NormAL LoRA: What is the perfect size?

Aastik[1], Meghana Topu[1,], Chinmay Kulkarni[1], Pragya Paramita Sahu[1]

[1]Samsung Research Institute - India, Bangalore

Correspondence: aastik.1@samsung.com

Abstract

Large Language Models (LLMs) are pivotal in enabling intelligent experiences across various applications, from summarization to advanced content organization and retrieval functionalities. However, deploying LLMs for diverse tasks is fundamentally constrained by memory and compute limitations, making it impractical to fine-tune separate models for each task. Parameter-Efficient Fine-Tuning (PEFT) methods like Low-Rank Adaptation (LoRA) offer a scalable solution for multi-task LLM deployment. Despite its potential, LoRA faces challenges in selecting optimal ranks and layers for each task-model pair, often resulting in inefficiencies and unnecessary parameters. We introduce **Norm A**daptive **L**ocalized (**NormAL**) **LoRA**, a novel variant that employs rank-norm regularization to dynamically determine the optimal rank for each weight matrix, ensuring adaptation is concentrated where it is most impactful. Our approach **reduces adapter parameters by 37%** while preserving full fine-tuning performance, making NormAL LoRA a transformative tool for enabling efficient, scalable, and space-constrained AI deployments across diverse industries and applications. We release our code here[1].

1 Introduction

The rapid ascent of transformer (Vaswani et al., 2017) based Large Language Models (LLMs) has reshaped what we thought was possible in natural language processing: from generating coherent paragraphs to answering complex questions, these billion-parameter models continue to push state-of-the-art performance (Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023). They are increasingly powering intelligent features in various device-based ecosystems, underscoring the need for scalable, on-device LLM deployment (Rajbhandari et al., 2022; Xu et al., 2023b; Sahu et al., 2024). Yet, their sheer size brings a practical challenge—fine-tuning for a downstream task is both compute and memory intensive, making it difficult for most users to customize these models without massive resources.

Parameter-Efficient Fine-Tuning (`peft`) methods offer an elegant solution by keeping the pre-trained weights frozen and learning only a small number of additional parameters (Ding et al., 2023). Among these, Low-Rank Adaptation (Hu et al., 2022) stands out: by factorizing weight updates into two low-rank matrices, it reduces the trainable footprint dramatically while preserving accuracy. But a new question arises: what rank should each adapter use? Too low, and the model underfits; too high, and we waste precious budget. Worse, different layers often need very different capacities, turning rank selection into an expensive grid search over dozens—or even hundreds of combinations. Identifying the optimal rank configuration for different layers manually is computationally expensive (Zhang et al., 2024; Liang et al., 2021).

To address this problem, we propose **Norm A**dpative **L**ocalized (**NormAL**) **LoRA** to find the optimal rank for each weight matrix. Central to our approach is a new rank-norm regularization mechanism, which promotes knowledge concentration within the adapter weights, enabling further compression through effective layer clipping.

Our contributions:

- We propose a novel knowledge concentration technique via rank-based norm regularization.
- We introduce NormAL-LoRA, which dynamically allocates rank to each layer module based on its estimated importance, enabling more efficient and adaptive fine-tuning.
- We show NormAL LoRA outperforms SOTA methods across multiple language understanding and generation benchmarks, highlighting its potential to unlock scalable and space-constrained LLM applications on-device.

[1]https://github.com/jingax/NormAL

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 19716–19731
November 4-9, 2025

2 Related work

We provide a brief summary of categories of PEFT and specifically Low Rank Adaptation (LoRA), and then highlight the gap our work addresses. The PEFT methods are primarily categorized into:

Additive modules: Prompt-tuning prepends a small set of trainable "soft prompts" to the model input while keeping all pretrained weights frozen (Lester et al., 2021). P-Tuning v2 (Liu et al., 2021) improves scalability to larger models. Adapters insert lightweight bottleneck layers between Transformer sub–layers, learning task-specific transformations with only a few million extra parameters (Houlsby et al., 2019; Karimi Mahabadi et al., 2021).

Selective Updates: Selective fine-tuning restricts updates to particular subsets of model parameters. BitFit updates only the bias terms of each layer, reducing trainable parameters to under 0.1% of the original model (Zaken et al., 2021). Other variants such as LayerNorm tuning (Li et al., 2022) or attention-only adaptation (Ben-Zion et al., 2023) achieve high efficiency with lower representational flexibility.

Low-Rank Reparameterizations: Low Rank Adaptation (Hu et al., 2022) factorizes each weight update into two low-rank matrices, cutting the number of trainable parameters by orders of magnitude with minimal performance degradation. Variants such as LLaMA-Adapter (Zhang et al., 2023c) and LoRA++ (Xu et al., 2023a) have extended LoRA to multi-modal and multilingual domains.

Hybrid & Dynamic Schemes: Hybrid methods combine elements of additive and low-rank approaches. Prefix-LoRA jointly learns soft prompts and LoRA adapters (He et al., 2023), while UniPELT unifies prompt, adapter, and sparse-update strategies in a single framework (Guo et al., 2023). More recently, adaLoRA dynamically allocates per-layer adapter rank based on gradient-norm statistics, optimizing parameter utility (Zhang et al., 2024). DiffFit (Gaur et al., 2023) further proposes fine-tuning by only training residuals of LoRA weights for improved generalization.

Among all the techniques listed, LoRA has emerged as a particularly powerful technique for efficient fine-tuning of pre-trained models, particularly in scenarios where computational resources are limited. It achieves task-specific adaptation by injecting low-rank trainable matrices into the frozen weight matrices of a pre-trained model. This approach allows the model to adapt to new tasks while preserving the original parameters of the base model. Mathematically, LoRA is expressed as:

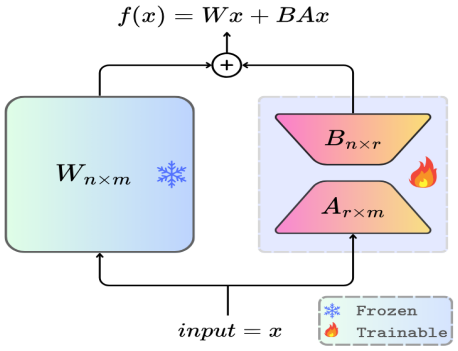

Figure 1: Representation of Low-rank adaptation

$$f(x) = W^{n\times m}x + (B^{n\times R} \times A^{R\times m})x \quad (1)$$

Here, W represents the frozen pre-trained weight matrix, while A and B are low-rank matrices of rank R. This formulation effectively introduces a low-rank adaptor of size R (Fig. 1), enabling fine-tuning by updating only the adapter weights. However, this traditional perspective treats LoRA as a unified adaptor, which can obscure the distinct contributions of its individual rank vectors towards specific tasks across various layers (Ding et al., 2023; Dai et al., 2022).

Prior studies have applied heuristic pruning to LoRA adapters—such as those of (Zhou et al., 2024; Zhang et al., 2023a; Li et al., 2025; Liu et al., 2024; Chang et al., 2025; Bhardwaj et al., 2024; Renduchintala et al., 2023; Damirchi et al., 2025)—showing that one can significantly reduce trainable parameters with negligible loss in performance; these methods typically rely on either singular-value decomposition to rank adapter subspaces or on gradient-norm metrics to select adapters for removal.

3 Proposed Methodology

We propose a reinterpretation of LoRA by viewing it not as a single low-rank adaptor but as R individual adaptors of size 1. This perspective allows us to decompose the low-rank matrices A & B into rank vectors v_r^a & v_r^b, each of size $n \times 1$ and $m \times 1$, respectively. By expressing the adapted model in terms of these rank vectors, we gain a more granular understanding of LoRA's adaptation process. Equation 1 can be reformulated as:

$$f(x) = Wx + \sum_{r=1}^{R} v_r^b v_r^a x \quad (2)$$

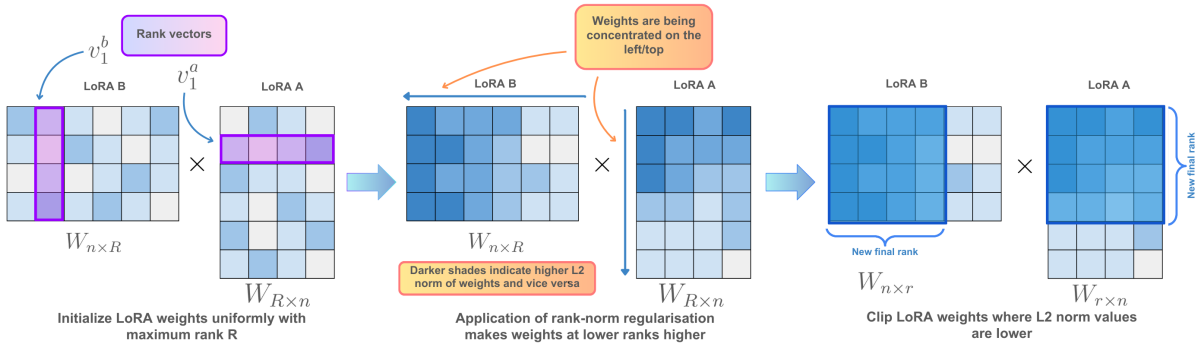

Figure 2: Stages of NormAL LoRA for a single layer. In the first step the weights are initialized uniformly. Then by application of our novel rank-norm regularization the L2 norms of the LoRA weights are shifted towards one end (top for LoRA A and left for LoRA B). We use norm as a surrogate of layer importance and clip/remove part of LoRA matrix where L2 norms are lower thus obtaining a new smaller rank LoRA for that particular layer.

By treating LoRA as a collection of rank-1 adaptors (as above), we can better analyze and optimize its performance, particularly in scenarios where computational efficiency and generalization are critical. This allows for an efficient rank reduction methodology by just determining the most optimal subset of rank vectors for both weight matrices. We propose a 2-stage methodology, described below, to effectively determine the most efficient subset of target vectors for the task. We first propose and demonstrate a novel regularization methodology (***Knowledge Concentration Regularization***) to effectively concentrate learning. This allows us to effectively remove redundant or low-impact rank-1 adaptors through a *Clipping* strategy.

3.1 Knowledge Concentration Regularization

Reducing adapter sizes can be most effectively achieved by grouping or concentrating the 'knowledge' into the minimum number of initial layers. This approach ensures that the most critical information is retained in the early adaptors, while less significant components can be pruned to optimize computational efficiency. To formalize this intuition, we draw inspiration from Taylor's Theorem in mathematics, which states that a function $f(x)$ can be approximated by a polynomial expansion around a point a:

$$f(x) \approx f(a)+f'(a)(x-a)+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n \quad (3)$$

Each additional term improves the approximation of $f(x)$, but with diminishing returns.

Analogously, we interpret increasing the LoRA rank as incrementally improving task performance, with each added rank contributing progressively less knowledge to the overall model. We use the norm of the adapter weights to represent the accumulated knowledge in each LoRA rank, reflecting the amount of information encoded in a given layer.

Extending from Eqn 1 and 2, we can represent LoRA matrices as a set of R *rank vectors* of size $N \times 1$. Specifically,

$$A = [v_1^a, v_2^a, \ldots, v_R^a], B = [v_1^b, v_2^b, \ldots, v_R^b] \quad (4)$$

We enforce regularization on these vectors to ensure that knowledge is concentrated in the initial rank vectors, as illustrated in Figure 2b. Our objective is to force or regularize the rank vectors such that their L2 norms decrease monotonically, i.e., $||v_i||_2 > ||v_{i+1}||_2$. (We provide justification for using L2 norm in appendix K).This decreasing order of L2 norms indicates that the most critical knowledge is being retained in the early rank vectors, while less significant contributions are progressively minimized. The regularization function used to achieve this is defined below

$$P(W_{N\times R}) = \sum_{r=1}^{R} \sqrt{r}||v_r||_2 \quad (5)$$

The coefficient $\sqrt{r}$ acts as a penalization for higher rank values, enforcing an optimal rank choice in the weight matrix. We apply this regularization for the first t_p steps in training, followed by a "cool-off" process to help the model generalize. We experiment with different choices for the penalization factor and, discussed is in section 4.3.

3.2 Weight Matrix Clipping

Once the condition of decreasing norms is satisfied, the reduction of trainable parameters can be achieved by pruning the low-impact rank vectors.

Based on the available computational constraints, the user can select the most optimal subset of the top k rank vectors. Prior to selection, we normalize all norm values by dividing them by $||v_1^a||_2 + ||v_1^b||_2$ for their respective weight matrices. This normalization ensures that the normalized norm value of all first rank vectors in each layer is 1, guaranteeing that each layer retains at least one rank vector. We propose two methods for LoRA clipping below:

We present two methods LoRA clipping first is budget constraint clipping presented in algorithm 3 and second is performance or norm value based clipping presented in algorithm 2.

3.2.1 Budget-Constrained Clipping

We define a *budget* k as the total number of rank vectors across all layers to be retained after clipping. In this constraint-based clipping method, we simply select the top k rank vectors with the highest normalized norms. By enforcing a decreasing norm condition, all retained rank vectors are naturally grouped on one side of the LoRA adaptor, enabling the remaining vectors to be efficiently clipped without disrupting the model's performance (Appendix H). An important condition to note is that the budget k does not directly correlate with number of trainable params. 2 adaptors with the same budget may differ in size due to variations in size of target modules and their corresponding rank allocations.

3.2.2 Norm threshold based Clipping

Norm-based clipping is particularly useful when the goal is to retain rank vectors based on their relative contribution to the model's performance, rather than strictly adhering to a fixed number of parameters. This method allows for dynamic selection of rank vectors that carry the most significant knowledge, ensuring that the adaptor retains its most impactful components (Appendix G). To achieve this, we define a threshold $\lambda \in [0, 1]$ and retain all rank vectors satisfying:

$$\frac{||v_i^a||_2 + ||v_i^b||_2}{||v_1^a||_2 + ||v_1^b||_2} > \lambda$$

While norm-based clipping provides less control over the final adaptor size, as the outcome depends on the distribution of norm values and can vary across different random seeds, it offers flexibility in adapting to the specific characteristics of the model. To comprehensively evaluate the efficacy and trade-offs of each method, we conduct experiments with both norm-based and constraint-based clipping approaches, detailed in Sec 4.2.

Algorithm 1 NormAL LoRA

Require: LoRA Model M, Train steps T, clipping stage t_c, warm up stage t_p, budget k
Ensure: $0 < t_p \leq t_c < T$
  for $t \leftarrow 1$ to T **do**
    $L \leftarrow M(X_t)$
    if $t < t_p$ **then**
      $L \leftarrow L + \eta_p P(M)$ ▷ Add rank penalty
    end if
    if $t = t_c$ **then**
      $M \leftarrow CLIP(M, S)$ ▷ Clip LoRA ranks
    end if
    Update M using Loss L
  end for
  return M

For all experiments detailed in Section 5, we adopt a consistent two-stage approach: knowledge concentration for first t_p steps followed by weight clipping at step t_c. After this step, we continue training the modified LoRA weights normally to ensure they align closely with the task at hand. Algorithm 1 formalizes the complete approach. In the subsequent ablations (Sec 4), we explore the impact of various design choices and hyperparameters to determine the most optimal configuration for the proposed NormAL LoRA methodology.

4 Ablations

4.1 When to Clip? Determining t_p and t_c

NormAL LoRA, introduces two critical hyperparameters: t_p, which controls when regularization is terminated, and t_c, which determines when the LoRA weights are clipped. We observe that NormAL LoRA is sensitive to the choice of these scheduling points and suboptimal values for t_p and t_c can noticeably degrade performance.

To mitigate this, we explore heuristic-based strategies for adaptively setting t_p and t_c. Specifically, we monitor the exponentially weighted moving average of the step-wise change in loss:

$$\begin{aligned} \Delta_t &= |L_t - L_{t-1}| \\ \hat{\Delta}_t &= \alpha \cdot \hat{\Delta}_{t-1} + (1 - \alpha) \cdot \Delta_t \end{aligned} \tag{6}$$

Here, L_t is the loss at step t, Δ_t is the absolute change in loss, $\hat{\Delta}_t$ is the smoothed estimate at step t, and $\alpha \in [0, 1)$ is the smoothing factor, set to 0.9.

When $\hat{\Delta}_t$ falls below a predefined regularization threshold(λ_p), we terminate regularization (marking t_p). After resetting the moving average, we continue monitoring. When $\hat{\Delta}_t$ again falls below the clipping threshold(λ_c), we clip the LoRA weights (marking t_c). This adaptive scheduling significantly improves training stability and reduces the need

Method	Avg rank	BLEU	NIST	METEOR	ROUGE_L	CIDEr
Budget clipping	4	0.579	7.493	0.372	0.624	1.762
Norm thresholding ($\lambda = 0.95$)	3.89	0.572	7.410	0.367	0.608	1.693
Budget clipping	5	0.589	7.537	0.376	0.63	1.779
Norm thresholding ($\lambda = 0.85$)	7.24	0.578	7.440	0.371	0.614	1.687

Table 1: Comparison of budget clipping and norm value threshold clipping

for manual tuning. We present detailed results in Appendix F. Please note that while this heuristics provides easier and more stable clipping points, the optimal performance is obtained by manual tuning and hence for all experiments, we use this heuristic to determine the correct range and then determine the optimal values of t_p and t_c with a fine-grained manual check.

4.2 Budget vs Norm Thresholding

We first evaluate the performance of two clipping methods — Budget-based (Algorithm 3) and norm-thresholding (Algorithm 2)—under identical training conditions on the NLG task (Section 5.2.1), as the choice of clipping methodology is a critical factor in determining overall efficiency and performance. The results, summarized in Table 1, reveal that norm-based clipping performs comparably to or, in some cases, slightly worse than budget-based clipping. However, norm-thresholding presents challenges in controlling the final adapter size, often requiring multiple training rounds to fine-tune λ to achieve a desired target size. In contrast, budget-based clipping offers more predictable control and efficiency, making it a more practical choice for our experiments. As a result, all subsequent experiments involving NormAL LoRA in this paper utilize the budget-based clipping method.

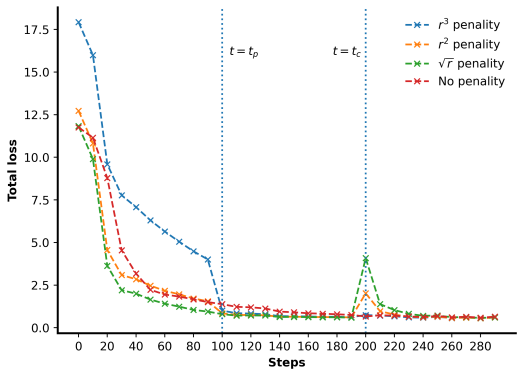

Figure 3: Evaluation of various penalizing factors

4.3 Regularization Penalization Function

We demonstrate our proposed regularization function in Eqn 5, wherein we use $\sqrt{r}$ as the penalization function. To explore the impact of different penalizing functions, we experiment with alternatives such as r, r^2, r^3, and $log\ r$, observing the impact of a diverse range of growth rates and behaviors. We present the loss curves for these functions alongside the loss plot without NormAL LoRA in Figure 3. For readability, the plots for $log\ r$ and r are omitted from the main text as they closely resemble the $\sqrt{r}$ curve (provided in the Appendix A).

Across all experiments, we observe that the loss decreases steadily from 0 to t_p. At t_p, the loss drops significantly for all penalizing factors because the regularization loss is removed. However, this trend doesn't apply to the $\sqrt{r}$ curve, where the regularization loss is already close to 0 at this step due to its smaller penalty factor. We also notice that for smaller penalizing functions, the initial loss is lower than the curve with no penalty (Original LoRA). This is because, before clipping at step t_c, NormAL LoRA has a rank of 8 (maximum rank), while the no penalty curve has a rank of 2. The larger number of trainable parameters in NormAL LoRA helps it offset the extra loss caused by rank regularization. On the other hand, when the penalizing factor is too high (e.g., r^3), the benefit of having more trainable parameters isn't enough to offset the very high regularization loss.

Another interesting observation is that at step t_c, the loss for some penalty factors spikes. We find that smaller penalty factors result in higher spikes, as pruning the LoRA weights at this step leads to a temporary loss of knowledge, which is eventually recovered in subsequent steps. Conversely, for very large factors like r^3, there is no significant spike because these factors have already reduced many rank vectors to zero, making pruning or clipping them less impactful on the total loss.

Considering the above observations, we choose $\sqrt{r}$ as the *most stable penalizing function* for the proposed regularization methodology.

Method	#Params	MNLI	SST-2	CoLA	QQP	QNLI	RTE	MRPC	STS-B	All
		m/mm	Acc	Mcc	Acc/F1	Acc	Acc	Acc	Corr	Avg
Full FT †	184M	89.90/90.12	95.63	69.19	92.40/89.80	94.03	83.75	89.46	91.60	88.09
HAdapter †	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter†	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
LoRA$_{r=2}$†	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA†	0.32M	**90.66/90.70**	95.80	70.04	91.78/89.16	94.49	87.36	90.44	91.63	88.86
Ours$_{r=2}$	0.32M	90.44/90.18	**95.99**	**70.64**	**92.04/89.44**	**94.56**	**88.81**	**90.93**	91.58	**89.19**

Table 2: DeBERTaV3-base on GLUE val set with different fine-tuning methods. Result reported for NormAL LoRA(Ours) with comparable trainable parameters by keeping avg adapter rank 2 † Taken from (Zhang et al., 2023b)

5 Experiments, Results & Observations

We implement NormAL LoRA by modifying the HuggingFace `peft` library's LoRA backend. For a simplified implementation, we apply a mask to the clipped weights during the forward pass and use gradient hooks in the backward pass to mask pruned component gradients. All experiments are conducted on NVIDIA A100 GPUs with 80GB memory. Our current implementation supports only single-GPU training; multi-GPU behavior may differ and is left for future work.

We evaluate our approach across various language understanding and generation tasks to prove model efficacy and robustness across general purpose capabilties. Additionally, we evaluate and visualize how the rank distribution is affected with the implementation of NormAL LoRA (Figure 5). Finally, we evaluate the training costs of implementation to further prove the efficacy and efficiency of the proposed approach in section 5.4.

5.1 Natural Language Understanding

To rigorously evaluate NormAL LoRA, we conducted experiments on the GLUE benchmark (Wang et al., 2018), following the experimental protocol established by AdaLoRA (Zhang et al., 2023b). We fine-tune public DeBERTa-v3-base model(12 layers & 86M backbone parameters). The model was evaluated on the various standard natural language understanding (NLU) tasks, detailed in Table 12. These tasks cover a broad range of linguistic phenomena, including syntax, semantics, and inference, making GLUE a comprehensive benchmark for evaluating general-purpose language representations. Our goal was to compare the performance and parameter efficiency of our method against SOTA methods like AdaLoRA under identical training conditions.

Table 2 presents the performance of NormAL LoRA on the GLUE (Wang et al., 2018) benchmark using a DeBERTaV3-base model with an average adapter rank of 2. Compared to existing parameter-efficient fine-tuning methods, including LoRA and AdaLoRA, our method achieves state-of-the-art results on a majority of tasks, notably outperforming baselines on SST-2, CoLA, QQP, QNLI, RTE, and MRPC tasks. This demonstrates that proposed norm-based rank allocation and knowledge concentration regularization effectively enhances task performance while maintaining a low parameter footprint, across all target tasks. Sginificantly, NormAL LoRA outperforms full model finetuning with just 0.16% of original trainable parameters.

5.2 Natural Language Generation

To further evaluate NormAL LoRA efficacy across multiple language tasks and architectures, we also evaluate its performance on various language generation tasks. Specifically, we evaluate on the E2E NLG Challenge dataset (Novikova et al., 2017) with TinyLlama-1.1B model and CNN/DailyMail summarization dataset (Hermann et al., 2015) with BART-large.

5.2.1 E2E NLG Challenge

The E2E NLG Challenge (Novikova et al., 2017) is a benchmark for data-to-text generation, where models are trained to generate natural language utterances from structured meaning representations (MRs). The task evaluates how well a model can fluently and accurately verbalize structured input into coherent sentences.

We evaluate both methods using TinyLlama-1.1B with ranks ranging from 1 to 8. For fair comparison, NormAL LoRA is configured with an equivalent average rank. All hyperparameters including learning rate, batch size, LoRA scaling factor (α), and number of epochs, are kept constant across experiments. Each setting is run 5 times with different seeds, and we report the mean and standard deviation across these runs in Figure 4.

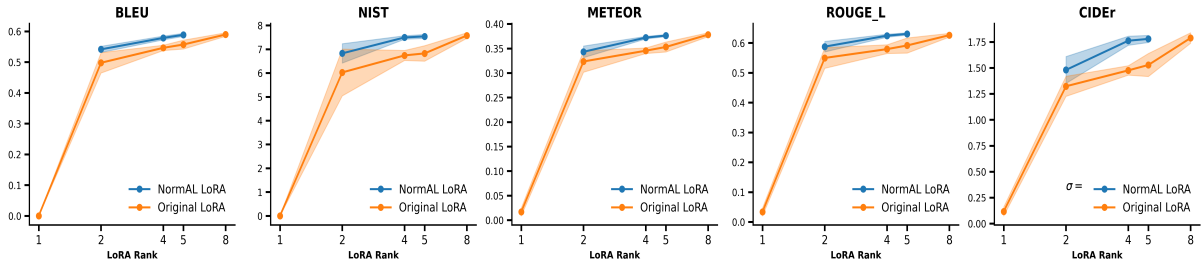

Figure 4: Comparison of standard LoRA (Hu et al., 2022) and NormAL LoRA (ours) on the **E2E NLG Challenge**. Experiments use TinyLlama-1.1B trained for a single epoch. The rank for NormAL LoRA reflects the average across layers, while baseline LoRA uses a uniform rank. The number of trainable parameters is held constant.

Method	Rank	BLEU		NIST		METEOR		ROUGE_L		CIDEr	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
LoRA	1	0.0	0.0	0.0	0.0	0.017	0.008	0.033	0.013	0.117	0.043
Ours		0.0	0.0	0.0	0.0	0.017	0.008	0.033	0.013	0.117	0.043
LoRA	2	0.498	0.033	6.024	0.982	0.323	0.022	0.55	0.034	1.323	0.097
Ours		0.541	0.01	6.825	0.405	0.343	0.012	0.588	0.018	1.48	0.129
LoRA	4	0.546	0.009	6.739	0.209	0.345	0.006	0.579	0.015	1.476	0.045
Ours		0.579	0.006	7.493	0.073	0.372	0.003	0.624	0.005	1.762	0.044
LoRA	5	0.557	0.014	6.823	0.324	0.353	0.01	0.591	0.025	1.528	0.111
Ours		0.589	0.005	7.537	0.082	0.376	0.002	0.63	0.004	1.779	0.034
LoRA	8	0.59	0.006	7.575	0.116	0.378	0.004	0.626	0.006	1.788	0.05
Ours		0.59	0.006	7.575	0.116	0.378	0.004	0.626	0.006	1.788	0.05

Table 3: Accuracy and Standard Deviation across different ranks for NLG tasks

As can be seen, NormAL LoRA with an average rank of 5 matches the performance of baseline LoRA at rank 8, achieving a **37.5% reduction** in adapter size. Across all tested ranks, proposed method consistently delivers better or equivalent performance with fewer parameters.

In addition, NormAL LoRA offers improved training stability as seen with the lower standard deviation across different runs. This is particularly evident at lower ranks: for example, at rank 2, variance across runs is significantly lower. Detailed standard deviation values have been noted in Table 3, showing a 2–3$\times$ **reduction in standard deviation** across most metrics. We extend our experiments to scale upto 7B parameneter count using Mistral-7B model (Appendix J) showcasing generalizability to very large model sizes.

5.2.2 Text Summarization

The CNN/DailyMail dataset (Hermann et al., 2015) is a popular benchmark for abstractive summarization, tasked with producing concise and coherent summaries of news articles while retaining their core information. We employ the BART-large model for this task, given its high performance for text generation tasks such as summarization. We assess NormAL LoRA's performance on this dataset, comparing it to baseline LoRA, with a focus on the quality of concise summaries produced and the reduction in trainable parameters.

The evaluation is conducted using standard metrics such as **ROUGE-1**, **ROUGE-2**, and **ROUGE-L**, which assess the overlap of n-grams and the longest common subsequences between the generated summaries and the reference summaries. Detailed results have been noted in Table 4.

Although NormAL LoRA use average rank 5 and the original LoRA use the uniform rank of 2, NormAL LoRA places more emphasis on smaller target modules, which leads to a **reduction in the number of parameters of 12%**. Despite this difference in module size, NormAL LoRA achieves comparable performance to the original model across all ROUGE metrics.

Method	Params (M)	ROGUE-1	ROGUE-2	ROGUE-L
LoRA	0.64	0.413	0.197	0.293
Ours	0.56	0.410	0.196	0.292

Table 4: Summarization performance on CNN dataset

Across all the tasks and experimental setups, we note that NormAL LoRA consistently outperforms baseline LoRA based methodologies while retaining the smallest number of parameter sizes.

5.3 Image Understanding

To assess the generalizability of NormAL LoRA beyond language tasks, we extend our experiments to object classification using standard vision benchmarks (CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Food-101 (Bossard et al., 2014)) and Object detection on COCO (Lin et al., 2014).

5.3.1 Image classification

We fine-tune widely used Vision Transformer (ViT) variants (Dosovitskiy et al., 2020), including ViT-Tiny and ViT-Base, and compare performance against the original LoRA baseline.

As shown in Table 5, NormAL LoRA delivers the largest improvements on smaller models, particularly ViT-Tiny. It outperforms original LoRA across all datasets on this architecture, with notable gains of up to $+0.26$ on Food-101 and $+0.18$ on CIFAR-100. These improvements are especially significant for edge applications, where compact models are favored due to tight memory and compute constraints.

On larger backbones like ViT-Base, NormAL LoRA remains competitive, with performance differences typically under 0.1% compared to original LoRA. This indicates that while both methods perform similarly at scale, NormAL LoRA offers distinct benefits in constrained environments.

Method	Params	cifar-10/100	food101
VIT-tiny			
Ours$_{r=1.5}$	20.7K	95.87/81.50	77.62
Ours$_{r=2}$	27.6K	**95.95/81.68**	**78.06**
LoRA$_{r=2}$	27.6K	95.87/81.62	77.8
VIT-base			
Ours$_{r=1.5}$	82.9K	98.49/90.60	**86.51**
Ours$_{r=2}$	110.5K	98.49/90.57	**86.51**
LoRA$_{r=2}$	110.5K	**98.53/90.63**	**86.51**

Table 5: Comparison of ViT-Tiny and ViT-Base models. Results highlight the higher efficacy for smaller ViT-Tiny model across multiple vision-language tasks.

Method	AP	AP50	AP75	AR1	AR10	AR100
LoRA$_{r=8}$	40.05	59.09	42.33	32.66	48.78	50.37
Ours$_{r=5}$	40.14	59.23	42.54	32.59	48.81	50.42

Table 6: Object detection task on COCO dataset

Importantly, these gains are achieved with reduced parameter overhead. Even with a lower average rank (e.g., 1.5), NormAL LoRA matches or exceeds the performance of original LoRA at rank 2, highlighting its efficiency in parameter usage. This makes it particularly well-suited for on-device fine-tuning and continual learning on mobile or embedded platforms, where minimizing trainable parameters is essential.

5.3.2 Object detection

To further evaluate the efficacy of NormAL LoRA, we integrate it with the YOLOS-base model (86M) (Fang et al., 2021), which comprises approximately 86 million parameters, and assess its performance on the object detection task using the COCO2017 dataset (Lin et al., 2014). Our experimental results (shown in table 6) enforce the erstwhile noted trend. NormAL LoRA is able to obtain adapter size reduction of up to 35%, while maintaining comparable performance. The generic efficacy of NormAL LoRA, across different input types, reinforces the impact that can be brought in for any transformer-powered application across domains.

5.4 Implementation Cost

We measure the end-to-end training time for NormAL LoRA versus standard LoRA and observed a 2 – 4% increase in total training time—an overhead we consider acceptable. This modest slowdown arises from two sources:

1. **Additional regularization:** NormAL LoRA applies its extra penalty only during the first t_p optimization steps, so the regularization cost is incurred solely in those initial iterations.

2. **Increased parameter count:** In our current implementation, we mask rather than prune adapter weights (for ease of implementation). As a result, a NormAL adapter with average rank 2 (maximum rank 8) carries four times as many masked parameters as its LoRA counterpart. (Pruning masked weights instead of masking them would eliminate this overhead.)

Given the improved performance with a much smaller parameter size, we consider this increase

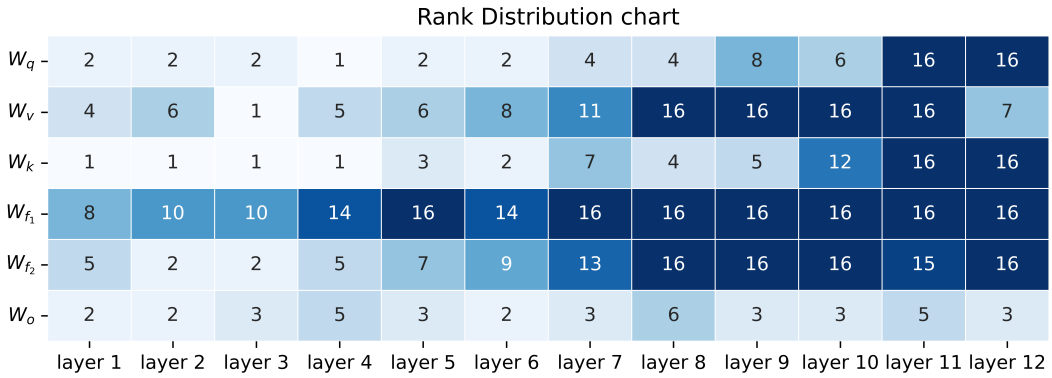

Figure 5: Rank distribution of final adapter using NormAL LoRA applied on DeBERTaV3-base trained for MNLI task with an average rank of 8 and max rank of 16.

in training time an acceptable cost to attain more efficient, smaller adapters to target real-time, on-device applications.

5.5 Rank distribution by NormAL LoRA

We plot the rank distribution assigned by NormAL LoRA for a natural language understanding task (MNLI) in Fig 5. The final ranks assigned to each layer align with the expectation that the later layers play a more significant role in the model's final output. As shown, earlier (shallower) layers are assigned lower ranks compared to deeper layers. Additionally, we observe that different modules are assigned varying levels of importance. Specifically, the W_K and W_O layers (corresponding to the `key_proj` and `output.dense`) receive the least importance (least ranks), while the W_{f1} (`intermediate.dense`) and W_{f2} (`attention.output.dense`) layers are assigned the highest ranks. We can attribute this relatively high weightage to the dense layers having more parameters hence playing more important role in a transformer.

6 Conclusion

In this work, we introduce **NormAL** (**Norm** **A**daptive **L**ocalized) LoRA, which dynamically allocates optical rank to each layer depending on its impact on the downstream task. We introduce a novel knowledge concentration regularization technique with multi-constrained clipping mechanisms. The proposed regularization scheme concentrates most of the model's learned information into the early columns of each adapter. This structure allows the remaining columns to be pruned, yielding a much smaller adapter.

To comprehensively assess the versatility and generalizability of NormAL LoRA, we conduct a thorough evaluation across a diverse range of natural language processing (NLP) tasks, including both Natural Language Understanding (NLU) and Natural Language Generation (NLG), as well as computer vision tasks such as object detection and image classification. Under equal average-rank budgets, NormAL LoRA matches or exceeds the performance of existing baselines while also reducing run-to-run variance. Specifically, Normal LoRA provides a 37% parameter reduction with no performance drop across NLG baselines. Our experiments span models of varying sizes, from 86 million to 7 billion parameters, thereby encompassing a broad spectrum of computational complexities. Furthermore, we investigate the applicability of NormAL LoRA across different model architectures, including encoder-only, decoder-only, and encoder-decoder configurations. This extensive evaluation aims to provide a holistic understanding of NormAL LoRA's efficacy and its potential as a universally applicable parameter-efficient fine-tuning method. By demonstrating its effectiveness across a wide range of tasks and model architectures, we establish NormAL LoRA as a robust and versatile technique for adapting large-scale pre-trained models to diverse downstream applications. As the capabilities of LLMs increase and more on-device solutions are built on top of base LLMs, Normal LoRA will be crucial in creating the most optimal and effective adapters in an on-device, highly resource constrained environment.

[1]Language and grammar in this manuscript were refined with the assistance of ChatGPT.

Limitations

While the proposed dynamic variant of LoRA demonstrates consistently strong performance, several minor limitations remain. First, the method is sensitive to the scheduling of two key hyperparameters: the regularization termination point (t_p) and the clipping step (t_c). Optimal performance was consistently observed only when t_p and t_c were aligned with empirically favorable values. Deviating significantly from these points led to reduced performance, indicating that careful tuning is essential. Second, although our implementation masks weights and gradients instead of pruning them outright, this choice leaves potential efficiency gains unrealized. A more sophisticated implementation could incorporate actual pruning for improved runtime and memory efficiency, which we leave to future work. Finally, the maximum rank remains a manually specified hyperparameter. Automating its selection, potentially via validation-based adaptation or meta-learning, could enhance the method's usability and generalization.

References

Idan Ben-Zion, Roy Schwartz, and Amir Globerson. 2023. Q-adapters: Quantized parameter efficient fine-tuning. *arXiv preprint arXiv:2305.11235*.

Kartikeya Bhardwaj, Nilesh Prasad Pandey, Sweta Priyadarshi, Viswanath Ganapathy, Shreya Kadambi, Rafael Esteves, Shubhankar Borse, Paul Whatmough, Risheek Garrepalli, Mart Van Baalen, Harris Teague, and Markus Nagel. 2024. Sparse high rank adapters (shira). *arXiv preprint arXiv:2406.13175*.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and 1 others. 2020. Language models are few-shot learners. *NeurIPS*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity – multilingual and cross-lingual focus. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Huandong Chang, Zicheng Ma, Mingyuan Ma, Zhenting Qi, Andrew Sabot, Hong Jiang, and H.T. Kung. 2025. Elalora: Elastic learnable low-rank adaptation for efficient model fine-tuning. *arXiv preprint arXiv:2504.00254*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and 1 others. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Xiang Dai, Yitong Xu, Kaijie Lin, and 1 others. 2022. Knowledge neurons in pretrained transformers. *ICLR*.

Hamed Damirchi, Cristian Rodriguez-Opazo, Ehsan Abbasnejad, Zhen Zhang, and Javen Qinfeng Shi. 2025. The quest for winning tickets in low-rank adapters. In *International Conference on Learning Representations (ICLR) 2025*.

Ming Ding, Yuxian Sun, Yiming Wang, and 1 others. 2023. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2307.07017*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Yuxin Fang, Xinggang Liao, Wenyu Liu, Alan Yuille, Xinge Yang, Ruiwei Gong, Xinyu Wang, and Wenhai Liu. 2021. You only look at one sequence: Rethinking transformer for generic object detection. *arXiv preprint arXiv:2106.00666*.

Divyanshu Gaur and 1 others. 2023. Difffit: Diffused parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2310.01405*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9.

Yifan Guo, Tong Zhou, Jie Li, and Hao Sun. 2023. Unipelt: Unified parameter-efficient and lightweight tuners. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 3100–3112.

Peng He, Xiang Liu, Hao Wang, and 1 others. 2023. Prefix-lora: A parameter efficient framework for instruction tuning. *arXiv preprint arXiv:2301.13679*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzębski, Behnam Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mohammad 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, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Rami Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

Alex Krizhevsky, Geoffrey Hinton, and 1 others. 2009. Learning multiple layers of features from tiny images.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Brian Lester, Rishi Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. 2025. Salora: Safety-alignment preserved low-rank adaptation. *arXiv preprint arXiv:2501.01765*.

Xianzhi Li, Zihang Dai, and 1 others. 2022. Branch-tuning: Efficient fine-tuning of pre-trained vision models via branching. *arXiv preprint arXiv:2202.06924*.

Chen Henry Liang, Jimmy Lin, and 1 others. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. *arXiv preprint arXiv:2101.11359*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Xiao Liu, Kaixuan Ji, Yicheng Fu, and 1 others. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *NeurIPS*.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024. Alora: Allocating low-rank adaptation for fine-tuning large language models. In *Proceedings of NAACL 2024*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

OpenAI. 2023. Gpt-4 technical report. Https://openai.com/research/gpt-4.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and 1 others. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2023. Tied-lora: Enhancing parameter efficiency of lora with weight tying. *arXiv preprint arXiv:2311.09578*.

Pragya Paramita Sahu, Abhishek Raut, Jagdish Singh Samant, Mahesh Gorijala, Vignesh Lakshminarayanan, and Pinaki Bhaskar. 2024. Pop-vqa-privacy preserving, on-device, personalized visual question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8470–8479.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Canwen Xu, Xiang Lin, and 1 others. 2023a. Exploring parameter-efficient fine-tuning for instruction tuning. *arXiv preprint arXiv:2307.06975*.

Shang Xu, Yuzhuo Liang, Tri Dao, and 1 others. 2023b. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2309.11338*.

Yonatan Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 7430–7444.

Hao Zhang, Wei Chen, and Jing Xu. 2024. adaLoRA: Adaptive lora via gradient-norm guided rank allocation. In *International Conference on Learning Representations (ICLR)*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023a. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Renrui Zhang, Yujun Xu, Yujie Zhang, and 1 others. 2023c. Llama-adapter: Efficient fine-tuning of llama for instruction following. *arXiv preprint arXiv:2303.16199*.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2024. Lora-drop: Efficient lora parameter pruning based on output evaluation. *arXiv preprint arXiv:2402.07721*.

A LOSS PLOTS

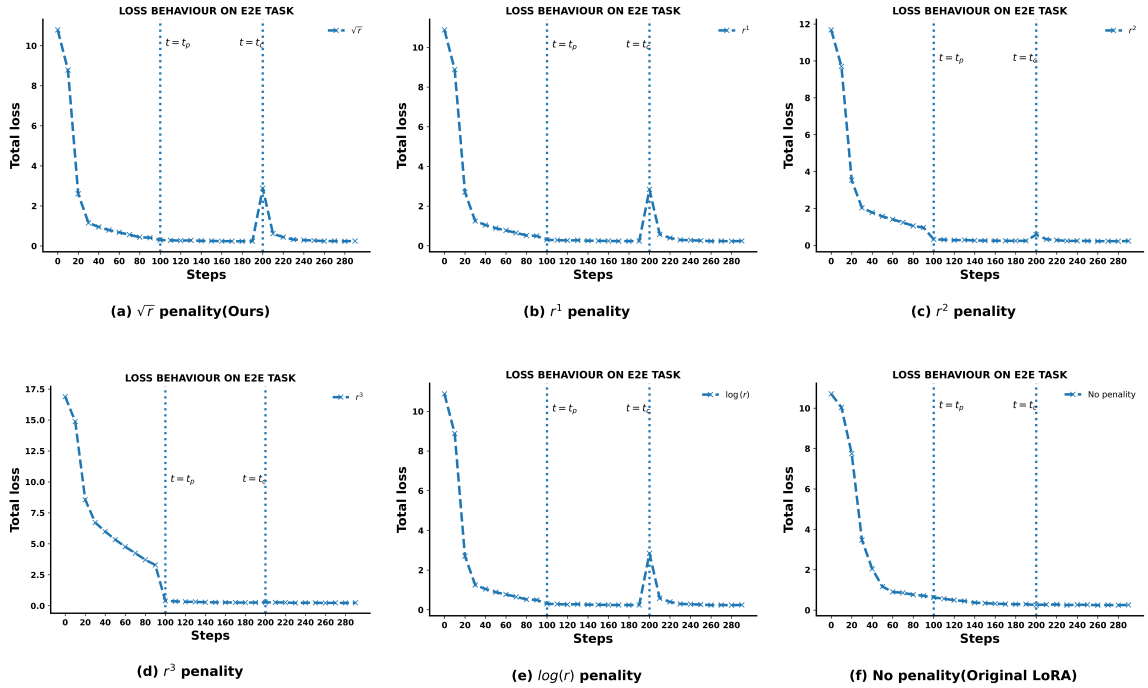

Figure 6: Comparison of training loss curves for different penalty functions on E2E challenge for a single Epoch on TinyLlama-1.1B.

B GLUE TRAINING SETUP

Task	learning rate	epochs	k	avg rank	max rank	t_p	t_c	seed
MNLI	5e-4	7	144	2	8	330	350	4
SST-2	8e-4	24	144	2	16	520	550	6
CoLA	8e-4	25	144	2	16	1005	2010	4
QQP	8e-4	5	144	2	16	330	350	4
QNLI	5e-4	5	144	2	8	400	440	4
RTE	1.2e-3	12	144	2	16	140	398	5
MRPC	1e-3	30	144	2	8	522	1044	4
STS-B	2.2e-3	25	144	2	16	330	360	6

Table 7: Detailed hyper parameter settings for GLUE benchmark on **DeBERTa-V3-Base** model

C E2E TRAINING SETUP

avg rank	k	learning rate	max rank	t_p	t_c
2	**132**	2e-5	8	329	526
4	**264**	2e-5	8	329	526
5	**330**	2e-5	8	329	526

Table 8: Detailed hyper parameter settings for training on **TinyLlama-1.1B** for **1 epoch**

D SUMMARIZATION TRAINING SETUP

Model	Task	learning rate	epochs	k	avg rank	max rank	t_p	t_c	seed
$BART_{(large)}$	**CNN**	5e-4	15	384	5	16	330	350	4

Table 9: Hyper parameter setting for Summarization task

E TRAINING TIME

Method	Time(min)
Original LoRA	**9.01**
NormAL LoRA(Ours)	10.08

Table 10: Training time comparision of NormAL LoRA and Original LoRA for **E2E challenge** on a **single epoch**

F THRESHOLD EVALUATIONS

$\hat{\Delta}_t(t_p/t_c)$	t_p/t_c	Avg Rank	BLEU	NIST	METEOR	ROUGE_L	CIDEr
2e-2/15e-3	181/407	2	0.548	6.774	0.343	0.589	1.440
		4	0.569	7.423	0.372	0.621	1.711
		5	0.565	7.338	0.367	0.618	1.668
3e-2/11e-3	115/510	2	0.550	6.849	0.345	0.601	1.467
		4	0.568	7.289	0.365	0.616	1.672
		5	0.571	7.351	0.367	0.618	1.696
5e-2/20e-3	92/118	2	0.564	7.099	0.347	0.615	1.602
		4	0.567	7.485	0.369	0.619	1.725
		5	0.574	7.315	0.366	0.624	1.692

Table 11: Results obtained on **E2E challenge** using different $\hat{\Delta}_t(t_p/t_c)$ configurations, reported with corresponding average ranks and performance scores

G Norm-Based Clipping Algorithm

Algorithm 2 Norm threshold based Clipping

Require: Model M with LoRA, Norm threshold λ

$ranks \leftarrow \{\}$ ▷ Default value 0
for $layer \leftarrow layers(M)$ **do**
  $ranks[layer] \leftarrow 0$
  for $r \leftarrow 1$ to R **do**
    $v^a \leftarrow layer_a[:][r]$
    $v^b \leftarrow layer_b[r][:]$
    $v_0^a \leftarrow layer_a[:][0]$
    $v_0^b \leftarrow layer_b[0][:]$
    if then $\frac{v^a+v^b}{v_0^a+v_0^b} > \lambda$
      $ranks[layer] \leftarrow r$
    end if
  end for
end for
for $layer \leftarrow layers(M)$ **do**
  $layer_a \leftarrow layer_a[:][: ranks[layer]]$
  $layer_b \leftarrow layer_b[: ranks[layer]][:]$
end for
return M ▷ Return LoRA with optimal ranks

H Constrained Clipping Algorithm

Algorithm 3 Budget-Constrained Clipping

Require: Model M with LoRA, LoRA adaptor budget k

$ranks \leftarrow \{\}$ ▷ Default value 0
for $i \leftarrow 1$ to k **do**
  $norms \leftarrow \{\}$ ▷ Default value 0
  for $layer \leftarrow layers(M)$ **do**
    $v^a \leftarrow layer_a[:][ranks[layer]]$
    $v^b \leftarrow layer_b[ranks[layer]][:]$
    $v_0^a \leftarrow layer_a[:][0]$
    $v_0^b \leftarrow layer_b[0][:]$
    $norms[layer] \leftarrow \frac{v^a+v^b}{v_0^a+v_0^b}$
  end for
  $s \leftarrow argmax(norms)$ ▷ Selected layer
  $ranks[s] \leftarrow ranks[s] + 1$ ▷ Increase rank
end for
for $layer \leftarrow layers(M)$ **do**
  $layer_a \leftarrow layer_a[:][: ranks[layer]]$
  $layer_b \leftarrow layer_b[: ranks[layer]][:]$
end for
return M ▷ Return LoRA with optimal ranks

I NLU Dataset Descriptions

Task	Description
MNLI (Williams et al., 2017)	Three-way classification: entailment, contradiction, or neutral.
SST-2 (Socher et al., 2013)	Sentiment classification: positive or negative.
CoLA (Warstadt et al., 2018)	Grammatical acceptability: yes or no.
QQP (Kwiatkowski et al., 2019)	Paraphrase detection: are two questions paraphrases?
QNLI (Wang et al., 2018)	Question answering: does a sentence contain the correct answer?
RTE (Giampiccolo et al., 2007)	Textual entailment: does the premise entail the hypothesis?
MRPC (Dolan and Brockett, 2005)	Paraphrase detection: are two sentences semantically equivalent?
STS-B (Cer et al., 2017)	Semantic similarity: score between 0 and 5.

Table 12: Datasets used for NLU tasks

J Scaling to 7B model

We further expand our experiments to Mistral-7B model to showcase that NormAL LoRA generalizes across all model sizes. We see that NormAL LoRA out performs the standard LoRA with rank 8 and with average rank of 5 it give performance similar to rank 8 LoRA. The trend is similar to what is observed with TinyLlama-1.1B.

Method	BLEU	NIST	METEOR	ROUGE_L	CIDEr
LoRA(Rank8)	0.4695	4.0880	0.3376	0.5584	1.0417
NormAL LoRA (avg $rank = 8$**)**	0.5856	7.0506	0.3810	0.6306	1.6385
NormAL LoRA (avg $rank = 5$**)**	0.4691	4.4528	0.3384	0.5488	0.9896

Table 13: Mistral-7b model on E2E NLG challenge

K Why use L2 norm?

Prior works like **Pruning filters for efficient convnets** and **Learning both Weights and Connections for Efficient Neural Networks** have shown that gradient norms are a good yard stick for pruning layers. We show correlation between the gradient norms and the weight norms of a LoRA adaptor to prove that L2 norm is also a good metric for pruning a layer.

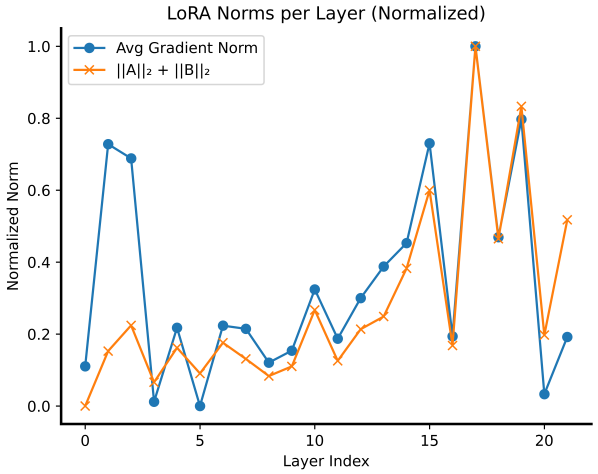

Figure 6: Comparison of Gradient norms and L2 norms across layers on E2E challenge for a single Epoch on TinyLlama-1.1B.