LoRMA: Low-Rank Multiplicative Adaptation for LLMs

Anonymous ACL submission

Abstract

Large Language Models have emerged to show remarkable capabilities in the NLP domain. Their effectiveness can mainly be attributed to their ability to adapt to an array of downstream tasks. However, generally, full fine-tuning is a computationally expensive job. To mitigate this, many techniques have been developed that prime efficiency, a prominent one being Low-Rank Adaptation (LoRA). However, LoRA and its variants employ re-parametrized additive updates. In this paper, we propose **Low Rank Multiplicative Adaptation** (**LoRMA**), which shifts the paradigm of additive updates to a much richer space of matrix multiplicative transformations. We tackle challenges such as computational complexity and rank inhibition by strategically ordering matrix operations and introducing rank inflation strategies. We conduct extensive experiments to show the effectiveness of our approach in terms of evaluation metrics and computational costs.

1 Introduction

Large Language Models (LLMs) have demonstrated strong performance across various NLP benchmarks (Fourrier et al., 2024). Though LLMs have shown impressive generalization capabilities (for example, via In-context learning (Dong et al., 2024)), sometimes these tend to have lower performance on some niche or low-resource tasks, thus requiring task-specific fine-tuning. LLMs usage follows a pre-train and fine-tune paradigm (Zhao et al., 2023), where the model is trained on a massive amount of text in an unsupervised fashion, and subsequently, the model is fine-tuned for some specific tasks/domains. Given the size of these models (order of billions of parameters), it may not always be feasible to fine-tune the entire model due to high computational costs. In recent years, a new class of techniques (referred to as PEFT (Parameter Efficient Fine Tuning)) has been proposed to address large computational costs associated with fine-tuning.

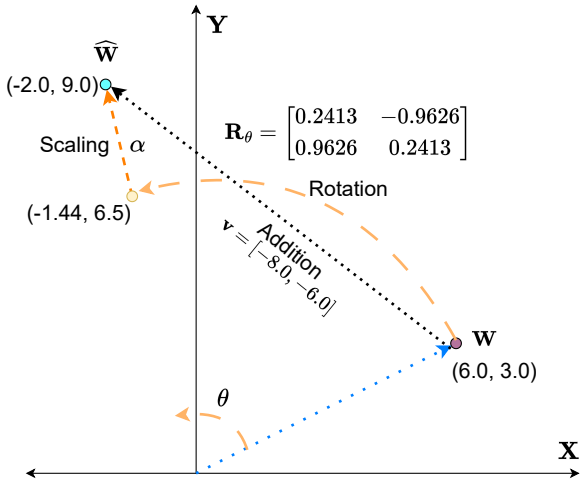

Figure 1: Transformation of a vector $\mathbf{W}$ by two methods: one is via rotation and scaling, the other is via the addition of a vector $\mathbf{v}$.

Various PEFT techniques have been introduced (Han et al., 2024); however, they often introduce trade-offs such as lack of parallelism, increased inference latency (e.g., Adaptors (Houlsby et al., 2019)), or restricted sequence lengths (Petrov et al., 2024). Consequently, re-parametrization-based techniques such as Low-Rank Adaptation (LoRA) (Hu et al., 2022) based fine-tuning methods have gained popularity. Typically, during fine-tuning, the weights (in the form of the weight matrix, e.g., query/key/value matrix) of LLMs are updated using additive update rule, i.e., $\mathbf{W}_0 + \Delta\mathbf{W}$, where $\Delta\mathbf{W}$ is the update in the weights obtained due to fine-tuning. The main idea behind LoRA is to approximate the update matrix $\Delta\mathbf{W} \in \mathbb{R}^{d\times k}$ by a low-rank approximation $\mathbf{BA}$, where $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$ are low-rank matrices ($r \ll d, k$), i.e., $\mathbf{W} = \mathbf{W}_0 + \frac{\alpha}{r} \cdot \mathbf{BA}$, where $\frac{\alpha}{r}$ is a scaling factor. The key intuition is that information required for task-specific updates has a smaller intrinsic rank and lies on a much smaller manifold compared to the entire space of $d\times k$ matrices (Aghajanyan et al.,

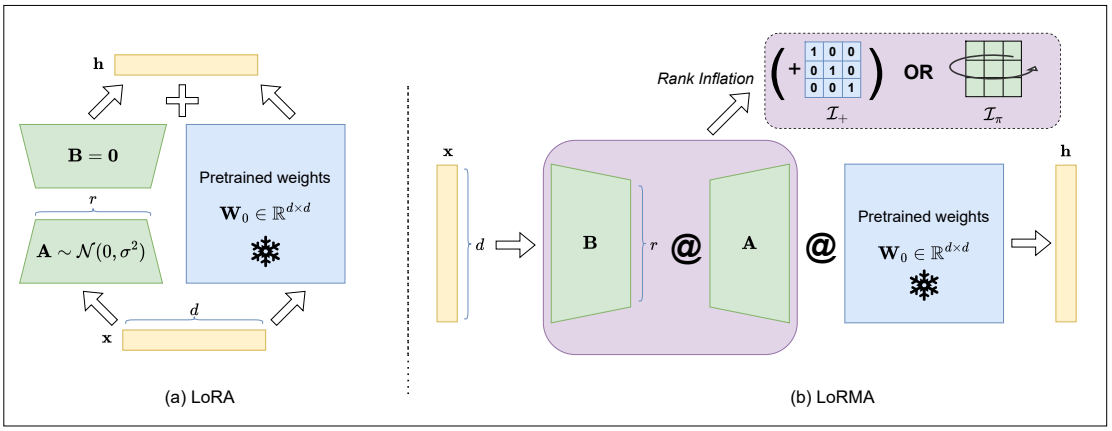

Figure 2: Comparing LoRA (a) and **LoRMA** (b). @ denotes matrix multiplication. $\mathcal{I}_+$ and $\mathcal{I}_\pi$ represent additive and permutation based rank inflation respectively (§3). In case of **LoRMA**, initialization of $\mathbf{A}$ and $\mathbf{B}$ depends on the type of inflation (§3).

2021; Hu et al., 2022). All the current LoRA-based approaches (Yang et al., 2024) have employed additive transformations, where the low-rank update matrix can be added to the original weight matrix during inference. However, similar transformation could also be achieved via multiplicative updates. For example, consider a weight vector $\mathbf{W}$ (Fig. 1) and we would like to transform it to vector $\widehat{\mathbf{W}}$, this could be accomplished via the addition of a vector $\mathbf{v}$, or it could also be done by rotating $\mathbf{W}$ by angle θ (done via Rotation Matrix $\mathbf{R}_\theta$) and subsequently by scaling it by scalar α. Inspired by this intuition, we propose **Low Rank Multiplicative Adaptation** (**LoRMA**) for efficiently fine-tuning LLMs on new tasks. **LoRMA** applies low-rank multiplicative update to a weight matrix, i.e., $\mathbf{W} = \frac{\alpha}{r} \cdot (\mathbf{BA})\mathbf{W}_0$, where $\frac{\alpha}{r}$ is a scalar and $\mathbf{A} \in \mathbb{R}^{d \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times k}$ are low-rank matrices ($r \ll d, k$). **LoRMA**, however, introduces two new challenges: an increase in computational complexity due to additional matrix multiplication operations and a limitation on the maximum rank of the product due to the property: $\mathcal{R}(\mathbf{AB}) \leq \min(\mathcal{R}(\mathbf{A}), \mathcal{R}(\mathbf{B}))$, where $\mathcal{R}(\cdot)$ denotes the rank of a matrix. We employ appropriate ordering of matrix multiplication to address the issue of computational complexity (§3). Additionally, to counteract the issue of *rank inhibition* caused by matrix multiplication, we introduce *rank inflation* strategies and demonstrate their effectiveness (Fig. 2). On average, the proposed techniques have better performance than LoRA (§5). Moreover, it has faster convergence (hence lower training time) than LoRA (§5). In a nutshell, we make the following contributions:

- We propose a new technique for adapting LLMs for downstream tasks: **Low Rank Multiplicative Adaptation** (**LoRMA**). We employ multiplicative updates as an alternative to additive updates used in LoRA. To make the proposed method computationally efficient and overcome rank inhibition brought in by matrix multiplication of low-rank matrices, we propose two variants: **Low Rank Multiplicative Adaptation** with additive inflation (**LoRMA**$_+$) and **Low Rank Multiplicative Adaptation** with permutation-based inflation (**LoRMA**$_\pi$). We propose a very generic framework that can adapted into existing enhancements of LoRA such as Q-LoRA (Dettmers et al., 2023), AutoLoRA (Zhang et al., 2024), DyLoRA (Valipour et al., 2023), and DoRA (Liu et al., 2024)).
- We perform an extensive set of experiments by applying **LoRMA** on transformer-based LLMs on various NLU and NLG tasks and compare it with existing LoRA baselines. On average, the proposed techniques perform better. We show that **LoRMA** shows faster convergence. Further, we perform various ablation studies to see the effect of rank, weight matrix choice, and correlation between updated weight matrices of LoRA and **LoRMA**. We release model code via anonymous repo: `https://anonymous.4open.science/r/LoRMA-7B44`.

2 Related Work

LLMs are generally fine-tuned using Parameter Efficient Fine Tuning (PEFT) methods. Existing PEFT techniques typically fall into three categories (Han et al., 2024): (1) Additive methods (these involve the inclusion of a small set of additional trainable parameters/modules into pretrained LLMs, e.g., Adaptors (Houlsby et al., 2019), Prefix-tuning (Li and Liang, 2021)); (2) Selective

methods (these involve selecting a smaller subset of parameters/modules (e.g. bias in the case of BitFit (Ben Zaken et al., 2022)) and fine-tuning only those (via application of binary masks), e.g., Diff pruning (Guo et al., 2021)); (3) Re-parametrization techniques (these involve re-parameterization of existing weight update matrix via low-rank approximation, e.g., LoRA (Hu et al., 2022)).

Several variants of LoRA have been proposed, each focusing on different aspects of the method (Mao et al., 2025). Here, we describe some of the prominent ones; for more details, please refer to the survey by Yang et al. (2024). DyLoRA (Valipour et al., 2023) dynamically searches for optimal ranks for different weight matrices of the model rather than using a fixed rank across all layers. Methods like AutoLoRA (Zhang et al., 2024) and AdaLoRA (Zhang et al., 2023) adaptively allocate the parameter budget across the model matrices by determining an importance score. ReLoRA (Lialin et al., 2024) introduces aggregated low-rank updates to large neural networks during the training phase with a jagged learning rate scheduler, which depends on the interval in which updates are made to the weight matrix. DoRA (Liu et al., 2024) improves convergence by splitting magnitude and directional updates, enabling weight updates close to traditional fine-tuning. VeRA (Kopiczko et al., 2024) further reduces storage requirements by using fixed matrices $\mathbf{A}$ and $\mathbf{B}$ across layers and introducing trainable diagonal matrices. PRoLoRA (Wang et al., 2024) introduces re-using parameters within the LoRA adapter matrix by replicating chunks across rows and columns. The paper introduces a *rotation enhancement operation* involving chunks in the adapter matrices to recover the expressivity in $\mathbf{BA}$ lost due to replicating parameters and add a set of trainable parameters to further enhance expressivity. Our work is different from PRoLoRA; we introduce operations at the row level to inflate the rank of the matrix. All these methods discussed are additive in nature. We explore the effect of replacing additive modules with *multiplicative transformations*. By investigating multiplicative updates, we aim to address some of the limitations of additive approaches while maintaining the efficiency and effectiveness of PEFT. Multiplicative updates offer a more expressive mechanism for modifying weight matrices. By leveraging matrix multiplication, we can encode richer transformations, which may better capture several complex relationships.

3 Proposed Technique: LoRMA

3.1 Background

Rank of a matrix ($\mathcal{R}(\cdot)$) is defined as the number of independent rows/columns of a matrix and is equivalent to the dimensionality of the space spanned by the rows/columns of the matrix. The rank of a matrix is a fundamental quantity that captures various important characteristics. Some of the key properties (Strang, 2009) are:

$$\mathcal{R}(\mathbf{M}) \leq \min(n, m), \text{ for } \mathbf{M} \in \mathbb{R}^{n \times m} \quad (1)$$
$$\mathcal{R}(\mathbf{M}_1 + \mathbf{M}_2) \geq |\mathcal{R}(\mathbf{M}_1) - \mathcal{R}(\mathbf{M}_2)| \quad (2)$$
$$\mathcal{R}(\mathbf{M}_1 \times \mathbf{M}_2) \leq \min(\mathcal{R}(\mathbf{M}_1), \mathcal{R}(\mathbf{M}_2)) \quad (3)$$
$$\mathcal{R}(\mathbf{M}) = n, \ \mathbf{M} \in \mathbb{R}^{n \times n} \text{ if } \mathbf{M} \text{ is invertible} \quad (4)$$

Property 1 indicates that the rank of a matrix is bounded by its dimensions. Property 2 specifies a lower bound for the rank when matrices undergo addition. Property 3 constrains the rank of the product of two matrices to be bounded by the smaller of both. Property 4 states that square matrices that are invertible (for example, identity matrix $\mathbf{I}_n$) have a rank equal to the number of rows/columns (n).

Existence: In LoRA, weights are updated via additive updates; however, we are proposing a different paradigm where weights are updated via a multiplicative process. One could argue if it is even feasible to attain the same updates via a multiplicative process. In this regard, we first provide proof that it is indeed possible to transform a matrix into another matrix via multiplicative mapping.

Theorem 1. *Given* $\mathbf{M}_0 \in \mathbb{R}^{n \times m}$ *where* $n > m$ *and let* $\mathcal{R}(\mathbf{M}_0) = m$. *For all* $\mathbf{M} \in \mathbb{R}^{n \times m}, \exists\ \mathbf{M}_A \in \mathbb{R}^{n \times n}$, *such that* $\mathbf{M} = \mathbf{M}_A\mathbf{M}_0$.

Proof. Given $\mathbf{M}_0 \in \mathbb{R}^{n \times m}$ where $n > m$ and $\mathcal{R}(\mathbf{M}_0) = m$, implies that $\mathbf{M}_0$ is a full column matrix, i.e., all its columns are independent. This implies that there exists a left inverse of the matrix $\mathbf{M}_0$, say $\mathbf{M}_0^+$, such that $\mathbf{M}_0^+\mathbf{M}_0 = \mathbf{I}_m$. We need to show the existence of a matrix $\mathbf{M}_A$ for any given $\mathbf{M} \in \mathbb{R}^{n \times m}$, such that *pre-multiplication* of $\mathbf{M}_A$ with $\mathbf{M}_0$ gives $\mathbf{M}$, i.e., $\mathbf{M} = \mathbf{M}_A\mathbf{M}_0$. Construct the matrix $\mathbf{M}_A = \mathbf{M}\mathbf{M}_0^+$. This proves the claim as $\mathbf{M}_A\mathbf{M}_0 = (\mathbf{M}\mathbf{M}_0^+)\mathbf{M}_0 = \mathbf{M}\mathbf{I}_m = \mathbf{M}$. □

Corollary 1.1. *Given* $\mathbf{M}_0 \in \mathbb{R}^{n \times m}$ *where* $n > m$ *and* $\mathcal{R}(\mathbf{M}_0) = m$. *There exists* $\mathbf{M} \in \mathbb{R}^{n \times m}$ *such that* $\forall\ \mathbf{M}_A \in \mathbb{R}^{m \times m}, \mathbf{M} \neq \mathbf{M}_0\mathbf{M}_A$.

Proof. Suppose that $\forall\ \mathbf{M} \in \mathbb{R}^{n\times m}, \exists\ \mathbf{M}_A$, such that *post-multiplication*, i.e., $\mathbf{M}_0\mathbf{M}_A = \mathbf{M}$. In other words $\mathbb{R}^{n\times m} = \{\mathbf{M}_0\mathbf{M}_A \,|\, \mathbf{M}_A \in \mathbb{R}^{m\times m}\}$. This does not hold as the *degrees of freedom* on the right-hand side for a given full column matrix $\mathbf{M}_0$ is m^2 (number of elements in $\mathbf{M}_A$), while the potential *degrees of freedom* required is nm-many in $\mathbb{R}^{n\times m}$. Formally, consider a counter-example. Assume the given $\mathbf{M}_0 = \begin{pmatrix}\mathbf{I}_m \\ \mathbf{0}_{n-m}\end{pmatrix}$. Let the required transformation be to $\mathbf{M} = \begin{pmatrix}\mathbf{0}_{n-m} \\ \mathbf{I}_m\end{pmatrix}$, where $\mathbf{0}_{n-m}$ denotes a zero matrix $\in \mathbb{R}^{(n-m)\times m}$. It is easy to verify that $\nexists\ \mathbf{M}_A \in \mathbb{R}^{m\times m}$ which satisfies the desired transformation. □

Corollary 1.2. *Given the square matrix* $\mathbf{M} \in \mathbb{R}^{n\times n}$ *and non-singular matrix* $\mathbf{M}_0 \in \mathbb{R}^{n\times n}$, *there exist matrices* $\mathbf{M}_{A_\ell}, \mathbf{M}_{A_r} \in \mathbb{R}^{n\times n}$ *that can transform* $\mathbf{M}_0$ *into* $\mathbf{M}$ *via pre-multiplication/post-multiplication respectively, i.e.,* $\mathbf{M} = \mathbf{M}_{A_\ell}\mathbf{M}_0$ *and* $\mathbf{M} = \mathbf{M}\mathbf{M}_{A_r}$.

Remark. We present the above results to motivate the existence of a multiplicative transformation that maps frozen pre-trained weight matrices $\mathbf{M}_0$ to potentially any other set of weights with the same dimensionality. A key requirement underlying this hypothesis is that the weight matrices—such as `attention.self.query` in RoBERTa or the spliced `c_attn` in GPT-2 (the models used in §4)—are invertible. To ensure this, we verify that these matrices are either full rank or close to full rank, typically within 99% of the maximum possible rank.

3.2 LoRMA

LoRA (Hu et al., 2022) updates a pre-trained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ by additive update, i.e., $\mathbf{h} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x}$, where $\mathbf{x}$ is the input and $\mathbf{W}_0$ is frozen during fine-tuning. The updates $\Delta\mathbf{W}$ are constrained to a low-rank decomposition $\mathbf{BA}$ where $\mathbf{B} \in \mathbb{R}^{d\times r}, \mathbf{A} \in \mathbb{R}^{r\times k}$ and $r \ll \min(d,k)$, i.e.

$$\mathbf{h} = (\mathbf{W}_0 + \underbrace{\frac{\alpha}{r}\cdot\mathbf{BA}}_{\Delta\mathbf{W}})\,\mathbf{x}$$

where α is a scalar. To ensure that the initial training pass resembles the pre-trained model, $\mathbf{B}$ is initialized to $\mathbf{0}$.

Theorem 1 guarantees the existence of a matrix $\mathbf{M}_A$ for a desired transformation. Hence, we propose a multiplicative update rule, i.e., $\mathbf{M}_A \times \mathbf{W}_0$.

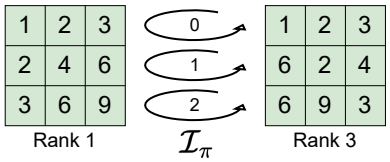

Figure 3: Permutation-Based Inflation $\mathcal{I}_\pi$ operation. Rearrange matrix entries to inflate the rank.

The update is approximated using low-rank approximation, i.e.,

$$\mathbf{h} = ((\mathbf{BA}) \times \mathbf{W}_0)\mathbf{x}$$

where, $\mathbf{B} \in \mathbb{R}^{d\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times d}$ with $r \ll \min(d,k)$ are low-rank matrices such that the product $\mathbf{BA}$ captures the desired transformation of matrix $\mathbf{W}_0$. However, this naive approach has a few shortcomings. In accordance with property 3, the resultant matrix product is limited to be of rank r since $\mathcal{R}(\mathbf{BAW}_0) \leq \mathcal{R}(\mathbf{B}) \leq r$. This significantly undermines the potential desirable independence of rows/columns in the final representation of the updated weights. Further, during the onset of the fine-tuning, in the case of LoRA, it is preferable to have $\Delta\mathbf{W} = \mathbf{0}$, so that $\mathbf{h} = \mathbf{W}\mathbf{x}$, this ensures stability during fine-tuning (Hu et al., 2022). This is achieved by initializing $\mathbf{B}$ with zeros, ensuring that the additive update starts at zero. In our case, this would require the matrix $\mathbf{BA}$ to be equal to the identity matrix $\mathbf{I}_d$. However, the property 3 dictates that this cannot be the case as $\mathcal{R}(\mathbf{I}_d) = d$. This forces the tuning to have a significant deviation from the beginning. We propose two strategies to mitigate the rank limitation imposed by low-rank matrices to capture the multiplicative transformation.

Permutation-Based Inflation ($\mathcal{I}_\pi$). Permutation-based rank inflation utilizes the idea of strategic re-arrangement of elements of the matrices to increase the rank of a matrix. The rows of the matrix are rotated cyclically in incremental steps. The i th row is rotated by i, i.e. (row 0 by 0, row 1 by 1 ...). As can be seen in Fig. 3, this effective rearranging of a matrix's elements has enhanced the matrix's rank from 1 to a full rank of 3. We introduce this operation on the product of the matrices $\mathbf{BA}$, which equips the model with the ability to learn a higher-rank representation. Since the operation is simply a re-arrangement of the parameters, it does not make the gradient in-tractable.

$$\mathbf{h} = (\mathcal{I}_\pi(\mathbf{BA}) \times \mathbf{W}_0)\mathbf{x}$$

This inflation strategy also provides a better initialization scheme. This is achieved by warranting $\mathcal{I}_\pi(\mathbf{BA}) = \mathbf{I}_d$. The first column of $\mathbf{B}$ is set to

Method	Computation	Complexity
LoRA	$(\mathbf{W}_0 + \mathbf{BA})\mathbf{x}$	$\mathcal{O}(dkb)$
LoRMA	$\mathbf{BAW}_0\mathbf{x}$	$\mathcal{O}(dkb)$
LoRMA$_\pi$	$\mathcal{I}_\pi(\mathbf{BA})\mathbf{W}_0\mathbf{x}$	$\mathcal{O}(d^2(r+b))$
LoRMA$_+$	$\mathbf{W}_0\mathbf{x} + \mathbf{BAW}_0\mathbf{x}$	$\mathcal{O}(dkb)$

Table 1: Time Complexity for computations incurred by different methods during training time.

ones, while the rest of the elements are randomly initialized. $\mathbf{A}[0,0]$ is set to one, while the rest of the elements in $\mathbf{A}$ are set to zero. We refer to this variant as **LoRMA**$_\pi$.

Additive Rank Inflation ($\mathcal{I}_+$). Motivated by the need for an identity initialization of the transformation matrix, we introduce another technique to address the rank limitation inherent in low-rank approximations. Drawing inspiration from ridge regression, where the solution is stabilized by adding a regularization term $\left(\hat{\theta} = (\mathbf{X}^T\mathbf{X} + \lambda \cdot \mathbf{I})^{-1}\mathbf{X}^\mathbf{T}\mathbf{Y}\right)$, we incorporate an identity matrix into our formulation through addition. Specifically, the resulting transformation takes the form:

$$\mathbf{h} = \mathcal{I}_+(\mathbf{BA})\mathbf{W}_0\mathbf{x} = \left(\frac{\alpha}{r} \cdot \mathbf{BA} + \mathbf{I}_d\right)\mathbf{W}_0\mathbf{x}$$

The rank of the sum $\left(\frac{\alpha}{r} \cdot \mathbf{BA} + \mathbf{I}_d\right)$ (here α is the scaling factor) is guaranteed to be at least $d - r$, as dictated by property 2. Since $r \ll d,\ d - r \approx d$, this preserves sufficient rank flexibility, enabling richer transformations during training. This approach ensures that the transformation begins with identity initialization at the start of the fine-tuning process by setting $\mathbf{B} = \mathbf{0}$ and randomly initializing $\mathbf{A}$. We refer to this variant as **LoRMA**$_+$.

To summarize, formally, the update rule for **LoRMA** is given by:

$$\mathbf{h} = (\mathcal{I}(\mathbf{BA}) \times \mathbf{W}_0)\mathbf{x}$$

where, $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$, $\mathbf{B} \in \mathbb{R}^{d\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times d}$ and $r \ll \min(d, k)$ and $\mathcal{I}$ denotes rank inflation techniques employed ($\mathcal{I}_\pi/\mathcal{I}_+$). $\mathbf{A}$ and $\mathbf{B}$ are initialized such that $\mathcal{I}(\mathbf{BA}) = \mathbf{I}_d$. In our case, the application of the **LoRMA** over RoBERTa and GPT-2 (§4) is over square matrices and Corollary 1.2 ensures the existence of a multiplicand which is being adapted.

Computational Complexity: An obvious consideration to take is the computational cost incurred by the multiplicative transformations that are being introduced. Table 1 (also see App. §A), provides a comparative analysis of the computational costs of LoRA for $\mathbf{x} \in \mathbb{R}^{k\times b}$ where b denotes the batch size. Utilizing associativity of matrix multiplications and first performing multiplication with $\mathbf{x}$ helps make the cost comparable to LoRA. The cost of **LoRMA**$_\pi$ is relatively higher since there is the requirement to first compute $\mathbf{BA}$ since the $\mathcal{I}_\pi$ operation is being applied on the product.

LoRMA Advantages: Similar to LoRA, **LoRMA** helps to avoid inference-time latency by permitting the merging of updates into the frozen weights, i.e., $\mathbf{W}_{\text{fine-tuned}} = \mathcal{I}(\mathbf{BA}) \times \mathbf{W}_0$. In the multiplicative representation, on updating a single parameter, the resultant weight matrix has many more updates as compared to additive transformations, as can be seen in App. Fig. 7. This can lead to the requirement of fewer updates to modify the weight matrix to another matrix. We observe this empirically in our experiments (§5).

4 Experiments

We conduct a comprehensive set of experiments to evaluate the effectiveness of our proposed techniques on two widely-used language models: RoBERTa ('base' and 'large')(Liu et al., 2019) and GPT-2 (medium) (Radford et al., 2019). The choice of models and tasks is motivated by the need for a fair comparison with the original LoRA. Our evaluation spans a variety of downstream tasks in natural language understanding (NLU) and natural language generation (NLG). For RoBERTa, we assess the performance of our approach on the GLUE benchmark (Wang et al., 2018). The GLUE benchmark provides a comprehensive set of tasks ranging from single-sentence tasks (CoLA and SST-2) to similarity and paraphrasing tasks (MRPC, STS-B, QQP) to natural language inference tasks (MNLI, QNLI, RTE). For GPT-2, we present results on the E2E dataset (Novikova et al., 2017), commonly used for evaluating NLG capabilities. Additional NLG experiments, including DART (Nan et al., 2021) and WebNLG (Gardent et al., 2017), to evaluate our approach have been discussed in App. §C. Details of tasks, dataset statistics, and task-specific hyperparameters can be found in App. §B and App. §D, respectively. We compare **LoRMA**$_\pi$ and **LoRMA**$_+$ with Full Finetuning (FT), BitFit (Ben Zaken et al., 2022), LoRA (Hu et al., 2022), and AutoLoRA (Zhang et al., 2024). Comparison with AutoLoRA is motivated by its superior performance over LoRA and its variants. We adhere to the standard experimental setup used in LoRA to ensure consistency with prior work. Specifically, our multiplicative transformation technique is applied to the query (W_q) and value (W_v) matrices within the attention mechanism of the models.

Method	# Params	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoBERTa$_{\text{base}}$ (FT)*	125.0M	87.6	94.8	90.2	63.6	92.8	91.9	78.7	91.2	86.4
RoBERTa$_{\text{base}}$ (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	**81.5**	90.8	85.2
RoBERTa$_{\text{base}}$ (AutoLoRA)*	0.3M	87.0	**94.9**	89.4	61.3	**92.9**	90.3	77.0	90.8	85.5
RoBERTa$_{\text{base}}$ (LoRA)	0.3M	**87.5**	94.6	91.0	63.6	92.7	**90.8**	78.0	89.5	85.9
RoBERTa$_{\text{base}}$ (**LoRMA**$_{\pi}$)	0.3M	87.4	94.2	91.1	63.5	92.1	90.5	75.4	90.6	85.6
RoBERTa$_{\text{base}}$ (**LoRMA**$_{+}$)	0.3M	**87.5**	94.7	91.3	**64.2**	92.6	90.6	76.5	**90.9**	**86.0**
RoBERTa$_{\text{large}}$ (FT)*	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
RoBERTa$_{\text{large}}$ (Adapter$^{\text{H}}$)*	0.8M	90.3	**96.3**	87.7	66.3	94.7	91.5	72.9	91.5	86.4
RoBERTa$_{\text{large}}$ (LoRA)	0.8M	**90.7**	96.2	**93.0**	**68.1**	94.6	**91.6**	85.2	92.0	88.9
RoBERTa$_{\text{large}}$ (**LoRMA**$_{\pi}$)	0.8M	89.3	95.2	92.3	66.8	93.5	90.0	84.5	91.9	88.0
RoBERTa$_{\text{large}}$ (**LoRMA**$_{+}$)	0.8M	**90.7**	95.9	**93.0**	67.8	**94.9**	91.3	**86.6**	**92.2**	**89.0**

Table 2: Performance on GLUE tasks. The metrics are Matthews correlation for CoLA, Pearson coefficient for STS-B, F1 for MRPC, and accuracy for other tasks. $*$ denotes metrics published in prior works. The values present are averaged over 3 runs on different seeds. Full tuning (FT) statistics are also reported for comparison purposes.

Method	# Params	E2E				
		BLEU	NIST	MET	ROUGE-L	CIDEr
Inference: Beam size 10						
GPT-2$_{\text{medium}}$ (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2$_{\text{medium}}$ (Adapter$^{\text{H}}$)*	11.09M	67.3	8.50	46.0	70.7	2.44
GPT-2$_{\text{medium}}$ (AutoLoRA)*	0.3M	67.9	8.68	46.0	68.9	2.37
GPT-2$_{\text{medium}}$ (LoRA)	0.3M	69.1	8.73	**46.5**	**71.4**	**2.51**
GPT-2$_{\text{medium}}$ (**LoRMA**$_{\pi}$)	0.3M	69.0	8.72	46.4	70.8	2.42
GPT-2$_{\text{medium}}$ (**LoRMA**$_{+}$)	0.3M	**69.3**	**8.75**	46.3	70.8	**2.51**

Table 3: Performance on NLG. $*$ denotes metrics published in prior works. Full tuning (FT) statistics are also reported for comparison purposes.

5 Results and Analysis

Results: Table 2 summarizes the results of NLU tasks performed on RoBERTa$_{\text{base}}$ and RoBERTa$_{\text{large}}$, whereas Table 3 presents the results of NLG tasks. Both multiplicative adaptation methods, **LoRMA**$_{+}$and **LoRMA**$_{\pi}$, show superior and/or comparable performances over a wide array of NLU and NLG tasks. Though on average, **LoRMA**$_{+}$ has slightly better performance. We perform various ablation experiments on the proposed model as described next.

Presence v/s Absence of Rank-Inflation: Earlier we explained (§3.1) why a naive low-rank multiplicative adaptation of $\mathbf{W}_0$ has limitations. We present here the empirical verification of the same, and the results are shown in Table 4. The experiments were done on RoBERTa$_{\text{large}}$ on a subset of GLUE tasks, and all the hyperparameters and training conditions were kept exactly the same, apart from the presence and absence of the rank inflation strategies. The results for the $\mathcal{I}_{+}$ have been reproduced for comparison. Further, we evaluate the effectiveness of the proposed rank inflation strategies by monitoring the rank of matrices throughout the training procedure. We observe that these operations successfully help break the rank bottleneck, and the matrices are almost full rank throughout (refer to App. §E.1).

Method	MRPC	STS-B	RTE
LoRMA	81.2	15.6	52.7
LoRMA$_{+}$	**92.9**	**92.2**	**86.6**

Table 4: The absence of rank inflation severely limits the model's capabilities.

Method	CoLA	MRPC	STS-B	RTE
LoRMA$_{+}$(*Post*)	**68.9**	92.5	91.8	86.3
LoRMA$_{+}$(*Pre*)	67.8	**92.9**	**92.2**	**86.6**

Table 5: Comparison of Pre-multiplication vs Post-multiplication.

Pre-multiplication v/s Post multiplication: The Corollary 1.2 allows for an equivalent representation of the multiplicative transformation, i.e., post and pre-multiplication. We test *post-multiplicative* **LoRMA**$_{+}$ (Table 5) and observe almost comparable performance with the strategy mentioned above.

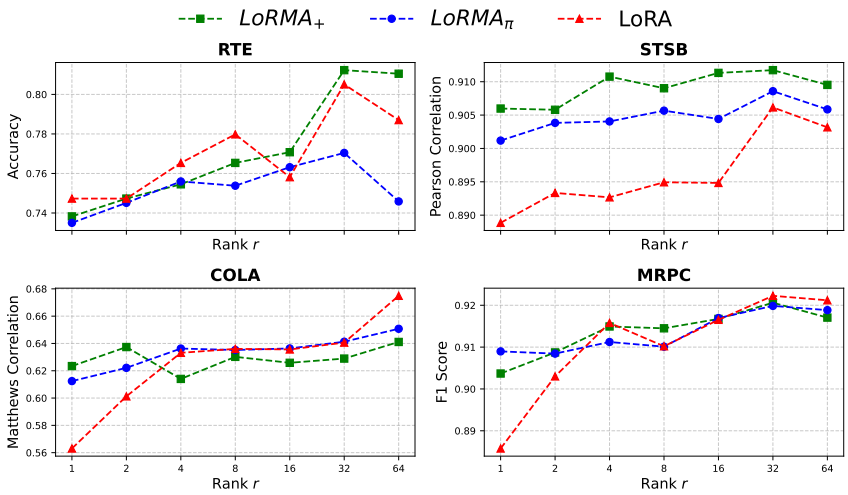

Figure 4: Comparing performance over different ranks for the GLUE tasks RTE, STSB, CoLA, MRPC for adaptation of RoBERTa$_{\text{base}}$.

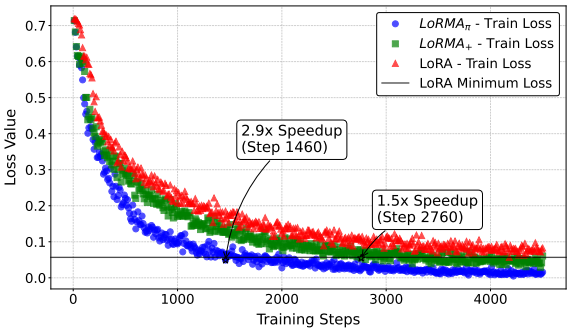

Figure 5: Train loss curves for CoLA: RoBERTa$_{\text{base}}$ for various techniques.

This verifies the discussion in §3.1.

Task	% AUC ↓ (I_+)	% AUC ↓ (I_π)
SST-2 (RoBERTa$_{\text{base}}$)	10.84	30.21
CoLA (RoBERTa$_{\text{base}}$)	23.20	51.97

Table 6: % AUC decrease in comparison with LoRA

Choice of Weight Matrix: With a fixed parameter budget, it becomes crucial to strategically allocate adaptive weights to achieve optimal performance. To investigate this, we set a parameter budget of approximately 150K parameters, corresponding to $r = 8$ for single-weight-type adaptation across the GLUE tasks MRPC and STS-B. The adapted model is RoBERTa$_{\text{base}}$, with a scaling factor $\alpha = r$ (for varying ranks r) used in the additive variant of our method(**LoRMA**$_+$). The trends observed in Table 7 suggest that, given a fixed budget, diversifying the adaptive tuning—i.e., distributing the adaptation across multiple weight matrices—leads to better performance.

Rank r v/s Performance: To see the effect of rank r over performance, we adapt $\mathbf{W}_q$, $\{\mathbf{W}_q, \mathbf{W}_k\}$ and $\{\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o\}$ weight matrices with **LoRMA**$_+$ and the results are compiled in Table 8. The general trend indicates that performance improves as the rank (and consequently the number of parameters) increases. This observation aligns with similar experiments conducted on **LoRMA**$_+$, **LoRMA**$_\pi$, and LoRA (Fig. 4). Once again, the overarching trend shows that performance improves with higher ranks across all techniques. However, this trend is neither strict nor monotonic, as performance dips at higher ranks are also observed. This could possibly be due to a low intrinsic rank of $\Delta\mathbf{W}$ being sufficient to capture the transformation and higher ranks leading to over-parametrization rather than learning additional information. Notably, **LoRMA** scales effectively across different ranks and demonstrates comparable or even superior performance to LoRA, particularly in highly parameter-constrained scenarios. This underscores the scalability and effectiveness of **LoRMA**, along with its rank-inflation variants, in resource-constrained settings.

Faster convergence of LoRMA: Convergence time reflects how quickly a model reaches a stable or desirable level of performance during training. To complement the competitive evaluation metrics presented in Table 2, we demonstrate in this section that our proposed techniques achieve faster convergence compared to LoRA. We quantify convergence speed using the *Area Under the Curve (AUC)* metric for the training loss curve, where a lower AUC indicates faster convergence. Fig. 5 illustrates the training loss curves for **LoRMA** (both $\mathcal{I}_+$ and $\mathcal{I}_\pi$ variants) compared to LoRA on the CoLA task while using RoBERTa$_{\text{base}}$ model. The results show a steeper decline in training loss. The percentage reduction in AUC for various tasks relative

	# Trainable Parameters ≈ 150K					
Weight Matrix	$\mathbf{W}_q$	$\mathbf{W}_k$	$\mathbf{W}_v$	$\mathbf{W}_q, \mathbf{W}_v$	$\mathbf{W}_q, \mathbf{W}_k$	$\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$
r	8	8	8	4	4	2
MRPC	90.2	91.0	91.4	90.2	91.3	**91.6**
STS-B	89.0	89.3	90.9	90.5	89.2	**91.2**

Table 7: RoBERTa$_{\text{base}}$ with **LoRMA**$_+$ on a fixed budget for the GLUE tasks MRPC and STS-B, with scaling factor $\alpha = r$ for respective r's depending upon the application.

	Weight Matrix	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
MRPC	$\mathbf{W}_q$	89.6	90.5	90.2	90.2	91.2
	$\mathbf{W}_q, \mathbf{W}_k$	90.6	91.4	91.3	91.4	91.8
	$\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$	90.7	91.6	91.7	91.7	93.2
STS-B	$\mathbf{W}_q$	88.4	88.6	88.6	89.0	89.3
	$\mathbf{W}_q, \mathbf{W}_k$	89.1	89.5	89.2	89.3	89.2
	$\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$	91.0	91.2	90.9	90.9	91.1

Table 8: RoBERTa$_{\text{base}}$ with **LoRMA**$_+$. Validation accuracy across different weights being adapted with varying ranks r for the GLUE tasks MRPC and STS-B.

	Layer 3			Layer 23		
Metric	$\Delta\mathbf{W}_{\text{LoRMA}_+}$	$\Delta\mathbf{W}_{\text{LoRMA}_\pi}$	Random	$\Delta\mathbf{W}_{\text{LoRMA}_+}$	$\Delta\mathbf{W}_{\text{LoRMA}_\pi}$	Random
$\|W - \Delta\mathbf{W}_{\text{LoRA}}\|_F$	3.54	31.93	1024.27	10.02	38.31	1022.40
$\texttt{cos-sim}(W, \Delta\mathbf{W}_{\text{LoRA}})$	74.2×10^{-3}	4.1×10^{-3}	$0.1 \times 10{-}3$	68.1×10^{-3}	0.3×10^{-3}	-1.2×10^{-3}
$(W, \Delta\mathbf{W}_{\text{LoRA}})^r_{\mathcal{S}}$	0.99	8.93	176.18	3.75	13.40	175.67
$(W, \Delta\mathbf{W}_{\text{LoRA}})^r_{\mathcal{E}}$	0.06	2.67	89.07	0.49	3.31	89.70
$\Theta_1(W, \Delta\mathbf{W}_{\text{LoRA}})$	2.28×10^{-6}	2.27×10^{-6}	1.56	2.34×10^{-6}	2.32×10^{-6}	1.57

Table 9: Correlation between the learned representations of $\Delta\mathbf{W}_{\text{LoRA}}$ and $\Delta\mathbf{W}_{\text{LoRMA}}$ for RoBERTa$_{\text{large}}$ model.

to LoRA is summarized in Table 6. Similar trends were observed for other tasks as well.

Comparision with $\Delta\mathbf{W}_{\text{LoRA}}$: Let us define $\Delta\mathbf{W}$ for any technique to be the difference between the final adapted weight matrix and the initial weight matrix (the frozen weights). We investigate the relationship between $\Delta\mathbf{W}_{\text{LoRA}}$, as learned by LoRA, and $\Delta\mathbf{W}_{\text{LoRMA}_+}$ as well as $\Delta\mathbf{W}_{\text{LoRMA}_\pi}$, which is obtained through our multiplicative adaptation method. To assess the correlation between any two two matrices, we employ a variety of metrics, the results of which are summarized in Table 9. We compute the Frobenius norm ($\|\cdot\|_F$) between the two matrices, where a *larger* value indicates a bigger deviation between the two matrices. We also evaluate the cosine similarity of the flattened matrices ($\texttt{cos-sim}(\cdot,\cdot)$) to measure their alignment. We compute the sum of squared differences between the top-r singular values $(\cdot,\cdot)^r_{\mathcal{S}}$ and eigenvalues $(\cdot,\cdot)^r_{\mathcal{E}}$ of the two matrices to assess their similarity. Finally, we measure the principal subspace angle $\Theta_1(\cdot,\cdot)$ between their column spaces to quantify their geometric alignment, where smaller angles indicate a higher degree of overlap between the subspaces. The overarching trend of these metrics implies a high correlation between $\Delta\mathbf{W}_{\text{LoRA}}$ and $\Delta\mathbf{W}_{\text{LoRMA}_+}$ and $\Delta\mathbf{W}_{\text{LoRMA}_\pi}$, which shows that out multiplicative techniques are learning similar representations to that learnt by LoRA. To assess the expressibility of the transformations, we compare the rank of $\Delta\mathbf{W}$. For LoRA, $\Delta\mathbf{W} = \mathbf{BA}$; hence, it is restricted to be a low-rank update (property 3). While for **LoRMA**$_\pi$, there are no such limitations. We empirically observe them to be almost full-rank matrices (refer to App. §E.2).

6 Conclusion

In this work, we proposed **LoRMA**, a completely different and new way of updating weights of a language model via multiplicative updates. We mathematically proved the existence of multiplicative updates. Further, to overcome the limitations of the naive approach of multiplicative updates, we propose two methods to inflate the rank of the update matrix via permutation and additive identity. Extensive experiments prove the good performance and training efficacy of the proposed approach. In the future, we plan to experiment with combining **LoRMA** with some of the existing LoRA-based enhancements like DoRA and DyLoRA.

Limitations

The ability to plug out the parameters. In a production setting, LoRA converts a base model to the model tuned to a task by adding BA to the weight matrix of the model, and one can recover the original model by subtracting out the original weight matrix. In the case of **LoRMA**, by updating the original weight matrix by multiplication with $\mathcal{I}(BA)$, the tuned model can be deployed. The recovery of original model weights from the updated form would require $\mathcal{I}(BA)$ to be invertible, which might not be the case, as discussed above. To mitigate this, a copy of the original parameters would have to be maintained.

Time complexity of $\mathcal{I}_\pi$ during training. As discussed in Section 3, while other variants of **LoRMA** have a similar order of time complexity as LoRA during the training process, **LoRMA**$_\pi$ has a slightly higher time complexity at training time. However, by merging weights during inference time, all of them would have no inference latency, which makes the method still a viable option.

Experiments with Smaller Models. In this paper, in order to be comparable with previous works, we experimented mainly with RoBERTa and GPT-2 models. We performed experiments on a variety of tasks, and the results are indicative of the efficacy of the proposed method. We expect the results to generalize to larger sized LLMs as well.

Ethical Considerations

We abide by the ACL Code of Ethics code during our research. This work introduces a new variant of parameter-efficient fine-tuning approaches for LLMs that do not directly have possible harms associated with them. The use of LLMs has ethical considerations that should be kept in mind. We have used public models (RoBERTa and GPT2) and public datasets (GLUE, E2E, WebNLG and DART) to evaluate the effectiveness of our proposed approach.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

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 and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2024. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2025. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7):197605.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Aleksandar Petrov, Philip Torr, and Adel Bibi. 2024. When do prompting and prefix-tuning work? a theory of capabilities and limitations. In *The Twelfth International Conference on Learning Representations*.

Adam Poliak. 2020. A survey on recognizing textual entailment as an NLP evaluation. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 92–109, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural language understanding with the quora question pairs dataset.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Gilbert Strang. 2009. *Introduction to Linear Algebra*, fourth edition. Wellesley-Cambridge Press, Wellesley, MA.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Sheng Wang, Boyang Xue, Jiacheng Ye, Jiyue Jiang, Liheng Chen, Lingpeng Kong, and Chuan Wu. 2024. PRoLoRA: Partial rotation empowers more parameter-efficient LoRA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2829–2841, Bangkok, Thailand. Association for Computational Linguistics.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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

Menglin Yang, Jialin Chen, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Qianru Zhang, Min Zhou, Irwin King, et al. 2024. Low-rank adaptation for foundation models: A comprehensive review. *arXiv preprint arXiv:2501.00365*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. AutoLoRA: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5048–5060, Mexico City, Mexico. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Appendix

Table of Contents

List of Tables

List of Figures

A Time complexity calculations

Here, we describe the strategic re-ordering of operations to mitigate the large time complexity incurred due to matrix multiplications. These have been summarized in Table 10.

LoRA

Multiply $\mathbf{W}_0$ with $\mathbf{x}$ ($\mathcal{O}(dkb)$). Multiply $\mathbf{A}$ with $\mathbf{x}$ ($\mathcal{O}(krb)$). Multiply $\mathbf{B}$ with $\mathbf{Ax}$ ($\mathcal{O}(drb)$). Add $\mathbf{W}_0\mathbf{x}$ with $\mathbf{BAx}$ ($\mathcal{O}(db)$).

LoRMA

Multiply $\mathbf{W}_0$ with $\mathbf{x}$ ($\mathcal{O}(dkb)$). Multiply $\mathbf{A}$ with $\mathbf{W}_0\mathbf{x}$ ($\mathcal{O}(drb)$). Multiply $\mathbf{B}$ with $\mathbf{AW}_0\mathbf{x}$ ($\mathcal{O}(drb)$).

LoRMA$_\pi$

Multiply $\mathbf{W}_0$ with $\mathbf{x}$ ($\mathcal{O}(dkb)$). Multiply $\mathbf{B}$ with $\mathbf{A}$. Inflation $\mathcal{I}_\pi$ of BA ($\mathcal{O}(d^2r)$). Multiply $\mathcal{I}_\pi(BA)$) with $\mathbf{W}_0\mathbf{x}$ ($\mathcal{O}(d^2b)$).

LoRMA$_+$

Multiply $\mathbf{W}_0$ with $\mathbf{x}$ ($\mathcal{O}(dkb)$) for first term. Multiply $\mathbf{W}_0$ with $\mathbf{x}$ ($\mathcal{O}(dkb)$) for second term. Multiply $\mathbf{A}$ with $\mathbf{W}_0\mathbf{x}$ ($\mathcal{O}(drb)$). Multiply $\mathbf{B}$ with $\mathbf{AW}_0\mathbf{x}$ ($\mathcal{O}(drb)$). Add $\mathbf{W}_0\mathbf{x}$ with $\mathbf{BAW}_0\mathbf{x}$ ($\mathcal{O}(db)$).

Method	Computation	Complexity Calculation
LoRA	$(\mathbf{W}_0 + \mathbf{BA})\mathbf{x}$	$dkb + krb + drb + db$ $\mathcal{O}(dkb)$
LoRMA	$\mathbf{BAW}_0\mathbf{x}$	$dkb + 2drb$ $\mathcal{O}(dkb)$
LoRMA$_\pi$	$\mathcal{I}_\pi(\mathbf{BA})\mathbf{W}_0\mathbf{x}$	$dkb + d^2r + d^2b$ $\mathcal{O}(d^2(r+b))$
LoRMA$_+$	$\mathbf{W}_0\mathbf{x} + \mathbf{BAW}_0\mathbf{x}$	$2dkb + 2drb + db$ $\mathcal{O}(dkb)$

Table 10: Time Complexity for computations incurred by different methods during training time.

B Dataset description

- **GLUE Benchmark**: The benchmark comprises wide-ranging natural language understanding tasks mostly restricted to English language. It consists of tasks like CoLA ((Warstadt et al., 2019), grammatical acceptability), SST-2 ((Socher et al., 2013), sentiment analysis), MRPC ((Dolan and Brockett, 2005), semantic textual similarity), STS-B ((Cer et al., 2017), semantic textual similarity), QQP ((Sharma et al., 2019), question answers), and inference tasks like MNLI

(Williams et al., 2018), QNLI (Rajpurkar et al., 2018) as well as RTE (Poliak, 2020).

The datasets are available under public license and were used using the `datasets` API provided by HuggingFace. The dataset statistics are presented in Table 11.

- **E2E NLG Challenge**: The E2E dataset was released in Novikova et al. (2017) and is a popular dataset for testing efficacy in natural language generation tasks. The dataset consists of 42061 training samples, 4672 dev, and 4693 test samples. Success on this task is typically measured by achieving a high BLEU, NIST, METEOR, Rouge-L, and CIDEr score, as presented in the paper.

- **DART**: This is a large dataset for open-domain record-to-text generation published in (Nan et al., 2021). It has a total of close to 82K samples. The underlying task is *rdf-to-text* (which is mapping entity relations to text).

- **WebNLG Challenge**: WebNLG challenge (Gardent et al., 2017) is yet another dataset that consists of mapping data to text. Data is a set of triples, and text is the verbalization of this data. It has close to 22K total samples. It involves examples from 9 distinct DBPedia categories during training, with the complete dataset having 15 categories.

C Additional Experiments

We conduct more experiments to test and benchmark the capabilities of our multiplicative adapters. In this series, we repeat our NLG experiments for DART (Nan et al., 2021) and WebNLG challenge (Gardent et al., 2017). The trends are similar to that observed in Table 3. Our adapters perform comparably in evaluation with LoRA. The results are presented in Table 12.

D Hyperparameters and Training Setup

We adhere to the standard experimental setup used in LoRA to ensure consistency with prior work. Specifically, our multiplicative transformation technique is applied to the query (W_q) and value (W_v) matrices within the attention mechanism of the models. This means that for a 12-layer RoBERTa$_{\text{base}}$ or GPT-2 M model, our multiplicative adapters are applied a total of 24 times—once for each query and value matrix across all layers. Similarly, for the 24-layer RoBERTa$_{\text{large}}$ model, the multiplicative adapter is applied 48 times. We use the pre-trained versions of RoBERTa$_{\text{base}}$ (125M parameters) and RoBERTa$_{\text{large}}$ (355M parameters) available in the HuggingFace Transformers library (Wolf et al., 2020). We employed the PEFT (Mangrulkar et al., 2022) support on the HuggingFace where available for running experiments. For NLG experiments based on GPT-2, we draw inspiration from LoRA's published code. The pre-trained GPT-2 models have been made available by HuggingFace.

D.1 RoBERTa

We utilize AdamW optimizer (Loshchilov and Hutter, 2019) along with a linear learning rate decay schedule. The results reported are the mean of runs for 3 random seeds, with the result for a single run taken to be from the best epoch. The pre-trained RoBERTa model is taken and fine-tuned for each task separately. The hyperparameters have been presented in Table 13. For the results of previous works, refer to Ben Zaken et al. (2022); Houlsby et al. (2019); Zhang et al. (2024).

D.2 GPT-2

The GPT-2 models have been trained via the AdamW optimizer using a linear learning rate schedule for 5 epochs. The hyper-parameters used for the experiments are presented in Table 14. For the results of previous works, refer to Zhang et al. (2024); Hu et al. (2022).

E Ablation

E.1 Rank Progression with training

As discussed in §3, the proposed initialization schemes, along with rank inflation, help to begin the training process with $\mathcal{I}(\mathbf{BA}) = \mathbf{I}_d$ which is a full rank matrix. To empirically verify whether the rank inflation techniques, beginning with a high rank during initialization, also retain it across the training process, we monitor the rank $\mathcal{I}(\mathbf{BA})$ where $d = 1024, \mathbf{B} \in \mathbb{R}^{1024\times 8}, \mathbf{A} \in \mathbb{R}^{8\times 1024}$. For both $\mathcal{I}_+$ and $\mathcal{I}_\pi$, throughout the fine-tuning process, it is observed that the inflated matrix product is always almost full rank (1024). Thus helping preserve the representational capacity of the matrix product.

	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
Train	8551	67349	3668	5749	363871	392702	104743	2490
Validation	1043	872	408	1500	40432	9815	5463	277

Table 11: GLUE Benchmark statistics

Method	# Params (M)	E2E BLEU	E2E NIST	E2E MET	E2E ROUGE-L	E2E CIDEr	DART BLEU	DART METEOR	DART TER	WebNLG BLEU	WebNLG METEOR	WebNLG TER
		Inference: Beam size 15					Inference: Beam size 10			Inference: Beam size 10		
GPT-2_{medium} (LoRA)	0.3M	67.5	8.53	**46.2**	**70.8**	2.49	**45.35**	**0.38**	**0.53**	**52.27**	**0.40**	**0.45**
GPT-2_{medium} (**LoRMA**$_+$)	0.3M	**68.4**	**8.63**	46.1	70.6	**2.50**	43.64	**0.38**	**0.53**	49.98	0.38	0.47

Table 12: Certain extra results for NLG. For all the metrics, higher is better except TER.

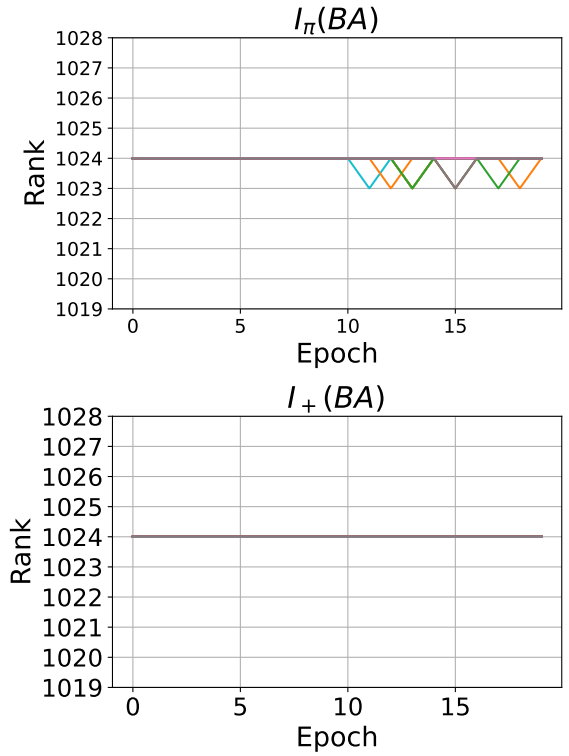

Figure 6: Variation of rank of resultant multiplicative adapters, i.e., $\mathcal{R}(\mathcal{I}_\pi(\mathbf{BA}))$ and $\mathcal{R}(\mathcal{I}_+(\mathbf{BA}))$ across epochs.

Fine-Tuning Method	Rank of Weight Update
LoRA	8
LoRMA$_+$	8
LoRMA$_\pi$	1021

Table 15: Rank of the $\Delta\mathbf{W}$ for Layer 13 of $\text{RoBERTa}_{\text{large}}$ being fine-tuned for CoLA for $r = 8$.

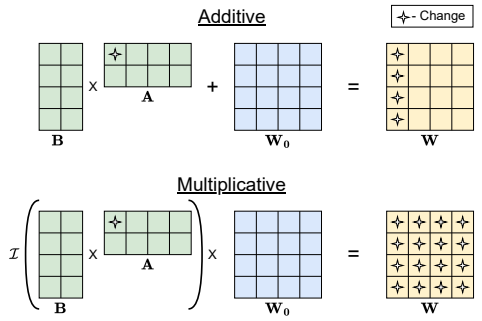

Figure 7: Impact on the resultant matrix on updating a single element in Additive vs Multiplicative updates.

E.2 Ranks of the weight updates

To measure the richness of the weight updates received by the end of the fine-tuning process, we compare the ranks of $\Delta\mathbf{W}$. Table 15 shows the substantial difference in their ranks. In the case of LoRA, the weight update ($\Delta\mathbf{W} = \mathbf{BA}$) is constrained to be of rank $r = 8$. A similar bound exists in the case of **LoRMA**$_+$($\Delta\mathbf{W} = \mathbf{BAW}_0$). For **LoRMA**$_\pi$, no such restriction exists, and the weight update is an almost full rank matrix.

Model and method	Task	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	Optimizer	AdamW							
	Warmup Ratio	0.06							
	LR Schedule	Linear							
RoBERTa$_{\text{base}}$ **LoRMA**$_{+}$	Batch Size	64	64	32	64	64	16	32	16
	Epochs	30	60	30	100	25	25	80	40
	Learning Rate	4E-4	5E-4	4E-4	4E-4	4E-4	5E-4	5E-4	4E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	4	8	8	4	4	8	8	8
	Max Seq. Len.	512							
	Weight decay	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1
RoBERTa$_{\text{base}}$ **LoRMA**$_{\pi}$	Batch Size	64	64	32	64	64	16	32	16
	Epochs	30	60	30	100	25	25	80	40
	Learning Rate	4E-4	5E-4	4E-4	4E-4	4E-4	5E-4	5E-4	4E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	8							
	Max Seq. Len.	512							
	Weight decay	0.1							
RoBERTa$_{\text{base}}$ LoRA	Batch Size	64	64	32	64	64	16	32	16
	Epochs	30	60	30	100	25	25	80	40
	Learning Rate	4E-4	5E-4	4E-4	4E-4	4E-4	5E-4	5E-4	4E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	8							
	Max Seq. Len.	512							
	Weight decay	0.1							
RoBERTa$_{\text{large}}$ **LoRMA**$_{+}$	Batch Size	8	8	4	8	4	4	8	4
	Epochs	10	10	20	20	20	20	20	10
	Learning Rate	3E-4	4E-4	3E-4	3E-4	2E-4	3E-4	4E-4	3E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	8	8	4	4	4	8	4	4
	Max Seq. Len.	128	512	512	128	512	512	512	128
	Weight decay	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1
RoBERTa$_{\text{large}}$ **LoRMA**$_{\pi}$	Batch Size	8	8	4	8	4	4	8	4
	Epochs	10	10	20	20	20	20	20	10
	Learning Rate	3E-4	4E-4	3E-4	3E-4	2E-4	3E-4	4E-4	3E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	8							
	Max Seq. Len.	128	512	512	128	512	512	512	128
	Weight decay	0.1							
RoBERTa$_{\text{large}}$ LoRA	Batch Size	8	8	4	8	4	4	8	4
	Epochs	10	10	20	20	20	20	20	10
	Learning Rate	3E-4	4E-4	3E-4	3E-4	2E-4	3E-4	4E-4	3E-4
	Matrices and r	$r_q = r_v = 8$							
	Scaling α	8							
	Max Seq. Len.	128	512	512	128	512	512	512	128
	Weight decay	0.1							

Table 13: The hyperparameters we used for RoBERTa on the GLUE benchmark.

Task	E2E	WebNLG	DART
		Training	
Optimizer		AdamW	
Weight Decay	0.01	0.01	0.0
Dropout Prob	0.1	0.1	0.0
Batch Size		8	
# Epoch		5	
Warmup Steps		500	
Learning Rate Schedule		Linear	
Label Smooth	0.1	0.1	0.0
Learning Rate ($\mathcal{I}_+$)		0.0002	
Learning Rate ($\mathcal{I}_\pi$)		0.0001	
Matrices and r		$r_q = r_v = 4$	
Scaling α ($\mathcal{I}_+$)		32	
Scaling α ($\mathcal{I}_\pi$)		8	
		Inference	
Beam Size	10/15	10	10
Length Penalty	0.9	0.8	0.8
no repeat ngram size		4	

Table 14: The hyperparameters for GPT-2 M **LoRMA** on E2E, WebNLG and DART.