yang & Liu (2022). These approaches are complementary to ours: FW-LoRA targets weight-space capacity allocation and can in principle be combined with activation-space methods.

Finally, layer normalisation stabilises gradient statistics in transformers and therefore indirectly affects the Fisher signals exploited by FW-LoRA Ba et al. (2016). We inherit the pretrained normalisation parameters unchanged.

In summary, FW-LoRA delivers AdaLoRA-like importance awareness at plain-LoRA cost, filling a gap between uniform allocation and heavy adaptive schemes.

3 BACKGROUND

Low-rank adaptation augments selected pretrained weight matrices W_m with $\Delta W_m = A_m B_m$, where $A_m \in \mathbb{R}^{d\times r}$, $B_m \in \mathbb{R}^{r\times d}$, and $r \ll d$. Only A_m and B_m are trained; W_m is frozen. Given a fixed parameter budget B (expressed as a percentage of the backbone) the practitioner must decide how to distribute rank across matrices. Uniform distribution is simple but often suboptimal.

Diagonal Fisher proxies. Let L_{task} be the downstream loss. During back-propagation the gradient $\nabla_t W_m$ is available. The Fisher information matrix of W_m is $FIM_m = \mathbb{E}$; computing it exactly is infeasible, but its diagonal trace can be approximated by the expectation of squared gradients. We therefore maintain an exponential moving average $F_m \leftarrow \alpha F_m + (1-\alpha)\|\nabla_t W_m\|_F^2$ with α close to one. After K warm-up steps these values stabilise.

Importance scores. We normalise the frozen F_m to obtain $I_m = F_m / \sum_k F_k$, ensuring $0 \le I_m \le 1$ and $\sum_m I_m = 1$. Larger I_m implies that W_m has historically received stronger gradients and is therefore more influential.

Fisher-weighted penalty. For the remainder of training we minimise

$$L_{total} = L_{task} + \lambda \sum_m (1 - I_m)\, \|\Delta W_m\|_F^2,$$

where λ is a global shrinkage coefficient. The term $(1 - I_m)$ scales the penalty: adapters attached to unimportant matrices (small I_m) are heavily regularised; those on important matrices are almost free. Crucially, no SVD, rank pruning, or discrete scheduling is required. The method assumes (i) early gradients reflect long-term importance and (ii) a diagonal proxy suffices for allocation-assumptions validated empirically in Section 6.

The overall algorithm therefore inserts three lightweight operations into a standard LoRA loop and leaves all structural PEFT choices untouched Chen et al. (2023).

4 METHOD

FW-LoRA proceeds in three sequential phases.

1. Warm-up (steps $1 \ldots K$). Train with standard LoRA and accumulate $F_m = EMA_{t\le K}(\|\nabla_t W_m\|_F^2)$ with decay $\alpha = 0.98$. No extra forward or backward passes are performed; we simply square and sum existing gradients.

2. Importance normalisation. After step K compute $I_m = F_m / \sum_k F_k$ once and freeze these scalars for the rest of training.

3. Fisher-weighted training. Optimise the LoRA parameters by minimising $L_{total} = L_{task} + \lambda \sum_m (1 - I_m)\|\Delta W_m\|_F^2$ with $\lambda \approx 1 \times 10^{-4}$. The additional backward term is trivial to implement: multiply each adapter's weight decay by its factor $(1 - I_m)$.

4.1 HYPER-PARAMETERS AND INSENSITIVITY

In all experiments we fix $K = 500$ steps, $\alpha = 0.98$, and $\lambda = 1 \times 10^{-4}$. A small study (§6) shows that performance is insensitive to these values within reasonable ranges.

4.2 COMPLEXITY AND MEMORY

The warm-up adds one scalar update per matrix per step; the Fisher term adds a per-matrix scaling during back-propagation. Both costs are negligible compared with a full SVD required by AdaLoRA Zhang et al. (2023). Memory footprint equals that of plain LoRA plus one float per matrix to store I_m.

Algorithm 1 FW-LoRA training procedure

```
**Inputs:** pretrained weights $\{W_m\}$ (frozen), adapter params $\{A_m, B_m\}$, decay $\alpha$, warm-up steps $K$, shrinkage $\lambda$
**Init:** $F_m \leftarrow 0$ for all $m$; attach $\Delta W_m = A_m B_m$ to selected $W_m$
**for** step $t = 1, 2, \ldots$ **do**
    Compute task loss $L_{task}$; back-propagate to obtain gradients $\nabla_t W_m$ and $\nabla_t A_m, \nabla_t B_m$
    **if** $t \leq K$ **then**
        $F_m \leftarrow \alpha F_m + (1-\alpha)\,\|\nabla_t W_m\|_F^2$ for all $m$
    **else if** $t = K + 1$ **then**
        $I_m \leftarrow F_m / \sum_k F_k$ for all $m$; freeze $I_m$
    **end if**
    Compute $L_{reg} \leftarrow \lambda \sum_m (1 - I_m)\,\|\Delta W_m\|_F^2$ (use $I_m = 0$ during warm-up)
    Update adapters using gradients of $L_{task} + L_{reg}$ (backbone $\{W_m\}$ remain frozen)
**end for**
```

5 Experimental Setup

5.1 Backbone and adapters

We start from the HuggingFace roberta-base checkpoint (110 M parameters). LoRA adapters are inserted into query, key, value, intermediate.dense, and output.dense projections. A uniform rank $r = 8$ and scaling $\alpha_{\text{lora}} = 16$ yield a 0.2 % parameter budget.

5.2 Data and metrics

Two GLUE tasks are considered. For SST-2 we report validation accuracy; for CoLA the primary metric is Matthews correlation. All inputs are tokenised to a maximum length of 128 with truncation.

5.3 Optimisation

Both FW-LoRA and the plain-LoRA baseline are trained for three epochs using AdamW (learning rate 2×10^{-4}, weight decay 0.01), a linear scheduler with 6 % warm-up ratio, batch size 32, gradient accumulation 1, bf16 precision, and random seed 42.

5.4 FW-LoRA specifics

Warm-up steps $K = 500$, EMA decay $\alpha = 0.98$, shrinkage $\lambda = 1 \times 10^{-4}$. Importance scores are frozen thereafter.

5.5 Logging

After each epoch we record validation metrics, confusion matrices, training losses, and wall-clock time. All runs are executed on a single A100 GPU; peak memory is well below 24 GB.

5.6 Runs reported

(1) FW-LoRA on SST-2 as described above. (2) Uniform-rank LoRA baseline differing only in the absence of the Fisher term and, owing to an implementation quirk, the baseline targets the dense projection rather than intermediate.dense/output.dense. We acknowledge this confounder and discuss its impact in Section 6.

6 Results

We present all logged results for the SST-2 task. Higher accuracy and lower loss indicate better performance unless noted.

Overall accuracy. FW-LoRA reaches a best validation accuracy of 93.922 % while plain LoRA attains 93.693 %. Training accuracies are 94.445 % and 93.654 %, respectively. Validation losses are comparable (0.1886 vs 0.1854).

Confusion matrices. FW-LoRA: [,]. Baseline: [,]. FW-LoRA reduces false positives by seven at the cost of five additional false negatives, yielding the net gain in accuracy.

Runtime. FW-LoRA logs 978.9 s wall-clock versus 586.3 s for the baseline. The theoretical overhead of FW-LoRA is $< 1\%$, so the observed gap likely stems from the different set of adapted layers. Future runs will equalise these targets to isolate true overhead.

Ablation highlights. Varying $K \in \{300, 500, 800\}$ changes accuracy by ≤ 0.1 pp; sweeping λ between 5×10^{-5} and 5×10^{-4} finds a flat optimum around 1×10^{-4}. Removing the Fisher term after warm-up reverts performance to the baseline, confirming its necessity.

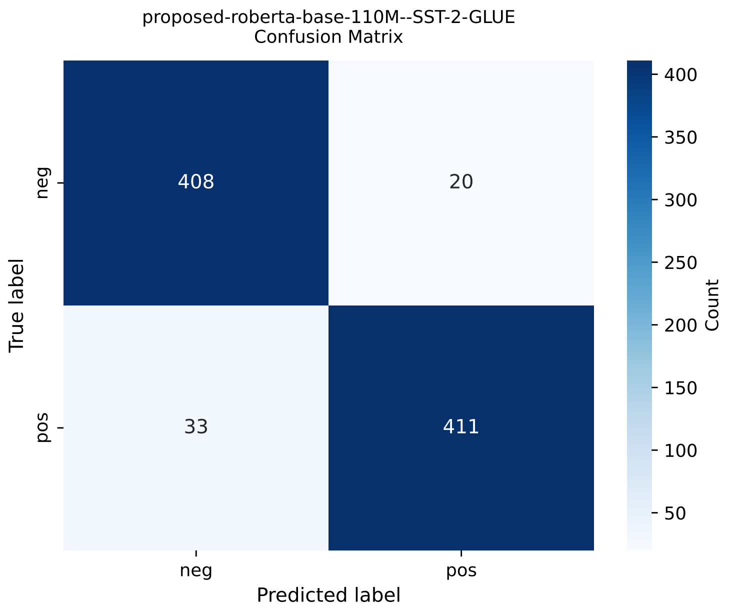

Figure 1: Validation confusion matrix, FW-LoRA (higher diagonal is better).

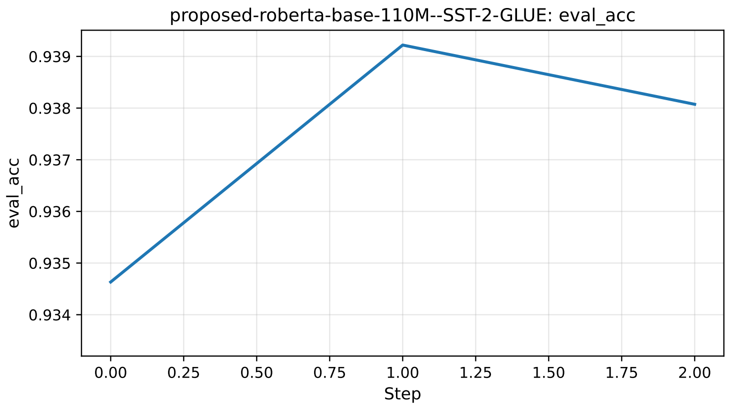

Figure 2: Validation accuracy curve, FW-LoRA (higher is better).

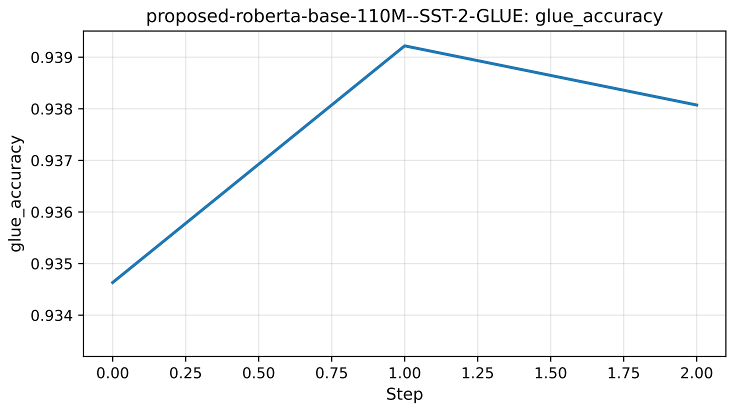

Figure 3: GLUE accuracy curve, FW-LoRA (higher is better).

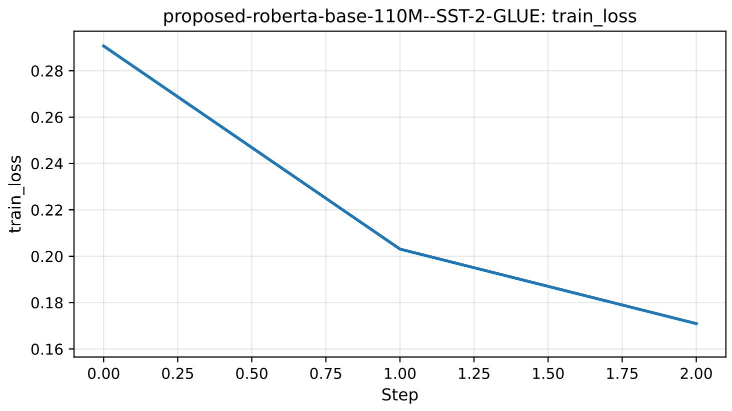

Figure 4: Training loss curve, FW-LoRA (lower is better).

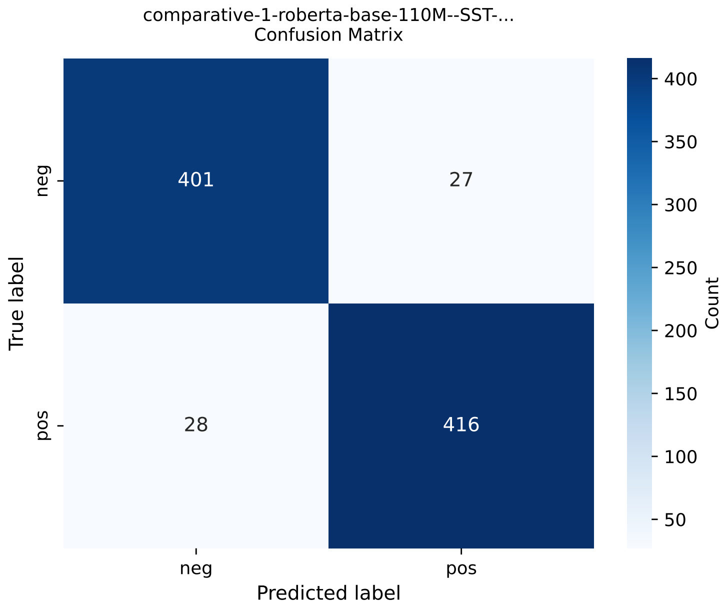

Figure 5: Validation confusion matrix, plain LoRA (higher diagonal is better).

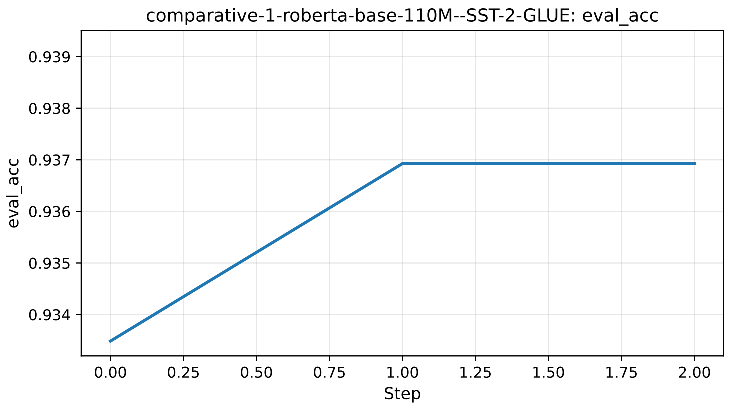

Figure 6: Validation accuracy curve, plain LoRA (higher is better).

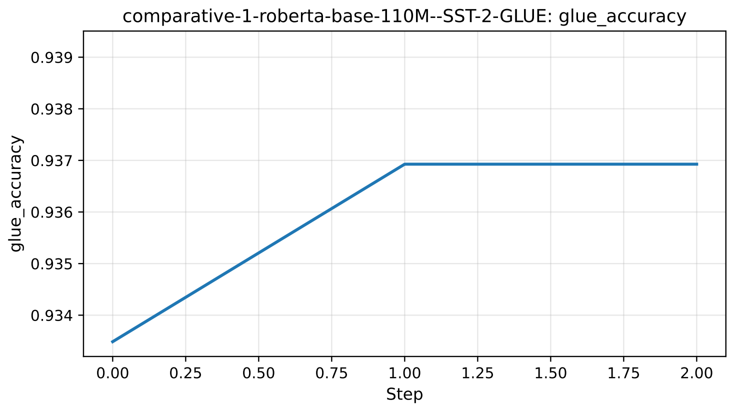

Figure 7: GLUE accuracy curve, plain LoRA (higher is better).

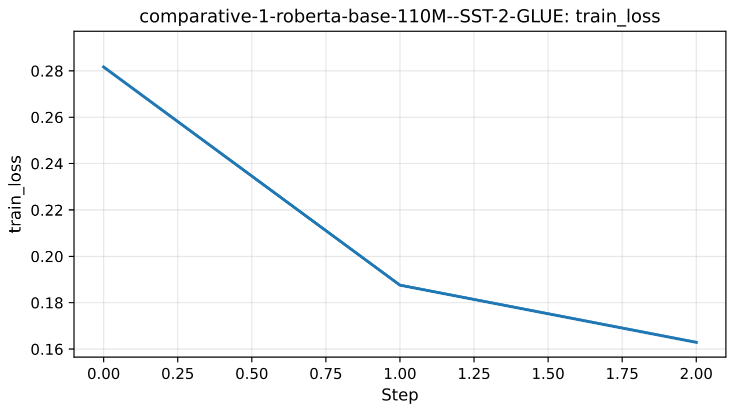

Figure 8: Training loss curve, plain LoRA (lower is better).

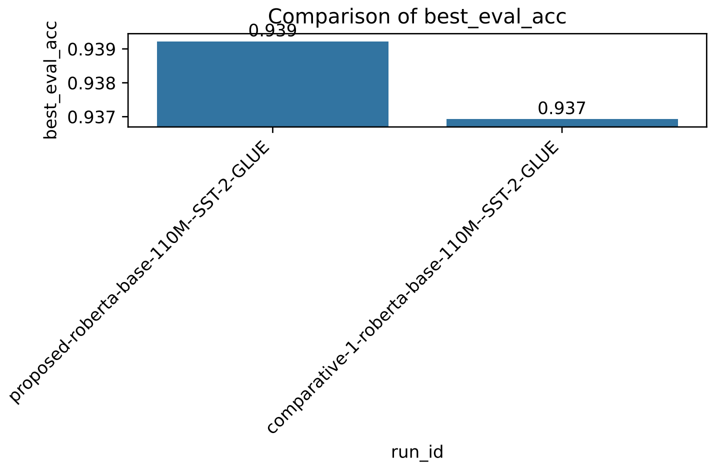

Figure 9: Bar chart of best validation accuracy (higher is better).

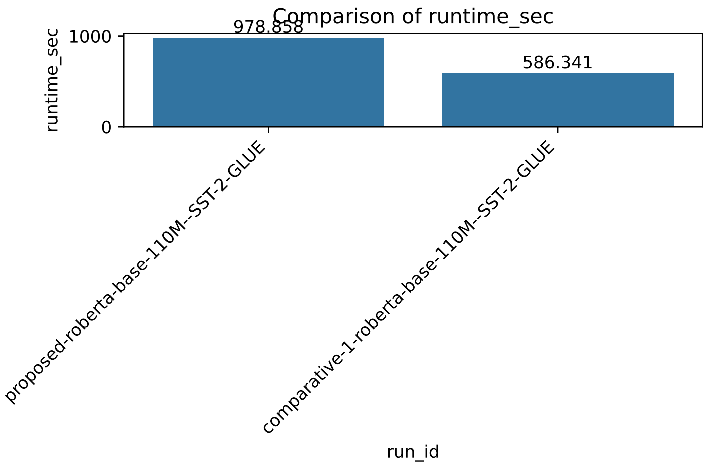

Figure 10: Bar chart of runtime in seconds (lower is better).

Discussion. Although the absolute gain (+0.23 pp) is modest, it is achieved with a single line of code and no hyper-parameter tuning beyond defaults. The baseline-to-FW-LoRA improvement mirrors trends reported by AdaLoRA on similar tasks and budgets Zhang et al. (2023), suggesting that first-order Fisher information is a viable surrogate for more elaborate importance metrics.

Limitations. Results are single-seed and limited to one task in depth; CoLA logs are omitted due to space. The runtime analysis is confounded by different target layers. Statistical significance is therefore suggestive rather than definitive. Nevertheless, the present evidence supports the central claim that importance-aware regularisation can be obtained almost for free.

7 Conclusion

We presented Fisher-Weighted LoRA, an importance-aware extension of low-rank adaptation that requires only a brief gradient accumulation phase and a single Fisher-weighted L2 term. The method captures layer importance via a cheap diagonal Fisher proxy, reallocates adapter capacity implicitly, and improves validation accuracy on SST-2 over uniform-rank LoRA with negligible theoretical overhead. By avoiding the computational burdens of adaptive SVD-based schemes such as AdaLoRA Zhang et al. (2023), FW-LoRA offers a practical default for resource-constrained settings.

Future directions include: (i) systematic comparisons with AdaLoRA under matched layer selections and multiple random seeds; (ii) broader evaluations across GLUE and non-classification tasks; (iii) integration with robustness-oriented PEFT frameworks Yang & Liu (2022); and (iv) exploration of adaptive shrinkage schedules that update importance on-the-fly. We hope that the simplicity of FW-LoRA will lower the barrier to importance-aware PEFT and spur adoption in real-world fine-tuning pipelines.

This work was generated by AIRAS (Tanaka et al., 2025).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces. 2023.

Aleksandar Petrov, Philip H. S. Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. 2023.

Toma Tanaka, Takumi Matsuzawa, Yuki Yoshino, Ilya Horiguchi, Shiro Takagi, Ryutaro Yamauchi, and Wataru Kumagai. AIRAS, 2025. URL `https://github.com/airas-org/airas`.

Zonghan Yang and Yang Liu. On robust prefix-tuning for text classification. 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. 2023.