

LoRA-E²: Effective and Efficient Low-rank Adaptation

Shengkun Zhu
School of Computer
Science
Wuhan University
Wuhan, China
whuzsk66@whu.edu.cn

Jinshan Zeng
School of Management
Xi'an Jiaotong University
Xi'an, China
jsh.zeng@gmail.com

Yiming Wang
School of Computer
Science
Wuhan University
Wuhan, China
yimwang12@whu.edu.cn

Sheng Wang*
School of Computer
Science
Wuhan University
Wuhan, China
swangcs@whu.edu.cn

Yuan Sun
La Trobe Business School
La Trobe University
Melbourne, Australia
yuan.sun@latrobe.edu.au

Shangfeng Chen
School of Computer
Science
Wuhan University
Wuhan, China
brucechen@whu.edu.cn

Yuan Yao
Department of
Mathematics
Hong Kong University of
Science and Technology
Hong Kong, China
yuany@ust.hk

Qiang Yang
The Hong Kong
Polytechnic University
Hong Kong, China
profqiang.yang@polyu.edu.hk

Abstract

Low-rank adaptation (LoRA) has emerged as an efficient fine-tuning technique for large language models, enabling parameter-efficient updates while maintaining task performance. However, LoRA suffers from two key issues: 1) inefficient feature learning when the width n (embedding dimension) is large, and 2) ineffective updates to the adapter matrix A due to the initialization of B as zero. We propose LoRA-E², which utilizes a Gaussian initialization with variance $\Theta(n^{-3/4})$ for A , and employs the Gauss-Seidel iteration to train B and A during the warm-up phase, while employing Jacobi iteration for efficiency during main training. We theoretically show that LoRA-E² enables more stable and efficient feature learning with effective parameter updates over LoRA. Empirically, LoRA-E² achieves consistent gains in both natural language understanding and generation tasks. On the GLUE benchmark with T5-base, it improves performance by 1–10% over LoRA. When fine-tuning Llama 2-7B on MetaMathQA with GSM8K as validation, LoRA-E² surpasses LoRA by 1–2% and converges up to $\sim 3\times$ faster. Code is available at <https://github.com/whu-totemdb/LoRA-E2>.

CCS Concepts

• **Computing methodologies** → **Natural language processing**.

Keywords

Low-rank adaptation, Fine-tuning, Large Language models, Initialization, Gauss-Seidel iteration

ACM Reference Format:

Shengkun Zhu, Jinshan Zeng, Yiming Wang, Sheng Wang*, Yuan Sun, Shangfeng Chen, Yuan Yao, and Qiang Yang. 2026. LoRA-E²: Effective and

Shengkun Zhu and Jinshan Zeng are co-first authors.
*Sheng Wang is the corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License. WWW '26, Dubai, United Arab Emirates
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
<https://doi.org/10.1145/3774904.3792500>

Efficient Low-rank Adaptation. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, April 13–17, 2026, Dubai, United Arab Emirates. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3774904.3792500>

1 Introduction

Large language models (LLMs), such as ChatGPT [1], Gemini [11], and DeepSeek [6], have achieved remarkable success across diverse applications, particularly in the context of web data, where large-scale information retrieval and natural language understanding are crucial [4, 8, 20, 37]. These models, trained on vast amounts of web data, serve as versatile foundations for a wide range of tasks [2, 49]. However, adapting them for specific downstream applications still requires fine-tuning to address task-specific requirements [27]. Modern LLMs are comprised of billions of parameters, making full model fine-tuning computationally prohibitive [19]. This challenge has spurred the development of parameter-efficient fine-tuning (PEFT) methods [10], which selectively update only a small subset of parameters while keeping the majority frozen, significantly reducing computational costs. PEFT methods have not only improved efficiency but also enabled comparable or even superior performance in specialized tasks, making LLMs more accessible and practical for real-world applications, especially in domains heavily reliant on dynamic web data [28, 29, 31, 36, 41].

Among PEFT methods, Low-Rank Adaptation (LoRA) [19] has become a well-known approach, using low-rank updates to the weights of large pretrained models for effective fine-tuning. Given a pretrained weight matrix $W_0 \in \mathbb{R}^{n_1 \times n_2}$ from the pretrained model, LoRA reparameterizes the update as

$$W = W_0 + \frac{\alpha}{r} \Delta W, \quad \text{where } \Delta W = BA,$$

where α is a scaling factor, and $A \in \mathbb{R}^{r \times n_2}$ and $B \in \mathbb{R}^{n_1 \times r}$ are trainable matrices with $\text{rank } r \ll \min\{n_1, n_2\}$. This reparameterization reduces the number of trainable parameters to $r(n_1 + n_2)$, significantly smaller than the full $n_1 \times n_2$ matrix. By freezing W_0 and updating only A and B , LoRA enables efficient adaptation of large-scale models to downstream tasks while maintaining a low computational and memory footprint.

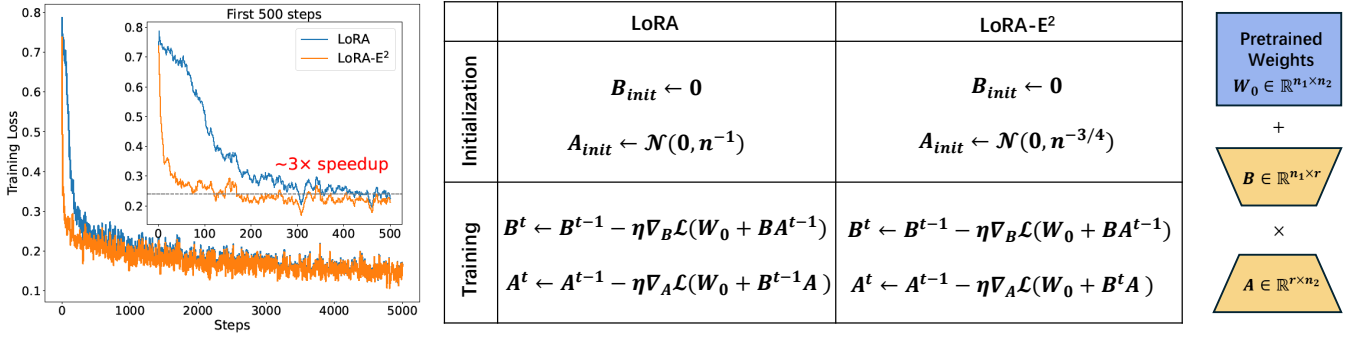


Figure 1: (Left) Training loss curves of Llama 2-7B on the MetaMathQA dataset over training steps. LoRA-E² demonstrates faster convergence and consistently outperforms the standard LoRA. **(Middle)** Detailed initialization and training procedures for LoRA and LoRA-E², exemplified with gradient descent for clarity. The principal distinction lies in LoRA-E² initialization of $A_{init} \sim \mathcal{N}(0, n^{-3/4})$ compared to LoRA initialization of $A_{init} \sim \mathcal{N}(0, n^{-1})$, alongside LoRA-E² adoption of a Gauss-Seidel iteration to alternately optimize matrices B and A . **(Right)** The reparametrization of LoRA and LoRA-E².

However, this initialization fails to guarantee efficient feature learning, resulting in a suboptimal standard LoRA configuration under the infinite width limit of its fine-tuning dynamics. Moreover, it gives rise to a form of *internal instability*, where the intermediate feature representations AZ (for some input Z) exhibit large magnitudes, while the final LoRA output BAZ remains small [14]. A further drawback arises when B is initialized to zero: the LoRA-induced update vanishes at the first training step, preventing informative gradients from propagating to A . Consequently, the model suffers from ineffective early-stage parameter updates, thereby limiting its capacity to fully exploit LoRA’s adaptation potential.

Existing approaches to ensure efficient feature learning in LoRA training primarily focus on increasing the learning rate. Hayou et al. [15] proposed LoRA+, which assigns a much larger learning rate to B than to A , i.e., $\eta_A = \Theta(n^{-1})$, $\eta_B = \Theta(1)$. However, setting high learning rates for trainable parameters can lead to divergence, and this approach does not address the issue of ineffective updates due to the initialization of B as zero. Initializing B with a non-zero value seems like a promising solution, but this prevents fine-tuning from the pretrained model. Wang et al. [44] introduced LoRA-GA, which initializes matrices A and B via singular value decomposition (SVD) of the gradient matrix, thereby ensuring that updates to A and B are directed toward the overall fine-tuning of all parameters. However, this method suffers from high computational costs and, in the absence of immediate gradient deletion strategies [33], incurs significant memory overhead. Therefore, achieving effective and efficient LoRA training remains a considerable challenge.

Without increasing the learning rate, we make simple but effective modifications to the standard LoRA and propose LoRA-E², a novel low-rank adaptation method for fine-tuning LLMs, which provides improvements in both effectiveness and efficiency. For initialization, we still initialize B with zero and A with a Gaussian distribution, but unlike the standard LoRA initialization, the variance is set to $\Theta(n^{-3/4})$. With a learning rate of $\Theta(n^{-1/2})$, we improve the growth of the feature learning BAZ , which typically vanish as $\Theta(n^{-1/2})$ in the standard LoRA, to $\Theta(1)$. Moreover, we

mitigate the inherent internal instability in LoRA training by reducing the growth rate of the internal features AZ from $\Theta(n^{1/2})$ to $\Theta(n^{1/4})$, while simultaneously controlling the update magnitude of B , which diminishes from $\Theta(n^{-1/2})$ to $\Theta(n^{-1/4})$, potentially improving numerical stability.

To address the issue of ineffective updates caused by the zero initialization of B , we alternately update B and A using a Gauss-Seidel iteration [7]. This not only facilitates effective updates but also ensures that the updates for B are more informed and aligned with the most current state of the model, thereby improving the overall optimization process. To further enhance training efficiency, we adopt Gauss-Seidel updates during the warm-up phase and subsequently transition to Jacobi iterations in the main training stage. This hybrid strategy strikes a balance between achieving effective early-stage updates and maintaining scalable, efficient optimization throughout the remainder of training. In Figure 1, we present a straightforward comparison between LoRA-E² and LoRA, focusing on algorithm design, as well as the training loss curves on the MetaMathQA [48] benchmark for Llama 2-7B [42]. Overall, we summarize our main contributions as follows:

- We propose an initialization strategy for the low-rank matrices, where A is initialized with a Gaussian distribution of variance $\Theta(n^{-3/4})$, improving numerical stability and promoting efficient feature learning, especially in the large-width regime.
- During the warm-up phase, we propose a Gauss-Seidel iterative training procedure that alternates updates between B and A , mitigating the issue of ineffective parameter updates arising from the zero initialization of B in LoRA.
- We present both theoretical analysis and empirical evidence showing that LoRA-E² outperforms LoRA in terms of effectiveness and efficiency, achieving up to 3× faster convergence while providing superior performance across multiple benchmarks.

2 Background & Related Work

PEFT has emerged as a powerful paradigm for adapting large pretrained models to downstream tasks, particularly in resource-constrained settings. Early approaches, such as that of Hously et al.

[18], introduced lightweight trainable modules known as adapters into each layer of the model. Subsequent methods, including prefix-tuning [25] and prompt-tuning [24], shifted the focus toward modifying input or hidden representations by appending trainable parameters to existing layers. Among these techniques, LoRA [19] has become one of the most influential and widely adopted PEFT strategies. By injecting low-rank adaptations into weight matrices, LoRA achieves substantial reductions in computational cost while effectively mitigating overfitting and catastrophic forgetting [3].

2.1 Effective LoRA

Effective LoRA focuses on generating high-quality model outputs during fine-tuning, ensuring fast convergence and superior performance across diverse tasks. Several methods have been proposed to improve LoRA’s effectiveness. AdaLoRA [51] introduces dynamic pruning of insignificant weights using Singular Value Decomposition (SVD), reallocating rank to more critical regions within a fixed parameter budget. DoRA [30] improves model expressiveness by incorporating learnable magnitudes into the direction adjustments made by low-rank matrix products. Despite these innovations, vanilla LoRA remains widely adopted due to its robust support in libraries and hardware. As such, advancing LoRA without altering its fundamental structure is essential. Several recent approaches focus on this aspect. For example, rsLoRA [21] introduces a scaling factor that ensures the output is invariant to rank. However, many of these approaches encounter a common issue: initializing matrix B to zero renders the gradient updates to matrix A ineffective during early training. A class of solutions circumvents this issue by initializing B with non-zero values. For instance, PiSSA [35] approximates the original weight matrix W using SVD and uses the resulting factors to initialize A and B . Similarly, LoRA-GA [44] approximates the gradient of W through SVD on sampled gradients, followed by appropriate scaling of the initialized matrices. Nevertheless, these SVD-based methods become computationally expensive and memory-intensive when applied to a large number of LoRA layers. In contrast, our approach retains the conventional initialization of $B = \mathbf{0}$, and instead introduces an alternative update of B and A , effectively overcoming the issue of invalid update and leading to improved training effectiveness.

2.2 Efficient LoRA

Efficient variants of LoRA are primarily designed to reduce the computational overhead of model fine-tuning while maintaining strong downstream performance. Hayou et al. [15] provided a principled framework for determining the LoRA learning rate, showing that increasing the learning rate of the parameter B can promote efficient feature learning. However, excessively large learning rates cause training instability and eventual divergence. Building upon this framework, we demonstrate that efficient feature learning can be achieved by modifying the LoRA initialization alone, without the need to adjust other hyperparameters. There are several other techniques that improve the efficiency of LoRA, but these are orthogonal to the concept of efficient feature learning. One such method involves quantization [9, 26, 38, 45]. Dettmers et al. [9] introduced a quantized version of LoRA, referred to as QLoRA, which reduces computational costs by quantizing pretrained weights to as few

as four bits. Other LoRA variants have been proposed to further reduce the number of trainable parameters [22, 50, 52]. For example, VeRA [22] freezes random weight-tied adapters and instead learns vector scalings of the internal adapter activations, thereby achieving substantial reductions in trainable parameters while preserving performance comparable to standard LoRA.

3 Proposed LoRA-E²

In this section, we present the design of LoRA-E², with particular emphasis on its initialization and training procedures. We first describe the key aspects of LoRA-E² initialization, which are crucial for ensuring both the stability and efficiency of LoRA fine-tuning updates. We then introduce the training methodology of LoRA-E², which employs an alternating update strategy for optimizing B and A , thereby addressing the limitations in standard LoRA. Finally, we provide a detailed description of the overall algorithmic design.

3.1 LoRA-E² Initialization

Given the additive update structure of LoRA, it is crucial to initialize the product BA to zero to ensure that fine-tuning begins from the pretrained model. This can be accomplished by setting one of the matrices, A or B , to zero. Initializing both to zero results in a saddle point where no learning occurs, as the parameter gradients would remain zero. Therefore, one matrix should be initialized to zero, while the other is initialized to a non-zero value. Hayou et al. [15] has theoretically and empirically demonstrated that initializing B to zero typically yields better performance, as initializing A to zero leads to undertraining of B , resulting in suboptimal training. Thus, in this paper, we focus exclusively on the initialization of B to zero.

We consider initializing the trainable weights by assigning the entries A_{ij} independently from Gaussian distribution, $A_{ij} \sim \mathcal{N}(0, \sigma^2)$ and $B_{ij} \leftarrow 0$. To maintain the stability as the input dimension n grows, the variance σ^2 is typically chosen proportional to n^{-1} , a choice grounded in the Central Limit Theorem and widely adopted in initialization strategies such as Kaiming initialization [16] and LeCun initialization [23].

In LoRA training, initializing with a variance of $\sigma^2 = \Theta(n^{-1})$ inevitably leads to inefficiency and instability, unless matrix B is trained with a learning rate significantly larger than that of A (i.e., $\eta_B/\eta_A = \Theta(n)$) [15]. However, using such a large learning rate risks non-convergence. To ensure stability and efficiency without modifying the learning rate, we propose initializing A with a variance of $\sigma^2 = \Theta(n^{-3/4})$. A theoretical justification for this choice is provided in Section 4.

3.2 LoRA-E² Training

For a given LoRA layer, we use Z to denote the input to that layer and \bar{Z} for the output after adding the pretrained weights. More precisely, we write the layer operation as

$$\bar{Z} = (W_0 + \frac{\alpha}{r}BA)Z. \quad (1)$$

We assume that the pretrained weights W_0 remain fixed throughout fine-tuning. The objective is to minimize empirical risk, also referred to as training loss, denoted by $\mathcal{L}(\bar{Z}(A, B), Y)$, where Y represents the ground truth labels. At fine-tuning step t , the gradients

with respect to the low-rank parameters are computed as follows:

$$\begin{aligned}\mathcal{G}_B^t &= \frac{\partial \mathcal{L}^t}{\partial \mathbf{B}} = \frac{\alpha}{r} d\bar{\mathbf{Z}}^{t-1} (\mathbf{A}^{t-1} \mathbf{Z}^{t-1})^\top, \\ \mathcal{G}_A^t &= \frac{\partial \mathcal{L}^t}{\partial \mathbf{A}} = \frac{\alpha}{r} (\mathbf{B}^{t-1})^\top d\bar{\mathbf{Z}}^{t-1} (\mathbf{Z}^{t-1})^\top,\end{aligned}\quad (2)$$

where $d\bar{\mathbf{Z}}$ represents the gradient of the loss function with respect to the layer output features \mathbf{Z} . It is worth noting that at step $t = 1$, the gradient \mathcal{G}_A^1 becomes zero due to the zero initialization of matrix \mathbf{B} . As a result, the update to matrix \mathbf{A} is ineffective at this stage. The root cause of this issue lies in the fact that matrices \mathbf{A} and \mathbf{B} are updated simultaneously. In other words, the optimization adopts the Jacobi iteration [7], where each update step computes the gradient of one matrix assuming the other remains fixed at its previous value. We propose employing Gauss-Seidel iteration to update the LoRA parameters instead of Jacobi iteration. In contrast to standard LoRA, LoRA-E² updates \mathbf{B} first and immediately uses the updated \mathbf{B} to compute the gradient for \mathbf{A} as follows:

$$\begin{aligned}\mathcal{G}_B^t &= \frac{\partial \mathcal{L}^t}{\partial \mathbf{B}} = \frac{\alpha}{r} d\bar{\mathbf{Z}}^{t-1} (\mathbf{A}^{t-1} \mathbf{Z}^{t-1})^\top \\ \mathcal{G}_A^t &= \frac{\partial \mathcal{L}^t}{\partial \mathbf{A}} = \frac{\alpha}{r} (\mathbf{B}^t)^\top d\bar{\mathbf{Z}}^{t-1} (\mathbf{Z}^{t-1})^\top,\end{aligned}\quad (3)$$

where $\bar{\mathbf{Z}}_B^{t-1} = (\mathbf{W}^{t-1} + \frac{\alpha}{r} \mathbf{A}^{t-1} \mathbf{B}^t) \mathbf{Z}^{t-1}$ denotes the intermediate LoRA feature obtained after updating the matrix \mathbf{B} while keeping \mathbf{A} fixed. When using gradient descent as the solver with learning rate η , the updates are given by:

$$\mathbf{A}^t = \mathbf{A}^{t-1} - \eta \mathcal{G}_A^t, \quad \mathbf{B}^t = \mathbf{B}^{t-1} - \eta \mathcal{G}_B^t. \quad (4)$$

Next, we compare the effectiveness of LoRA and LoRA-E² by measuring the evolution of the learned LoRA features over training. We consider the standard LoRA update formulation, where the LoRA feature at iteration t is given by

$$\bar{\mathbf{Z}}^t = (\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{B}^t \mathbf{A}^t) \mathbf{Z}^{t-1} = \bar{\mathbf{Z}}^{t-1} + \delta_1^t + \delta_2^t + \delta_3^t, \quad (5)$$

where the update can be decomposed into three components:

$$\delta_1^t = -\eta \left(\frac{\alpha}{r}\right)^2 \mathbf{B}^{t-1} (\mathbf{B}^{t-1})^\top d\bar{\mathbf{Z}}^{t-1} \|\mathbf{Z}^{t-1}\|^2, \quad (6)$$

$$\delta_2^t = -\eta \left(\frac{\alpha}{r}\right)^2 d\bar{\mathbf{Z}}^{t-1} (\mathbf{A}^{t-1} \mathbf{Z}^{t-1})^\top \mathbf{A}^{t-1} \mathbf{Z}^{t-1}, \quad (7)$$

$$\delta_3^t = \left(\frac{\eta\alpha}{r}\right)^2 d\bar{\mathbf{Z}}^{t-1} (\mathbf{B}^{t-1} \mathbf{A}^{t-1} \mathbf{Z}^{t-1})^\top d\bar{\mathbf{Z}}^{t-1} \|\mathbf{Z}^{t-1}\|^2. \quad (8)$$

The terms δ_1^t and δ_2^t represent ‘‘linear’’ feature updates, which are derived when one low-rank matrix is held fixed while the other is trained. The third term δ_3^t captures the ‘‘multiplicative’’ feature update, reflecting the compounded effect of updating both the matrices \mathbf{A} and \mathbf{B} simultaneously. However, the magnitude of δ_3^t is typically negligible in practice when compared to δ_1^t and δ_2^t , and can thus be safely ignored in several studies [44].

Next, we define the feature of LoRA-E² at iteration t as

$$\bar{\mathbf{Z}}^t = (\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{B}^t \mathbf{A}^t) \mathbf{Z}^{t-1} = \bar{\mathbf{Z}}^{t-1} + \epsilon_1^t + \epsilon_2^t, \quad (9)$$

where the update can be decomposed into two components:

$$\epsilon_1^t = -\eta \left(\frac{\alpha}{r}\right)^2 d\bar{\mathbf{Z}}^{t-1} (\mathbf{A}^{t-1} \mathbf{Z}^{t-1})^\top \mathbf{A}^{t-1} \mathbf{Z}^{t-1}, \quad (10)$$

$$\epsilon_2^t = -\eta \left(\frac{\alpha}{r}\right)^2 \mathbf{B}^t (\mathbf{B}^t)^\top d\bar{\mathbf{Z}}_B^{t-1} \|\mathbf{Z}^{t-1}\|^2. \quad (11)$$

In LoRA-E², we propose an alternative update strategy for the matrices \mathbf{B} and \mathbf{A} , which contrasts with the standard LoRA method that updates both matrices simultaneously. This alternative strategy presents two significant advantages. First, during the initial update, since \mathbf{B} is initialized to zero, we have $\delta_1^1 = \delta_3^1 = 0$. In standard LoRA, this means that only δ_2^1 contributes to the parameter update. However, in LoRA-E², due to the alternative update approach, both ϵ_1^1 and ϵ_2^1 have an active role in the model parameter updates, which facilitates a more gradual and potentially more stable convergence in the early stages of training. Second, LoRA-E² improves the accuracy of the update for \mathbf{A} by leveraging the updated parameter \mathbf{B} during the update of \mathbf{A} . Specifically, since \mathbf{B} is updated before \mathbf{A} , the new values of \mathbf{B} provide a refined basis for updating \mathbf{A} , leading to a more precise and targeted adjustment of the parameters. This alternative update approach ensures that the updates for \mathbf{A} are more informed and aligned with the most current state of the model, ultimately improving the overall optimization process. In Section 5, we present an illustrative example to validate the effectiveness and efficiency of our proposed initialization and training methods.

3.3 Algorithm Design

Although Gauss-Seidel iteration improves stability, it introduces a noticeable computational overhead. Whereas the Jacobi method updates both \mathbf{A} and \mathbf{B} in a single forward-backward pass, the alternative Gauss-Seidel strategy requires two propagation cycles per iteration: one to update \mathbf{A} with the most recent \mathbf{B} , and another to update \mathbf{B} with the updated \mathbf{A} . This effectively doubles the per-iteration cost and reduces training efficiency. To balance the trade-off between stability and efficiency, we apply Gauss-Seidel iteration only during the warm-up phase of LoRA training. Warm-up refers to an initial training phase in which the learning process is deliberately regularized or slowed to stabilize optimization and prepare the model for efficient subsequent convergence [12]. In this early stage, when the model parameters are still adapting and issues such as invalid updates caused by zero initialization of \mathbf{B} are most pronounced, the benefits of stable convergence outweigh the cost of slower iteration. By confining Gauss-Seidel updates to warm-up, we establish a well-conditioned low-rank parameterization. Afterward, training reverts to the Jacobi iteration, which restores full efficiency while retaining the stable initialization achieved earlier. This phase-aware strategy ensures that the advantages of Gauss-Seidel iteration are realized precisely when they are most critical, without undermining the overall practicality of LoRA training.

As outlined in Algorithm 1, the proposed method begins with the initialization of \mathbf{A} and \mathbf{B} (Line 1). Specifically, \mathbf{B} is initialized to zero, and \mathbf{A} is sampled from a normal distribution with variance $n^{-3/4}$. During the warm-up phase, the algorithm alternates between updating \mathbf{A} and \mathbf{B} through Gauss-Seidel iteration (Lines 3–5). In each step of this phase, \mathbf{B} is first updated using the current \mathbf{A} , followed by updating \mathbf{A} with the newly updated \mathbf{B} . The warm-up continues for

Algorithm 1: LoRA-E²

Input: r : LoRA rank; T_{warm} : warm-up steps; T : total steps.
Output: Low-rank matrices A, B .

```

/* Initialization */
1  $B \leftarrow 0, A \sim \mathcal{N}(0, n^{-3/4})$  */

/* Warm-up (Gauss-Seidel alternating) */
2 for  $t \leftarrow 1$  to  $T_{\text{warm}}$  do
3   Sample mini-batch  $\mathcal{B}_t$ 
4   Gauss-Seidel step 1: update  $B$  given current  $A$ 
5   Gauss-Seidel step 2: update  $A$  given updated  $B$ 

/* Main training (Jacobi simultaneous) */
6 for  $t \leftarrow T_{\text{warm}}+1$  to  $T$  do
7   Sample mini-batch  $\mathcal{B}_t$ 
8   Jacobi step: update  $A$  and  $B$  simultaneously
9 return  $A$  and  $B$ 

```

T_{warm} steps to ensure that the low-rank matrices are properly conditioned prior to the main training stage. After the warm-up phase, the algorithm transitions to Jacobi iteration for the main training (Lines 7–8), where both A and B are updated simultaneously within each iteration using a single forward–backward pass.

4 LoRA-E² Finetuning Dynamics

In this section, we aim to rigorously investigate the fine-tuning dynamics of LoRA with a focus on efficient feature learning. Our goal is to prove that LoRA-E² induces efficient and stable feature learning. To this end, we formalize the concepts of LoRA features, stability, and feature learning, and derive precise conditions under which efficient learning is theoretically guaranteed. These results further allow us to identify the critical learning rate thresholds that delineate stable fine-tuning regimes from unstable ones. Since the primary aim of this work is methodological, the theoretical analysis is presented at a physics-level of rigor, intended to distill the key insights. We intentionally omit certain technical assumptions that, while necessary for full mathematical formalization, would obscure the core ideas with excessive complexity.

4.1 Stability and Feature Learning

Our main analysis relies on a careful estimation of the magnitude of several quantities involving LoRA features. Before delving into the full complexity of the setting, we introduce a simplifying assumption to clarify the role of individual LoRA components. When multiple LoRA layers are present, the feature updates are influenced not only by the changes in the low-rank matrices A and B , but also by the latent representations Z and dZ , which are continuously updated during finetuning. To isolate the specific contribution of an individual LoRA layer to feature learning, we adopt a common methodological simplification: we assume that only a single LoRA layer is trainable, while all other LoRA layers remain frozen. Since LoRA layers are typically initialized to zero, this setup is effectively equivalent to incorporating only one active LoRA module in the model. Under this configuration, we can quantitatively evaluate

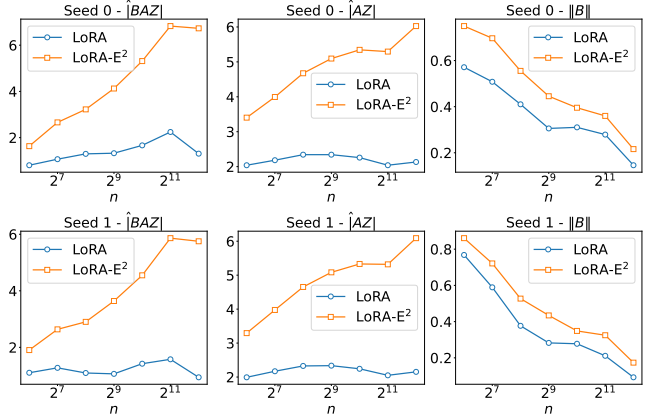


Figure 2: Evolution of the norms of the extracted features Z_A and Z_B averaged over the training data, and the norm of B . We compute the norm $\|B\|$, and the average feature norm $\hat{\|Z_A\|} := \frac{1}{N} \sum_{i=1}^N \|Z_A(x_i)\|$, and similarly for Z_B , where $\{x_i\}_{i=1}^N$ are training samples.

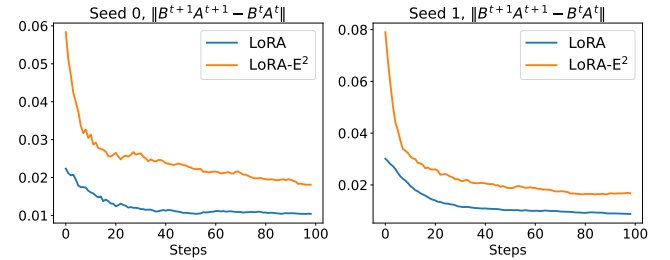


Figure 3: Magnitude of low-rank update steps, measured by $\|B^{t+1}A^{t+1} - B^tA^t\|$, during training for two random seeds. Compared to LoRA, LoRA-E² consistently exhibits larger update magnitudes, indicating more effective parameter updates at each step.

the extent of feature adaptation attributable solely to the trainable LoRA layer throughout the fine-tuning process. This strategy is widely employed in the study of LoRA fine-tuning dynamics [14, 15]. Next, we present a formal definition of the LoRA features defined in Hayou et al. [15].

Definition 1 (LoRA Features). Consider a general neural architecture equipped with a LoRA layer, we define the LoRA features (Z_A, Z_B) as $Z_A = AZ$, and $Z_B = BZ_A = BAZ$. At fine-tuning step t , we use the superscript t to denote the value of LoRA features Z_A^t, Z_B^t , and the weights A^t, B^t .

Stability ensures that no quantity in the network becomes unbounded as the model width increases, which is a critical property when scaling the model. To maintain stability, it is necessary to adjust hyperparameters such as initialization and learning rate as the model size n grows. However, arbitrarily scaling the learning rate with width may result in suboptimal learning dynamics, even if fine-tuning remains stable [14]. To address this issue, we adopt the stability formulation for LoRA features in the asymptotic regime of increasing width [15].

Definition 2 (Feature Stability). We say that LoRA fine-tuning is stable if for all LoRA layers, and all training steps t , we have $Z, Z_B = \Theta(1)$, as the width n goes to infinity.

Feature stability refers to the boundedness of the LoRA output Z_B in the ℓ_2 -norm as the network width increases. Achieving this condition also requires $Z = \Theta(1)$, which is intrinsically linked to pretrained dynamics through its dependence on the pretrained weights W_0 . We assume that the pretrained model is designed to ensure such stability.

As previously noted, feature updates are governed by the terms $\{\epsilon_i^t\}_{i \in \{1,2\}}$. With increasing width n , these updates risk becoming either trivial (vanishing as $n \rightarrow \infty$) or divergent (growing without bound). To prevent such pathological behaviors, it is essential to maintain $\Delta Z_B = \Theta(1)$. This principle underlies the design of parameterizations such as μP [46] and Depth- μP [47], which promote stable feature learning in wide and deep neural networks during pretraining. We refer to the formal definition introduced in [15].

Definition 3 (Feature Learning). We say that LoRA finetuning induces stable feature learning in the limit of large width if the dynamics are stable (Definition 2), and for all fine-tuning steps t , we have $\Delta Z_B := Z_B^{t+1} - Z_B^t = \Theta(1)$.

The update ΔZ_B comprises the terms $\{\epsilon_i^t\}_{i \in \{1,2\}}$. To facilitate effective feature learning, it is necessary that both $\epsilon_1^t, \epsilon_2^t = \Theta(1)$, ensuring that the matrices A and B are actively involved in the update of Z_B . This forms the basis for the definition of efficient learning with LoRA-E² as follows.

Definition 4 (Efficient Learning with LoRA-E²). LoRA fine-tuning is efficient if it is stable (Definition 2), and for all LoRA layers and all fine-tuning steps t , we have

$$\epsilon_i^t = \Theta(1), i \in \{1, 2\}. \quad (12)$$

In the next result, we provide a precise characterization of stability and feature learning when using LoRA-E².

Theorem 1. For any $t > 0$, fine-tuning with LoRA-E² and training by gradient descent with learning rate η yields the following:

- **Stability:** $Z_B^t = O(1)$ if and only if $\eta = O(n^{-1/2})$.
- **Feature Learning:** $\Delta Z_B^t = \Theta(1)$ if and only if $\eta = \Theta(n^{-1/2})$. In this case, we also have $\epsilon_1^t, \epsilon_2^t = \Theta(1)$ (efficient feature learning, Definition 4).
- **Internal instability:** $A^t Z = \Theta(n^{1/4})$ and $B^t = \Theta(n^{-1/4})$ when η lies between $\Theta(n^{-1})$ and $\Theta(n^{-1/2})$.

The proof of Theorem 1 is provided in Appendix A. Theorem 1 reveals that, under the proposed LoRA-E², fine-tuning can be performed with a learning rate scaling as $\Theta(n^{-1/2})$ without inducing instability in the LoRA output Z_B . This reflects an asymptotic characterization of the stability threshold, consistent with prior findings [14], which show that exceeding a certain learning rate results in instability. At this critical rate, feature updates remain efficient (see Definition 4). Nevertheless, this regime entails a caveat: While Z_B remains bounded at $\Theta(1)$, the internal features $A^t Z$ grow at the rate of $\Theta(n^{1/4})$, and the update magnitude of B^t diminishes as $\Theta(n^{-1/4})$, potentially leading to numerical instability. We refer to this phenomenon as *internal instability*, where the divergence

occurs only in the internal representations. Notably, LoRA-E² significantly alleviates this issue. Compared to the standard LoRA formulation in Hayou et al. [14], where Z_A grows as $\Theta(n^{1/2})$ and B^t vanishes at $\Theta(n^{-1/2})$, our approach exhibits markedly improved stability in internal dynamics.

5 Verifying the Results on a Toy Model

To rigorously evaluate the feature dynamics of LoRA and LoRA-E² in a controlled setting, we design a synthetic task based on a multi-layer perceptron (MLP) model. The network architecture consists of an input layer, a single hidden layer, and an output layer. The output is computed via the following transformation:

$$\hat{y} = W_{out} \phi(W_h \phi(W_{in} Z)),$$

where $W_{in} \in \mathbb{R}^{d \times n}$, $W_h \in \mathbb{R}^{n \times n}$, and $W_{out} \in \mathbb{R}^{n \times 1}$ are weight parameters. The function $\phi(\cdot)$ denotes the activation function, for which we use ReLU. We train this model in two stages: the pretraining phase and the fine-tuning phase. In the pretraining phase, the weight matrices W_{in}, W_h, W_{out} are optimized using SGD with loss of the mean squared error (MSE). After pretraining, the pretrained weights are frozen, and we proceed with fine-tuning by introducing low-rank adapters $\Delta W = BA$.

5.1 Efficiency of LoRA-E²

Figure 2 illustrates the dynamics for widths $n \in \{2^6, 2^7, \dots, 2^{12}\}$, across two random seeds, and for both LoRA and LoRA-E². We observe that the magnitude of Z_B is significantly higher with LoRA-E² compared to LoRA. This can be explained by the distinct behavior of the norms of Z_B in both methods: for LoRA-E², $Z_B = \Theta(1)$, while for LoRA, $Z_B = \Theta(n^{-1/2})$. Consequently, LoRA-E² exhibits a larger magnitude of Z_A , as Z_A for LoRA-E² follows $\Theta(n^{1/4})$, while Z_A for LoRA remains constant at $\Theta(1)$. In terms of the norm of B , LoRA-E² demonstrates a larger value, with $B = \Theta(n^{-1/4})$, compared to $B = \Theta(n^{-1/2})$ for LoRA. These differences in behavior, despite the internal instability observed in LoRA-E² during the updates of both A and B , highlight that LoRA-E² maintains efficiency. On the other hand, LoRA exhibits worse internal instability during the update of B , and does not guarantee the efficiency as LoRA-E².

5.2 Effectiveness of LoRA-E²

Figure 3 provides a quantitative comparison of the low-rank update magnitudes between LoRA and our proposed LoRA-E² during training. Specifically, we measure the update magnitude at each step using the Frobenius norm $|B^{t+1} A^{t+1} - B^t A^t|$, averaged across two random seeds. The results clearly show that LoRA-E² consistently exhibits larger update magnitudes than standard LoRA, indicating more substantial parameter adjustments at each step. This behavior comes from the alternative update strategy employed in LoRA-E². By updating B prior to A , we ensure that $B^{t+1} \neq \mathbf{0}$ when computing the gradients for A^{t+1} . Consequently, the update to A is meaningful from the first step, effectively utilizing gradient information and avoiding wasted updates. In contrast, the standard LoRA initializes with $B^0 = \mathbf{0}$, which causes the gradient with respect to A to vanish during the initial update, rendering the first step ineffective. In general, the consistent increase in update magnitudes observed in

Table 1: Performance comparison of various adaptation methods on the GLUE benchmark. We report the mean and standard deviation of the overall Matthew’s correlation for CoLA, Pearson correlation for STS-B, and accuracy for the remaining tasks, averaged over 10 trials. For all metrics, higher values indicate better performance.

Methods	SST-2	MRPC	STS-B	QNLI	CoLA	MNLI	RTE	QQP
LoRA	93.43 ± 0.05	75.82 ± 0.46	92.62 ± 0.01	92.89 ± 0.04	42.61 ± 0.59	85.55 ± 0.04	61.49 ± 0.34	97.21 ± 0.02
LoRA-FA	92.16 ± 0.29	67.57 ± 0.90	91.67 ± 0.08	90.51 ± 0.24	23.31 ± 16.2	71.95 ± 2.80	60.38 ± 0.42	97.15 ± 0.02
rsLoRA	93.96 ± 0.11	73.69 ± 0.12	92.65 ± 0.03	92.98 ± 0.03	41.34 ± 0.35	85.80 ± 0.01	64.14 ± 0.61	97.12 ± 0.06
DoRA	93.46 ± 0.09	69.28 ± 0.31	92.64 ± 0.02	92.90 ± 0.03	29.58 ± 2.50	85.56 ± 0.04	61.49 ± 0.74	97.21 ± 0.01
LoRA+	94.06 ± 0.09	83.15 ± 0.70	92.53 ± 0.03	92.96 ± 0.04	47.82 ± 0.69	85.89 ± 0.01	64.74 ± 0.45	97.26 ± 0.02
LoRA-GA	93.96 ± 0.11	73.69 ± 0.12	92.65 ± 0.03	92.98 ± 0.03	41.34 ± 0.35	85.80 ± 0.01	65.52 ± 0.38	97.12 ± 0.06
LoRA-E²	94.10 ± 0.14	83.91 ± 1.20	92.72 ± 0.09	93.06 ± 0.08	52.1 ± 0.18	85.92 ± 0.09	66.79 ± 0.78	97.35 ± 0.04
LoRA-E² (rank=32)	94.18 ± 0.12	84.21 ± 0.14	92.65 ± 0.06	93.13 ± 0.04	53.09 ± 0.22	85.71 ± 0.05	67.46 ± 0.41	97.37 ± 0.04
LoRA-E² (rank=128)	94.80 ± 0.03	85.62 ± 0.15	92.71 ± 0.02	93.25 ± 0.02	56.10 ± 0.27	86.24 ± 0.08	69.74 ± 0.26	97.36 ± 0.02
LoRA-E² +LoRA+	94.53 ± 0.05	86.76 ± 0.20	92.90 ± 0.04	93.13 ± 0.04	55.59 ± 0.45	86.14 ± 0.02	70.64 ± 0.95	97.71 ± 0.05
LoRA-E² +rsLoRA	94.11 ± 0.05	84.40 ± 0.50	92.75 ± 0.03	93.13 ± 0.08	50.48 ± 1.40	85.96 ± 0.13	68.90 ± 0.64	97.67 ± 0.06

LoRA-E² highlights its improved ability to use gradient information, enabling more effective learning throughout the training.

6 Experiments with Language Models

We present empirical findings on fine-tuning a suite of language models with LoRA-E² across multiple benchmark datasets. We begin by assessing the performance of our approach in comparison to various baselines across two language tasks: NLU and NLG. Next, we conduct an ablation study to demonstrate the efficacy of our proposed methods.

Baselines. In this paper, we leverage several baselines, all of which are based on LoRA and its variants, including LoRA [19], LoRA-FA [52], LoRA+ [15], rsLoRA [21], DoRA [30], and LoRA-GA [44]. LoRA-FA is a method that updates parameter B while keeping parameter A fixed. LoRA+ modifies the learning rate of parameter B , setting it to be greater than the learning rate of parameter A . rsLoRA sets the scaling factor for the adapter as $\frac{\alpha}{\sqrt{r}}$, decomposes the updates of the weights into two parts, magnitude and direction. Direction is handled by standard LoRA, whereas the magnitude is handled by a separate learnable parameter. LoRA-GA applies SVD to decompose the gradient matrix, using the resulting components as the initial values for parameters A and B .

6.1 Natural Language Understanding

Datasets and models. We evaluate the NLU tasks from the GLUE benchmark [43], including MNLI, SST-2, CoLA, QNLI, STS-B, QQP, RTE, and MRPC using the T5-Base model [39].

Results. The evaluation of various adaptation methods on the GLUE benchmark is presented in Table 1. The results clearly indicate that LoRA-E² outperforms all other methods across multiple tasks, achieving the highest scores overall. In contrast, LoRA-FA demonstrates relatively poor performance, particularly on tasks such as MRPC (63.57 ± 0.90) and CoLA (23.31 ± 16.2), which suggests that fixing parameter A while only updating parameter B is ineffective for these tasks. Interestingly, the combination of LoRA-E² with LoRA+ or rsLoRA yields additional improvements, delivering superior performance compared to other adaptation methods. This indicates that LoRA-E², when combined with additional adaptation

techniques, improves performance across various tasks. In conclusion, LoRA-E² and its combinations, such as LoRA-E² + rsLoRA, demonstrate superior performance across the GLUE benchmark, underscoring the efficacy of our approach. The performance of LoRA-E² with various ranks further validates the versatility and effectiveness of our approach. Specifically, when LoRA-E² is applied with a rank of 32, it yields an outstanding performance, with the highest score on SST-2 (94.18 ± 0.12) and consistent improvements across other tasks, including MNLI (85.92 ± 0.09) and QQP (97.35 ± 0.04). In contrast, increasing the rank to 128 leads to even better performance, with noticeable improvements across various tasks, particularly in CoLA and QQP. This highlights that LoRA-E² continues to enhance its performance with higher ranks, further demonstrating the scalability and robustness of our method across different configurations.

Figure 4 (see Appendix B.4) illustrates the training loss trajectories of various parameter-efficient fine-tuning methods across multiple GLUE benchmark tasks. LoRA+ is excluded from comparison because its increased learning rate leads to accelerated but non-comparable convergence dynamics. Across all tasks, LoRA-E² exhibits consistently superior convergence behavior, achieving both the *fastest decline* in loss and the *lowest final training loss*. This demonstrates its strong capacity for rapid adaptation and efficient feature learning. The advantage is particularly evident in tasks such as SST2, QNLI, and MNLI, where LoRA-E² reaches convergence significantly earlier than other methods. The smoothness of its loss curves also indicates enhanced training stability, aligning with our theoretical analysis of improved gradient conditioning. In contrast, LoRA-FA converges more slowly and attains higher steady-state losses, especially on MRPC and STSB, highlighting that fixing parameter A while updating only B constrains the representational flexibility required for these tasks. rsLoRA and DoRA achieve intermediate performance, better than LoRA-FA but still inferior to LoRA-E², suggesting that while their design alleviates part of the inefficiency in standard LoRA, they fall short in maintaining stable yet expressive feature updates. Overall, these results confirm that LoRA-E² not only accelerates convergence but also achieves a more optimal training trajectory across diverse task types, validating its effectiveness as a robust and scalable PEFT framework.

Table 2: Results of fine-tuning Llama 2-7B using various LoRA variants, evaluated on the GSM8K dataset with three different random seeds.

Methods	GSM8K seed 0	GSM8K seed 1	GSM8K seed 2
LoRA	57.47	57.09	57.85
LoRA-FA	54.97	56.48	56.79
rsLoRA	57.32	58.15	57.85
Dora	57.70	56.56	56.79
LoRA+	57.85	58.17	57.60
LoRA-GA	58.00	58.23	58.16
LoRA-E²	58.07	58.30	59.36
LoRA-E² ($r = 32$)	58.21	58.40	59.06
LoRA-E² ($r = 128$)	58.42	59.56	58.81
LoRA-E² +LoRA+	58.44	58.33	59.61
LoRA-E² +rsLoRA	58.30	58.23	57.47

6.2 Natural Language Generation

Datasets and models. We train our model on a 100K subset of MetaMathQA [48], a dataset derived from other math instruction tuning datasets such as GSM8K [5] and MATH [17], which offers increased complexity and diversity. We select data sampled from the GSM8K training set and apply appropriate filtering. The accuracy is reported on the GSM8K evaluation set.

Results. The results in Table 2 highlight the superior performance of LoRA-E² across all three random seeds when fine-tuning Llama 2-7B. LoRA-E² consistently outperforms all other methods, achieving the highest scores in each seed, with a notable improvement in seed 2. In comparison, LoRA-FA shows the weakest performance across all seeds, suggesting that fixing one parameter while updating the other is less effective. Methods like rsLoRA and Dora perform better than LoRA-FA, but still fall short of LoRA-E², indicating the importance of more flexible adaptation strategies. The combination of LoRA-E² with LoRA+ or rsLoRA results in marginal improvements, but LoRA-E² alone remains the most effective approach. These findings demonstrate that LoRA-E² is the optimal method for achieving the best performance on the NLG task.

We also evaluate the impact of different rank settings on the performance of LoRA-E². As shown in Table 2, both LoRA-E² ($r = 32$) and LoRA-E² ($r = 128$) achieve consistently strong results, surpassing all baseline methods across the three random seeds. Importantly, increasing the rank from 32 to 128 yields a clear performance gain, with average improvements of approximately 0.5–1.0% across seeds. For instance, while LoRA-E² ($r = 32$) achieves 58.21, 58.40, and 59.06 on seeds 0, 1, and 2, respectively, the higher-rank LoRA-E² reaches 58.42, 59.56, and 58.81. These results highlight the scalability of LoRA-E², where a larger rank enables richer parameterization and more effective adaptation to the downstream task.

6.3 Ablation Study

The results presented in Table 3 clearly demonstrate the effectiveness of LoRA-E², which integrates both the proposed initialization and the Gauss-Seidel iterative training for **B** and **A**. Comparing LoRA, Init, Training, and LoRA-E² across various datasets reveals that each component independently contributes to improved

Table 3: Comparison of performance across GLUE and GSM8K. LoRA is the standard method, Init uses our proposed initialization, and Training applies Gauss-Seidel iteration to train A and B. LoRA-E² combines both techniques.

Datasets	LoRA	Init	Training	LoRA-E ²
SST-2	93.43 ± 0.05	93.49 ± 0.02	93.92 ± 0.16	94.10 ± 0.14
MRPC	75.82 ± 0.46	76.31 ± 0.38	83.66 ± 0.31	83.91 ± 1.20
STS-B	92.62 ± 0.01	92.70 ± 0.02	92.67 ± 0.06	92.72 ± 0.09
QNLI	92.89 ± 0.04	92.95 ± 0.03	92.97 ± 0.05	93.06 ± 0.08
CoLA	42.61 ± 0.59	42.72 ± 0.33	51.86 ± 1.40	52.10 ± 0.18
MNLI	85.55 ± 0.04	85.65 ± 0.043	85.62 ± 0.01	85.92 ± 0.09
RTE	61.49 ± 0.34	62.31 ± 0.34	65.34 ± 0.51	66.79 ± 0.78
QQP	97.21 ± 0.02	97.22 ± 0.01	97.30 ± 0.03	97.35 ± 0.04
GSM8K	57.47 ± 0.31	57.56 ± 0.56	57.96 ± 0.34	58.58 ± 0.56

performance. Specifically, employing the proposed initialization (Init) or adopting Gauss-Seidel training (Training) yields consistent gains over the standard LoRA baseline. For example, on CoLA, Init achieves a noticeable increase in accuracy (42.72 vs. 42.61), while Training provides a substantial improvement (51.86), highlighting the benefit of alleviating ineffective updates in early training. Similarly, on MRPC, Training markedly enhances performance (83.66 vs. 75.82), underscoring the importance of iterative optimization.

More importantly, the integration of both techniques in LoRA-E² leads to the best overall results, consistently outperforming all other variants. On SST-2, LoRA-E² achieves 94.10 accuracy, surpassing both Init (93.49) and Training (93.92). A similar trend is observed on MRPC, where LoRA-E² reaches 83.91, exceeding Init (76.31) and Training (83.66). Across all datasets, including GSM8K, LoRA-E² achieves the highest scores.

These findings confirm that both initialization and Gauss-Seidel updates are essential for enhancing LoRA’s effectiveness, but their combination is particularly powerful. By integrating these strategies, LoRA-E² achieves substantial improvements in stability, convergence, and task performance, thereby validating its design as an effective and efficient fine-tuning framework.

7 Conclusions

LoRA-E² enhanced LoRA fine-tuning by addressing inefficient feature learning and early updates through a principled initialization and Gauss-Seidel optimization. Our framework provided provable stability and efficiency guarantees while delivering faster convergence and superior performance across NLU and NLG benchmarks compared to standard LoRA variants.

Acknowledgment

This work was supported by the National Key R&D Program of China (2023YFB4503600), National Natural Science Foundation of China (62202338), Key R&D Program of Hubei Province (2023BAB081). This work of Jinshan Zeng is supported in part by the National Natural Science Foundation of China (62376110) and the Major Key Project of PCL (PCL2024A06). Yao Yuan was partially supported by the Research Grants Council (RGC) of Hong Kong, SAR, China (GRF-16308321), the NSFC/RGC Joint Research Scheme Grant N_HKUST635/20, and gift funds from Edge Science and Zhuhai Kehui.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Jinheon Baek, Nirupama Chandrasekaran, Silviu Cucerzan, Allen Herring, and Sujay Kumar Jauhar. 2024. Knowledge-augmented large language models for personalized contextual query suggestion. In *WWW*. 3355–3366.
- [3] Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673* (2024).
- [4] Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2025. Large language models empowered personalized web agents. In *WWW*. 198–215.
- [5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).
- [6] DeepSeek-AI. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954* (2024). <https://github.com/deepseek-ai/DeepSeek-LLM>
- [7] James W Demmel. 1997. *Applied numerical linear algebra*. SIAM.
- [8] Yang Deng, An Zhang, Yankai Lin, Xu Chen, Ji-Rong Wen, and Tat-Seng Chua. 2024. Large language model powered agents in the web. In *WWW*. 1242–1245.
- [9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS*.
- [10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.* 5, 3 (2023), 220–235.
- [11] Team Gemini. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
- [12] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation. In *ICLR*.
- [13] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. 2019. On the Impact of the Activation function on Deep Neural Networks Training. In *ICML*, Vol. 97. 2672–2680.
- [14] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. The impact of initialization on lora finetuning dynamics. *NeurIPS* 37 (2024), 117015–117040.
- [15] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient Low Rank Adaptation of Large Models. In *ICML*. 17783–17806.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.
- [17] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874* (2021).
- [18] 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 *ICML*, Vol. 97. PMLR, 2790–2799.
- [19] 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 *ICLR*.
- [20] Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, Fritz Behr, and C Lee Giles. 2010. Exploring web scale language models for search query processing. In *WWW*. 451–460.
- [21] Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732* (2023).
- [22] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. VeRA: Vector-based Random Matrix Adaptation. In *ICLR*.
- [23] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 2012. Efficient BackProp. In *Neural Networks: Tricks of the Trade - Second Edition*. Vol. 7700. Springer, 9–48.
- [24] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *EMNLP*. 3045–3059.
- [25] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL*. 4582–4597.
- [26] Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2024. LoftQ: LoRA-Fine-Tuning-aware Quantization for Large Language Models. In *ICLR*.
- [27] Shao-En Lin, Brian Liu, Miao-Chen Chiang, Ming-Yi Hong, Yu-Shiang Huang, Chuan-Ju Wang, and Che Lin. 2025. BETA: Behavior-enhanced Item Tagging with Finetuned Large Language Models. In *WWW*. 4996–5009.
- [28] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. In *NeurIPS*.
- [29] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024. When MOE Meets LLMs: Parameter Efficient Fine-tuning for Multi-task Medical Applications. In *SIGIR*. ACM, 1104–1114.
- [30] 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. In *ICML*.
- [31] Zheng Liu, Yujia Zhou, Yutao Zhu, Jianxun Lian, Chaozhao Li, Zhicheng Dou, Defu Lian, and Jian-Yun Nie. 2024. Information retrieval meets large language models. In *WWW*. 1586–1589.
- [32] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR*.
- [33] Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. 2024. Full Parameter Fine-tuning for Large Language Models with Limited Resources. In *ACL*. 8187–8198.
- [34] 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>.
- [35] Fanxu Meng, Zhaohui Wang, and Muhun Zhang. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *NeurIPS*.
- [36] Carina Negreanu, Alperen Karaoglu, Jack Williams, Shuang Chen, Daniel Fabian, Andrew Gordon, and Chin-Yew Lin. 2022. Rows from Many Sources: Enriching row completions from Wikidata with a pre-trained Language Model. In *WWW*. 1272–1280.
- [37] Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *WWW*. 20–28.
- [38] Haotong Qin, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. 2024. Accurate LoRA-Finetuning Quantization of LLMs via Information Retention. In *ICML*.
- [39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
- [40] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. Deep Information Propagation. In *ICLR*.
- [41] Yiwen Tang, Ray Zhang, Zoey Guo, Xianzheng Ma, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. 2024. Point-PEFT: Parameter-Efficient Fine-Tuning for 3D Pre-trained Models. In *AAAI*. 5171–5179.
- [42] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shrutu Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- [43] Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).
- [44] Shaowen Wang, Linxi Yu, and Jian Li. 2024. LoRA-GA: Low-Rank Adaptation with Gradient Approximation. In *NeurIPS*.
- [45] Yifei Xia, Fangcheng Fu, Wentao Zhang, Jiawei Jiang, and Bin Cui. 2024. Efficient Multi-task LLM Quantization and Serving for Multiple LoRA Adapters. In *NeurIPS*.
- [46] Ge Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer. In *NeurIPS*. 17084–17097.
- [47] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. 2023. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244* (2023).
- [48] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284* (2023).
- [49] Chuxu Zhang, Kaize Ding, Jundong Li, Dongkuan Xu, Haoyu Wang, Derek Zhiyuan Cheng, and Huan Liu. 2025. Resource-Efficient Learning for the Web. In *WWW*. 77–80.
- [50] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303* (2023).
- [51] 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 *ICLR*.
- [52] Jiacheng Zhu, Kristjan H. Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brülé Gabrielson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in Low-Rank Adapters of Foundation Models. In *ICML*.

A Theory and Proofs

A.1 Scaling of Neural Networks

Scaling refers to systematically enlarging certain components of a model to enhance its performance. This may involve increasing model width (e.g., embedding dimensions), depth (e.g., number of layers), compute resources, training iterations, or a combination thereof. This paper specifically focuses on scaling the model’s width, a direction motivated by the observation that large language and vision models typically operate at large widths.

It is well established that as the model width n increases, initialization strategies and learning dynamics must be carefully tuned to preserve numerical stability and enable efficient training. For instance, to prevent excessively large pre-activations with growing n , the variance of weight initialization should scale as $1/n$, as seen in He et al. [16]. A principled way to derive such rules involves analyzing the statistical behavior of core model components (e.g., pre-activations) in the asymptotic regime and accordingly adjusting initialization, learning rates, and architectural design to maintain desirable training dynamics as $n \rightarrow \infty$.

Building on this foundation, Yang et al. [47] introduced the *Maximal Update Parameterization* (μP), which formalizes a family of scaling principles for initialization, learning rates, and network architectures. These principles aim to preserve training stability and enable effective representation learning as model width tends to infinity. In this framework, stability is characterized by the condition $Z_l^i = \Theta(1)$ across all layers l and units i , where $\Theta(\cdot)$ reflects asymptotic behavior with respect to width n . Similarly, efficient feature learning is ensured by maintaining $\Delta Z_l = \Theta(1)$, where Δ denotes the change in features induced by gradient descent. To satisfy both conditions during training, μP prescribes that hidden layer weights be initialized with variance $\Theta(n^{-1/2})$ and that their updates scale as $\Theta(n^{-1})$. Input weights and their updates should remain of order $\Theta(1)$, while output weights are initialized as $\Theta(n^{-1})$ and updated proportionally. These prescriptions distinguish μP from conventional parameterizations which may suffer from instability or vanishing feature learning in the infinite-width limit. For example, in the Neural Tangent Kernel regime, feature updates shrink as $\Delta Z_l = \Theta(n^{-1/2})$, i.e., no feature learning as $n \rightarrow \infty$.

A.2 Revisiting the γ -Operator for Scaling Analysis

When analyzing scaling behavior in neural networks, a central concern is understanding how various quantities evolve as the model width n increases. Specifically, we are interested in identifying the asymptotic scaling laws that govern the behavior of different components in the computational graph. For example, when the width n of a network is scaled, we examine how internal variables such as activations, gradients, and weights change with n . This methodology underpins many principled approaches to model scaling and has informed foundational work on initialization strategies [40], activation function design [13], and network parametrization schemes [47].

To systematically analyze such scaling trends, we adopt the γ -operator, which tracks the exponent governing the polynomial scaling of a quantity v with respect to n . We write $v = \Theta(n^{\gamma[v]})$, where $\gamma[v]$ captures the asymptotic exponent of v . Basic operations with this operator obey intuitive algebraic rules:

- **Zero:** If $v = 0$, we define $\gamma[v] = -\infty$ (interpreted as the limit $\gamma[n^{-\beta}] \rightarrow -\infty$ as $\beta \rightarrow \infty$).
- **Multiplication:** For any two scalar quantities v, v' , it holds that $\gamma[v \times v'] = \gamma[v] + \gamma[v']$.
- **Addition:** For general v, v' , we have $\gamma[v + v'] = \max(\gamma[v], \gamma[v'])$, unless $v' = -v$, a zero probability event when v and v' are random variables that are not perfectly (negatively) correlated, which is the case in most situations where we make use of this formula.

We now have the complete formal machinery to proceed with our asymptotic analysis.

A.3 Proof of Theorem 1

In this section, we present an informal yet conceptually insightful argument supporting Theorem 1. Our approach is based on a key assumption as follows.

Assumption 1. Under the same conditions described in Section 4, we assume that at the training iteration t , the variables Z and dZ remain on the order of $\Theta(1)$.

Assumption 1 concerns the parameterization adopted during pre-training. We posit that the conditions stated are satisfied throughout our analysis. Let us now provide a proof for the main theorem.

Theorem 2 (Restate of Theorem 1). *Under Assumption 1, for any $t > 0$, fine-tuning with LoRA-E² and training by gradient descent with learning rate η yields the following:*

- **Stability:** $Z_B^t = O(1)$ if and only if $\eta = O(n^{-1/2})$.
- **Feature Learning:** $\Delta Z_B^t = \Theta(1)$ if and only if $\eta = \Theta(n^{-1/2})$. In this case, we also have $\epsilon_1^t, \epsilon_2^t = \Theta(1)$ (efficient feature learning, Definition 4).
- **Internal instability:** $A^t Z = \Theta(n^{1/4})$ and $B^t = \Theta(n^{-1/4})$ when η lies between $\Theta(n^{-1})$ and $\Theta(n^{-1/2})$.

PROOF OF THEOREM 2. To facilitate effective feature learning, it is essential that both ϵ_1^t and ϵ_2^t remain on the order of $\Theta(1)$, indicating that matrices A and B are both actively updated and significantly influence the evolution of Z_B . To substantiate this requirement, we examine

how perturbations in A and B impact the LoRA-modified feature $Z_B = \mathbf{BAZ}$. We rewrite the contributions ϵ_1^t and ϵ_2^t as:

$$\begin{cases} \epsilon_1^t = -\eta \left(\frac{\alpha}{r}\right)^2 dZ(\mathbf{A}^{t-1}\mathbf{Z})^\top \mathbf{A}^{t-1}\mathbf{Z}, \\ \epsilon_2^t = -\eta \left(\frac{\alpha}{r}\right)^2 \mathbf{B}^t (\mathbf{B}^t)^\top dZ \|\mathbf{Z}\|^2. \end{cases}$$

To ensure efficiency, it is required that $\epsilon_i^t = \Theta(1)$ for all t and $i \in \{1, 2\}$ and $\mathbf{B}^t \mathbf{A}^t \mathbf{Z} = \mathcal{O}(1)$. Applying Assumption 1 and the basic expressions derived in Appendix A.2, this condition further implies that for all t , we have:

$$\begin{cases} \gamma[\eta] + 2\gamma[\mathbf{B}^t] + 1 = 0 \\ \gamma[\eta] + 2\gamma[\mathbf{A}^{t-1}\mathbf{Z}] = 0 \\ \gamma[\mathbf{B}_{t-1}] + \gamma[\mathbf{A}_{t-1}\mathbf{Z}] = 0. \end{cases} \quad (13)$$

With LoRA-E² initialization, we have $\gamma[\mathbf{B}^0] = -\infty$ and $\gamma[\mathbf{A}^0\mathbf{Z}] = \frac{1}{4}$. As a result, we have for all $t > 0$

$$\begin{cases} \gamma[\mathbf{B}^t] = \max(\gamma[\mathbf{B}^{t-1}], \gamma[\eta] + \gamma[\mathbf{A}^{t-1}\mathbf{Z}]) \\ \gamma[\mathbf{A}^t\mathbf{Z}] = \max(\gamma[\mathbf{A}^{t-1}\mathbf{Z}], \gamma[\eta] + \gamma[\mathbf{B}^t] + 1) \end{cases} \quad (14)$$

Starting from $t = 1$, under the LoRA-E² initialization, we have $\gamma[\mathbf{B}^1] = \gamma[\eta] + \frac{1}{4}$ and $\gamma[\mathbf{A}^1\mathbf{Z}] = \max\left(\frac{1}{4}, 2\gamma[\eta] + \frac{5}{4}\right)$. We consider two cases:

- **Case 1:** If $2\gamma[\eta] + \frac{5}{4} > \frac{1}{4}$, it follows that $\gamma[\eta] > -\frac{1}{2}$. By substituting $\gamma[\mathbf{B}^1]$ and $\gamma[\mathbf{A}^1\mathbf{Z}]$ into Equation (13), we obtain $\gamma[\eta] = -\frac{1}{2}$.
- **Case 2:** If $2\gamma[\eta] + \frac{5}{4} \leq \frac{1}{4}$, we deduce that $\gamma[\eta] \leq -\frac{1}{2}$. Similarly, substituting $\gamma[\mathbf{B}^1]$ and $\gamma[\mathbf{A}^1\mathbf{Z}]$ into Equation (13) yields $\gamma[\eta] = -\frac{1}{2}$.

Substituting $\gamma[\eta] = -\frac{1}{2}$ into Equation (14), we obtain $\gamma[\mathbf{A}^t\mathbf{Z}] = \frac{1}{4}$ and $\gamma[\mathbf{B}^t] = -\frac{1}{4}$ for any $t > 0$. This indicates that stability can be maintained when the learning rate scales as $\Theta(n^{-1/2})$. However, such a learning rate introduces internal instability, as the feature Z_A grows unboundedly with network width. Specifically, under this scaling, $\gamma[\mathbf{A}^t\mathbf{Z}] = 1/4$, implying $\mathbf{A}^t\mathbf{Z} = \Theta(n^{1/2})$, which diverges as $n \rightarrow \infty$. Nonetheless, this growth is compensated by the fact that $\gamma[\mathbf{B}^t] = -1/4$, i.e., $\mathbf{B}^t = \Theta(n^{-1/4})$, thereby preserving overall stability. Moreover, in this case, feature learning is efficient according to Definition 4, with $\epsilon_1^t = \Theta(1)$ and $\epsilon_2^t = \Theta(1)$, thereby completing the proof. \square

B Complete Experiments

This section complements the empirical results reported in the main text.

B.1 Empirical Details of Toy Examples

To rigorously evaluate the feature dynamics of LoRA and LoRA-E² in a controlled setting, we design a synthetic task based on a multi-layer perceptron (MLP) model. The network architecture consists of an input layer, a single hidden layer, and an output layer. The output is computed via the following transformation:

$$\hat{\mathbf{y}} = \mathbf{W}_{out}\phi(\mathbf{W}_h\phi(\mathbf{W}_{in}\mathbf{Z}))$$

where $\mathbf{W}_{in} \in \mathbb{R}^{d \times n}$, $\mathbf{W}_h \in \mathbb{R}^{n \times n}$, and $\mathbf{W}_{out} \in \mathbb{R}^n$ are weight parameters. The function $\phi(\cdot)$ denotes the activation function, for which we use ReLU. We train this model in two stages: the pretraining phase and the fine-tuning phase. In the pretraining phase, the weight matrices $\mathbf{W}_{in}, \mathbf{W}_h, \mathbf{W}_{out}$ are optimized using SGD with loss of the mean squared error (MSE). After pretraining, the pretrained weights are frozen, and we proceed with fine-tuning by introducing low-rank adapters $\Delta\mathbf{W} = \mathbf{BA}$.

Data Generation. We generate synthetic input data $X \in \mathbb{R}^{N \times d}$ with $d = 10$ and $N = 1000$, where each row is sampled i.i.d. from a standard normal distribution $X \sim \mathcal{N}(0, I_d)$. The targets $Y \in \mathbb{R}^{N \times 1}$ are also sampled from $\mathcal{N}(0, I_1)$, and remain fixed for each experiment seed.

Pretraining. The base MLP is trained using SGD for 100 steps with a learning rate of 0.01. All model weights are initialized with Gaussian distributions scaled by: $\mathbf{W}_{in} \sim \mathcal{N}(0, 1/d)$, $\mathbf{W}_h \sim \mathcal{N}(0, 1/n)$, and $\mathbf{W}_{out} \sim \mathcal{N}(0, 1/n)$. After pretraining, the parameters of $\mathbf{W}_{in}, \mathbf{W}_h$, and \mathbf{W}_{out} are frozen.

LoRA Fine-Tuning. We apply LoRA-style low-rank adaptation on the second layer. Two variants are considered:

- LoRA: $A \sim \mathcal{N}(0, 1/n)$
- LoRA-E²: $A \sim \mathcal{N}(0, n^{-3/4})$

Matrix B is initialized to zero. For LoRA-E², the updates alternate between optimizing B (with fixed A) and optimizing A (with fixed B). The learning rate is set to $\eta = n^{-1/2}$, and optimization is conducted via SGD for 100 iterations. All experiments are repeated for $n \in \{2^6, 2^7, \dots, 2^{12}\}$ and two random seeds.

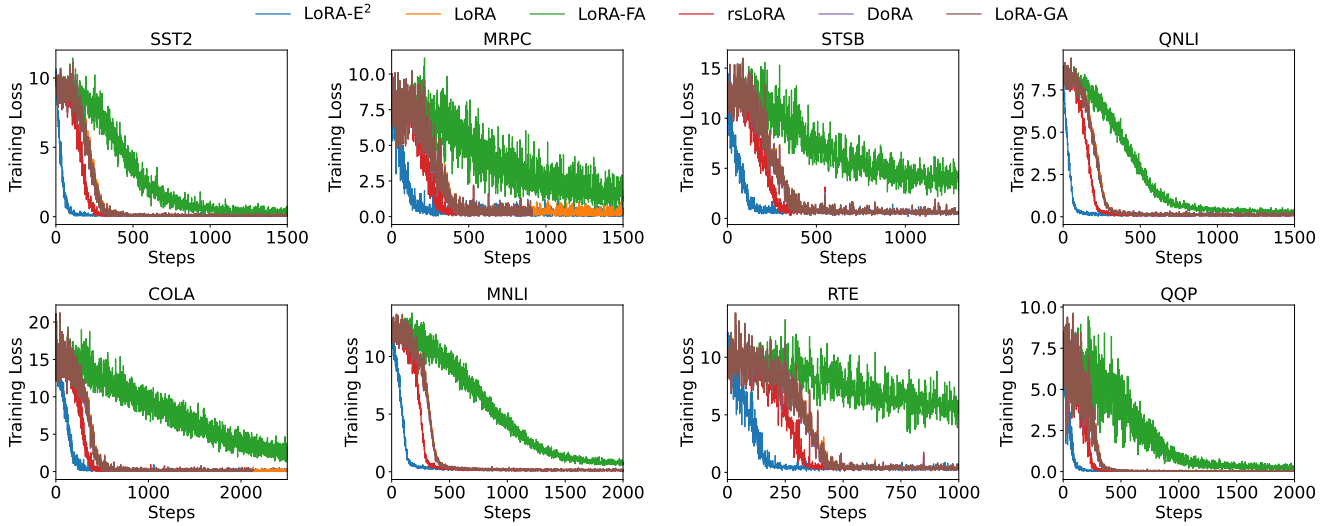


Figure 4: Comparison of the training loss across various baselines on the GLUE benchmark.

B.2 Empirical Details of Natural Language Understanding

The training is performed using the AdamW optimizer [32] with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and weight decay set to 0. For the standard LoRA, and its variants, a learning rate of 1×10^{-4} is employed, along with a warm-up ratio of 0.03 and cosine learning rate decay, the rank is set to $r = 8$, and the scaling factor α is set to 16. The LoRA adaptation is applied to all linear layers.

B.3 Empirical Details of Natural Language Generation

We conduct experiments on the LLaMA 2-7B model using the following training and evaluation settings:

- **Training Algorithm.** We employ the AdamW optimizer [32] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and zero weight decay. For LoRA-based methods, a learning rate of 2×10^{-5} is used, with a cosine learning rate schedule with a warm-up ratio of 0.03. All variants adopt the same warm-up strategy and decay scheduling.
- **Numerical Precision.** The base model uses bf16 precision during inference. However, LoRA’s low-rank matrices A and B are maintained in fp32 precision throughout training, consistent with the PEFT framework [34].
- **LoRA Configuration.** Across all experiments, we set the LoRA rank to $r = 8$ and the scaling factor $\alpha = 16$. LoRA layers are inserted into all linear transformations except embeddings, layer normalizations, and the output head.
- **General Hyperparameters.** Training is performed for a single epoch ($E = 1$) using a batch size of 1 with gradient accumulation of 4 steps. The sequence length is fixed at 1024.

B.4 Complete Results

Figure 4 illustrates the training loss trajectories of various parameter-efficient fine-tuning methods across multiple GLUE benchmark tasks. LoRA+ is excluded from comparison because its increased learning rate leads to accelerated but non-comparable convergence dynamics. Across all tasks, LoRA-E² exhibits consistently superior convergence behavior, achieving both the *fastest decline* in loss and the *lowest final training loss*. This demonstrates its strong capacity for rapid adaptation and efficient feature learning. The advantage is particularly evident in tasks such as SST2, QNLI, and MNLI, where LoRA-E² reaches convergence significantly earlier than other methods. The smoothness of its loss curves also indicates enhanced training stability, aligning with our theoretical analysis of improved gradient conditioning. In contrast, LoRA-FA converges more slowly and attains higher steady-state losses, especially on MRPC and STSB, highlighting that fixing parameter A while updating only B constrains the representational flexibility required for these tasks. rsLoRA and DoRA achieve intermediate performance, better than LoRA-FA but still inferior to LoRA-E², suggesting that while their design alleviates part of the inefficiency in standard LoRA, they fall short in maintaining stable yet expressive feature updates.