Final Report- A Survey on LoRA and its Variants' preservation to Llama3 8B's math and logical reasoning abilities

Jian Lu
University of Rochester
jlu59@u.rochester.edu

Hanzhang Yin
University of Rochester
hyin12@u.rochester.edu

Tianyi Zhou
University of Rochester
tzhou25@u.rochester.edu

Abstract

LoRA (Low-Rank Adaptation) has emerged as a strategy for efficiently updating dense neural network layers with plug-gable low-rank matrices derived from the original matrix of the network's parameters. Moreover, it has significant advantages in cross-task generalization and privacy preservation via different training conditions. Hence, gaining a well-rounded understanding of LoRA and its other variants is essential. This survey will focus on the LoRA variants' ***(1) Downstream adaptation****,* ***(2) math tasks performance****, and* ***(3) Reasoning skills given different difficulty levels*** *assessing their abilities on Large Language Model. Specifically, we will run several benchmark tests on different variants of LoRA (including LoRA itself) using model LLaMA-3 (8 bit) on dataset GSM8k and GSM-Plus, dataset that primarily focusing on leverage model's mathematical inference ability, and analyze their behavior. At last, this survey will discuss the potential future directions in the field, propagating some novel ideas worth attention.*

1. Introduction

Due to the fast boost and growing attention to LLMs, the parameter scales of the current pre-trained LLMs are rapidly increasing, enabling better generalization of various tasks with enhanced emergent abilities. In the past few years, the parameter scales of pre-trained language models have increased in a scale at least for about thousands of times (e.g., BERT with 330M parameters [5] to GPT4 with 1.8 trillion parameters [20]). Although these foundational models with significant parameters showed a significant ability in general problem-solving, their abilities on some downstream tasks are still limited due to the knowledge boundaries. Therefore, applying further fine-tuning to a specific focus is essential to expand the knowledge boundaries for such foundational models.

However, the novel approach of model fine-tuning is to fine-tune the full parameters of an LLM, known as full fine-tuning, which is extremely computationally and memory-expensive. For instance, fully fine-tuning *LLaMA2-7B* [3] requires approximately 62GB of memory, which is way beyond the scope that regular commercial GPUs can provide (e.g., RTX4090 provides only 24 GB GDDR6X memory). Hence, various parameter-efficient fine-tuning (PEFT) methods have been proposed to reduce the high computation cost. Regarding whether extra parameters are involved in the fine-tuning procedure or not, PEFT methods can be divided into two categories, namely: *Extra-parameter method* and *Intra-parameter method*. The extra-parameter method freeze all of the original parameters of an LLM and insert a set of learnable parameters to optimize the model input or model layers. This is similar to adapter tuning and prompt-engineering. In comparison, intra-parameter methods freeze most of the original parameters and only tune a small number of parameters such as LISA, LoRA, etc.

In 2020, the predominant parameter-efficient adaptation technique was *Adapter* [11] [8]. This method sequentially integrates two adaptation modules (one after attention and the other after the feed-forward layer) in each Transformer [23] [8]. While the adapter did reduce memory costs in some respects, such modification unwillingly leads to extra inference latency with an increase in the network's depth. Moreover, empirical observation shows that applying an adapter to LLMs might lead to training instability [2] [8]. Around the same time, a separate project introduced a hyper-parameter transfer strategy (HPT), demonstrating the practicality of transferring trained hyper-parameters across a model's width [24] [8]. Such an approach provides more credence to the rationale behind the extending networks but lacks comprehensive evidence or theory to explain them thoroughly while pointing out potential difficulties in actual adoption. However, the following emergence of LoRA dragged scholars' attention due to its well-rounded theory behind its highly simple idea.

When we do not have access to modify the model's architecture, using intra-parameter methods is optimal. LoRA is the most widely used intra-parameter due to its ability

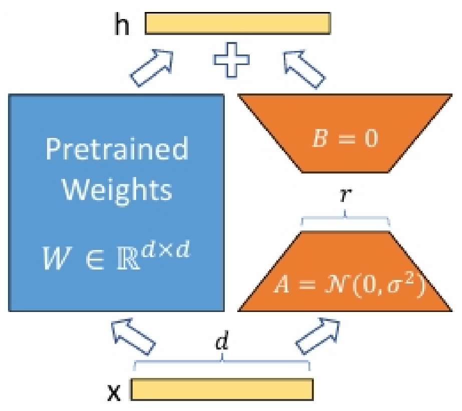

h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 1. Visual abstract of this survey's focus
Adapted from [25]

to adapt to downstream tasks and simple implementation requirements. Specifically, LoRA achieves parameter efficiency by updating the dense neural network layers of an LLM with pluggable low-rank matrices. These matrices are independent of the LLM and can be stored and reused in other related downstream tasks. Moreover, such plugin matrices can be combined to derived synthesize pattern learning to achieve cross-task generalization.

Noting that some previous surveys have mentioned LoRA and only introduce a small number of LoRA-related works, or only introduce a few benchmark testing for performance comparison among different LoRA variant. In this survey, we plan to give a comprehensive overview of the current progress on LoRA on: *Downstream adaptation*, *cross-task generalization*, and *variants with novel efficiency-improving methods that boost the computation efficiency of LoRA*. Moreover, this survey will introduce a few LoRA applications. Overall, this survey is planned to give a detailed review from background knowledge, current research trends, and technical insights for LoRA.

2. Background

The general idea of performing natural language processing consists large scale pretraining on general domain data and task-wise adaptation on specific domain. To nurture strong downstream task's solving ability, the novel full-parameter fine-tuning might leads to downsides on high computation costs and low transfer ability of model parameters. Hence, the low-dimensional intrinsic dimensionality hypothesis [1] presents that over-parameterized mdoels reside on a low intrinsic dimentison. This gives scholars a hint that we can somewhat achieve proper learning performance by only updating parameters related to instrinsic rank [18]. Using this hypothesis as a substrate, LoRA [25] proposed an idea to update dense parameter layers with low-rank matrices *disintegration*. This leads to enhancement on both memory and computational efficiency. This section will introduce basic concepts of the theory behind LoRA and argue its enhancement in training efficiency.

2.1. LoRA [25]

A neural network contains many dense layers (parameterized as $W_0 \in \mathbf{R}^{d\times k}$) which perform matrix multiplication during parameter updates. To adapt the network to such downstream tasks, the novel approach is to update all parameters with a gradient $\Delta W \in \mathbf{R}^{d\times k}$. Formally, we can write the formula for parameter updating as:

$$W = W_0 + \Delta W$$

This process is called full fine-tuning, which is time consuming and also computation and memory unfriendly. To make improvement, LoRA proposed that for a pre-trained weight matrix $W_0 \in \mathbf{R}^{d\times k}$, we constrain its update by representing the weight matrix with a low-rank decomposition:

$$W_0 + \Delta W = W_0 + BA$$

where $B \in \mathbf{R}^{d\times r}$, $A \in \mathbf{R}^{r\times k}$, $d, r, k \in \mathbf{N}$, and $rank\ r << \min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. For initialization, B and A are initialized with a random Gaussian distribution and zero respectively. In total the parameter number of the LoRA plugins is $r \times (d + k)$, which is significantly less the $d \times k$. Here we referenced a graph comparing *full fine-tuning* and *LoRA*.

LoRA is highly *parameter efficient* as it only updates a small subset of model parameters, enhancing computation and memory efficiency meanwhile maintaining the inference latency in a reasonable scope [9] [18]. Further enhancement on parameter efficiency contains extending low-rank matrix to low-rank tensor, and applying Kronecker decomposition [6] [10] [18], etc.

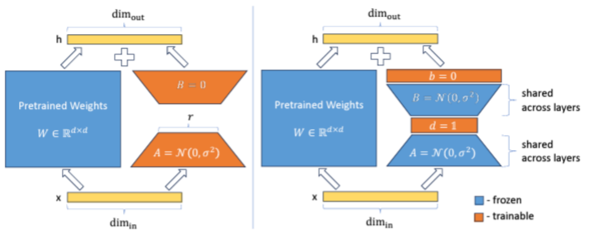
Furthermore, LoRA is also "Removable" and "independent" from the parameter's of the model after training, making the captured patterns in LoRA's parameters reusable by different users on different purposes. When we have multiple LoRA modules that are trained for multiple tasks, it is reasonable to combine these modules and expect a proper *cross-task generalization performance*.

In practice, for a Transformer-based LLM specifically, the dense layer typically refer to the **projection matrices** in attention head and **feed forward network** at the last. Typically, the original LoRA settings apply it to the query and value weight matrices in the attention model, and some following subsequent work showed that applying it to FFN layer can further improve model's performance in some respect [8].

2.2. LoRA variants

Though the initiation of LoRA brought researchers a general concept of PEFT, many aren't satisfied with the accuracy gap due to the limitation of matrics A and B's relative small sizes compare to most models' original weights. Recent variants of LoRA have aimed to further improve LoRA's generalization and efficiency. For instance:

VeRA: VeRA (Vector-based Random Matrix Adaption) [13] further reduced the parameter size of the LoRA adapters. VeRA borrows the concept of random projections, beliving that only a small fraction of weights in a large neural network is used to veer its behavior which lead to expected performance. By adding small vectors d and b, the matrix multiplication can lead to the same destination as training a few parameters in the entire matrix would.



dim_out
h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
x
dim_in
$b = 0$
$B = \mathcal{N}(0, \sigma^2)$
$d = 1$
$A = \mathcal{N}(0, \sigma^2)$
shared across layers
shared across layers
- frozen
- trainable

Figure 2. From LoRA to VeRA. A and B initialized with random weights and won't be trained. Vectors d and b is trained Adapted from [13]

LoRA-FA: LoRA-FA [26] (LoRA with Fozen-A), is literally what its name implies: it freezes matrix A. Hence, same concept of random projections is used here as well. LoRA-FA utilizes matrix A to serve as a random projection. Though, LoRA-FA introduces no additional vectors, suggesting training on matrix B. Therefore, consider original weight matrix has dimension $d \times k$ where $d = k$, freezing matrix A halves the number of parameters while obtaining comparable performance to normal LoRA [26].

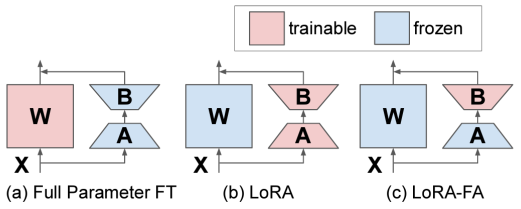


Figure 3. LoRA-FA freezes matrix A and only trains on matrix B Adapted from [26]

LoRA-drop: LoRA-drop [14] introduces an algorithm calculating the importance of each layer. This importance is exploited further for determining which layers can be enhanced using LoRA. Layers with less importance will not be injected additional LoRA layer. The algorithm of LoRA-drop hence provides flexibility in terms of using only a subset of LoRA layers, achieving both comparable accuracy as the original LoRA method and less training time.

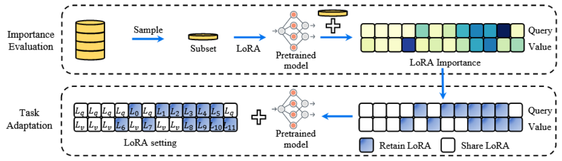
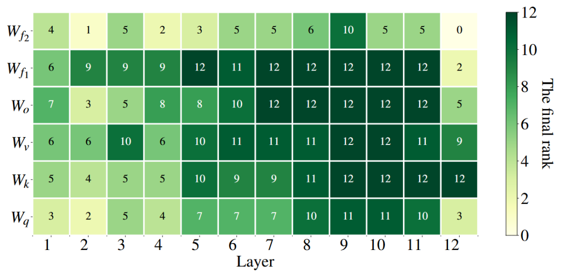


Figure 4. LoRA-drop algorithm overview [14]

LISA: LISA (Lyaerwise Importance Sampled AdamW) [21], similar to lora-drop, addresses memory efficiency issues when fine-tuning LLMs by selectively freezing most intermediate layers while updating only the important ones. What is different is that LISA is inspired by the skewed distribution oflayerwise weight norms observed in LoRA training. Pan et.al then perform layerwise sampling aimed to optimize the important layers. Pan et.al claimed LISA is able to retain or even surpass performance of both LoRA and full-parameter fine-tuning in a wild range of settings.

AdaLoRA: AdaLoRA (Adaptive LoRA) [27] proposes another way to determine importance of LoRA parameters. The authors suggested to consider the singular values of LoRA matrices as indicators of their importance. That is, by calculating the square roots of the eigenvalues of the matrices, we can obtain how much variance is captured by the rows of matrix each. AdaLoRA can thus decide which rows can be retained or omitted. It's different from LoRA-drop that it allows adapters with different ranks to be injected in various layers, whereas LoRA-drop can only choose either to inject LoRA layers (all with the same rank) or not to inject LoRA layer at all.



	1	2	3	4	5	6	7	8	9	10	11	12
W_{f_2}	4	1	5	2	3	5	5	6	10	5	5	0
W_{f_1}	6	9	9	9	12	11	12	12	12	12	12	2
W_o	7	3	5	8	8	10	12	12	12	12	12	5
W_v	6	6	10	6	10	11	11	11	12	12	11	9
W_k	5	4	5	5	10	9	9	11	12	12	12	12
W_q	3	2	5	4	7	7	7	10	11	11	10	3

Figure 5. LoRA matrics are given different ranks at different layers [27]

DoRA: DoRA (Decomposed Low-Rank Adaptation) [17], brought on the table by NVIDIA, starts with the idea that each matrix can be decomposed into a product of a magnitude and a direction. The authors, after decomposed update weight matrices (δD) from normal fine-tuning process and fine-tuning process using LoRA, found a change in direction and magnitude, suggesting a property mismatch between normal fine-tuning and fine-tuning using LoRA. They then proposed DoRA which separate the pretrained weight matrix W into magnitude vector m of size $1 \times d$ and a direction matrix V enhanced by $B \times A$. On several benchmarks, DoRA outperforms LoRA in accuracy. The authors attribute the results to the decomposition of weight updates into magnitude and direction, causing training similar to the training done in fine-tuning.

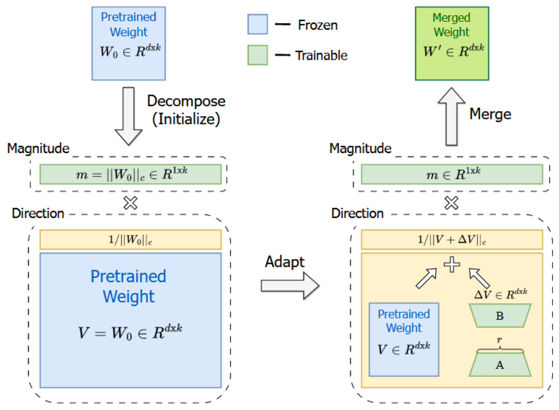

Pretrained Weight $W_0 \in R^{dxk}$
Frozen
Trainable
Merged Weight $W' \in R^{dxk}$
Decompose (Initialize)
Merge
Magnitude
$m = ||W_0||_c \in R^{1xk}$
$m \in R^{1xk}$
Direction
$1/||W_0||_c$
$1/||V + \Delta V||_c$
Pretrained Weight $V = W_0 \in R^{dxk}$
Adapt
$\Delta V \in R^{dxk}$
Pretrained Weight $V \in R^{dxk}$
B
r
A

Figure 6. weight matrix W is decomposed into magnitude m and direction V [17]

LISA: LISA (Lyaerwise Importance Sampled AdamW) [21], similar to lora-drop, addresses memory efficiency issues when fine-tuning LLMs by selectively freezing most intermediate layers while updating only the important ones. What is different is that LISA is inspired by the skewed distribution oflayerwise weight norms observed in LoRA training. Pan et.al then perform layerwise sampling aimed to optimize the important layers. Pan et.al claimed LISA is able to retain or even surpass performance of both LoRA and full-parameter fine-tuning in a wild range of settings.

While the above variants represent key improvements, it is worth noting that there are many other LoRA-based methods proposed to address similar challenges in parameter-efficient fine-tuning, each focusing on optimizing specific aspects such as memory usage, adaptability, or computational efficiency. These survey aims to select representative variants that aim to improve different aspect of the original LoRA approach and evaluate their performance compare to FT on SOTA Large Language Models.

3. Method

We evaluate the performance of various LoRA fine-tuning variants on the Llama3-8B model, specifically fine-tuning on the GSM8k [4] and GSM-Plus [16] datasets. The Llama3-8B model [19], part of the LLaMA family, has been chosen for its moderate size, striking a balance between being suitable for research experiments and sufficiently large to demonstrate significant task-specific adaptation capabilities. To optimize efficiency and performance, we use the quantized version of Llama3-8B, which reduces memory requirements and enhances computational speed during fine-tuning. The original LoRA serves as our baseline, against which we compare multiple LoRA variants, including VeRA, DoRA, LoRA-FA, LoRA-drop, and AdaLoRA. While maintaining uniform training setups for consistency, exceptions are made for variants like **AdaLoRA**, which auto-adjusts ranks for each LoRA layer. By analyzing these adaptations, we aim to understand their impact on the model's ability to handle diverse tasks while maintaining resource efficiency, and to explore architectural differences that contribute to variations in performance.

3.1. Model

In this study, we select LLaMA-3 to explore various types of LoRA fine-tuning. To optimize both efficiency and performance, we use the quantized version of LLaMA-3-8B, which reduces memory requirements and improves computational speed during the fine-tuning process.The original *LoRA* serves as our baseline against which we compare multiple LoRA variants. Specifically, we include *VeRA*, *DoRA*, *LoRA-FA*, *LoRA-drop*, and *AdaLoRA* in our comparisons to evaluate their respective strengths and adaptability. By analyzing these variants, we aim to understand how different adaptations to LoRA impact the model's ability to handle different mathematical reasoning tasks (reasoning, calculation, etc.) while maintaining resource efficiency, and potentially explore what difference in architecture leads to various performances.

3.2. Evaluation and Metrics

We use Llama3:8b model adapted from ollama to evaluate the correctness of our baseline and fine-tuned models' generated answers. We acknowledge the two major challenges [16] in evaluating LLM-generated answers. They either require costly human annotators to judge if the generated answer is correct, or require researchers to pay the price for api calling to use large models for evaluation purposes. Yet, new experiments [7, 15] suggest using SOTA small size LLM such as llama3-8bit may serve benchmarking purposes as emerging LLMs can also help giving annotations. Specifically, we ask Llama3:8b to compare ground truth solution with our model generated answers, and formed a chain of reasoning in judging if the generated answer is correct. It at the end of its response give a token that is either $< True >$ or $< False >$: does the answer match the solution or not.

To compare if there is really statistically important different between each lora variants, we followed the methodlogy used compute the confidence interval of accuracy / error using the formula:

$$\varepsilon = z\sqrt{\frac{a(1-a)}{n}}$$

(where ε represents the radius of the confidence interval, a is the measured accuracy or error, n is the total number of test samples and z is the number of standard deviations from the Gaussian distribution) For a 95% of confidence interval, we set $z = 1.96$. This binomial confidence interval quantifies the uncertainty in accuracy or error, ensuring robust comparisons.

These variants are further evaluated based on metrics such as **efficiency** and **trainable parameters** horizontally. This provides insights into how different LoRA adaptations impact task performance while maintaining resource efficiency.

3.3. Datasets

To comprehensively evaluate the performance of different LoRA variants in terms of mathematical reasoning and inference capabilities, we selected two datasets, GSM8k [4] and GSM-PLUS [16], as benchmarks for our experiments. GSM8k is a widely adopted dataset consisting of around **8k** grade-school-level math problems that cover fundamental arithmetic, basic geometry, and elementary algebra. It is commonly used to test the understanding and computational ability of language models for relatively straightforward mathematical tasks. However, evaluating models solely on such foundational problems may not sufficiently reveal their performance in advanced logical reasoning, complex inference chains (Chain of Thought, etc.), and multi-step decision-making. To address this limitation, GSM-PLUS [16] "extends" GSM8k [4] by introducing significantly more challenging problems with higher complexity of reasoning, longer solution paths and questions requiring intricate logical breakdowns in multiple steps. In total, GSM-PLUS [16] contains more refined and complex problems around **13k** with no potential overallap with GSM8k [4]. By using both datasets, we aim to assess not only the fundamental mathematical capabilities of the models but also their robustness and generalization when confronted with sophisticated reasoning tasks, seeking if there will be expected *"intruder dimension"* (large singular values that are almost orthogonal to the singular vectors of parameters for a fine-tuned model) [22] behavior happen that cause unwilling performance on the modal powered by catastrophic forgetting. This comprehensive evaluation provides critical insights to guide further optimization and enhancement of LoRA variants.

4. Experiments Setup

4.1. data preparation:

Both datasets share similar data split that separates samples into train and test set. Following this intuitive pattern, we fine-tuned our model using the train split and performed evaluation on the test split. For **GSM8k**, each sample contains a $question$ and a $answer$ texts, where $question$ is the math question prompt and $answer$ is the ground truth solution. For **GSM-plus**, though each sample contains more columns, we only used $question$ and $solution$ (similar to GSM8k's $answer$) column for training and evaluation.

4.2. hyperparameter setup:

For the fine-tuning of LLaMA 3-8B Instruct with various LoRA variants, we set the rank to 8 to balance performance and memory usage based on our device capacity. For AdaLoRA, we initialized the rank at 128 and gradually decayed it to 8 during training, leveraging its adaptive rank reduction strategy.

4.3. Dataset size:

The datasets used for fine-tuning were **GSM8k** and **GSM-Plus**. Following the scaling laws suggested in **Kaplan et al.** (2020) [12], the number of training samples should be aligned with the model's parameter count. According to the scaling law $D \propto N^{0.74}$, for large models like LLaMA 3.1, the dataset size should increase proportionally to avoid overfitting while ensuring that the model can generalize effectively. The selected training set sizes of 5,603 and 7,908 examples for GSM8k and GSM-Plus respectively are justified by several considerations:

- **Computational Efficiency:** The evaluation process involves using Large Language Models (LLMs) to assess generated answers' correctness, which is computationally intensive and time-consuming. It is practical and necessary to find a balance between dataset size and computational efficiency.

- **Statistically Significant:** We randomly selected 500 samples from the test set for evaluation with a confidence level of 95% and margin of error of approximately 4.4%, which is within acceptable ranges for machine learning research.

5. Result

Our main result is recorded in Table 1.

On the **GSM8k** mathematical reasoning task, **DoRA** achieved the highest accuracy of 80.80% (95% CI: 77.35-84.25%), marginally outperforming the **zero-shot** baseline at 80.60% (95% CI: 77.13-84.07%). The standard **LoRA** and **LoRA-drop** showed comparable performance at 78.40% and 79.00% respectively, two confidence interval overlap, suggesting no statistically significant difference between these variants. However, **AdaLoRA** and **LoRA-FA** demonstrated substantially lower performance, achieving only 43.00% and 38.40% accuracy respectively, indicating these adaptations may not be well-suited for mathematical reasoning tasks.

For the **GSM-Plus** logical reasoning task, we observe generally lower performance across all variants, suggesting this dataset does contain more challenging problems. **DoRA** again emerged as the top performer with 60.60% accuracy (95% CI: 56.31-64.88%), showing a clear advantage over other variants. Interestingly, **LoRA-FA** performed relatively better on this task at 59.60%, contrasting sharply with its poor performance on **GSM8k**. The standard **LoRA** and **LoRA-drop** showed similar performance levels (57.60% and 58.40% respectively), maintaining their consistent behavior across both tasks. Several key findings emerge from these results:

1. The consistent superior performance of **DoRA** across both tasks suggests its effectiveness in preserving and adapting the model's reasoning capabilities, while requiring relatively few parameters (0.8540B).

2. The significant performance gap between mathematical (**GSM8k**) and logical reasoning (**GSM-Plus**) tasks indicates that logical reasoning problems may require different or more sophisticated fine-tuning approaches.

3. The narrower confidence intervals and width metrics for **GSM8k** compared to **GSM-Plus** suggest more consistent model performance on mathematical reasoning tasks.

6. Discussion

Our experimental results reveal significant insights into the performance characteristics and architectural implications of various **LoRA** variant's fine-tuning approaches across both **GSM8k** [4] and **GSM-Plus** [16] datasets. The findings demonstrate a complex interplay between model architecture, parameter efficiency, and task complexity, with notable implications for future development of parameter-efficient fine-tuning methods.

The most striking observation from our experiments is the consistent superior performance of **DoRA** across both datasets, achieving **80.80%** accuracy (CI: [**77.35%, 84.25%**]) on GSM8k and **60.60%** (CI: [**56.31%, 64.88%**]) on GSM-Plus. This exceptional performance can be attributed to DoRA's innovative approach to parameter updates through *direction vectors* rather than traditional weight matrices. The direction-based methodology enables more precise control over the geometric relationships between parameter updates while maintaining the pre-trained model's knowledge structure [17]. This architectural choice appears particularly beneficial for mathematical reasoning tasks, where preserving relationships between concepts is crucial for accurate problem-solving.

Traditional **LoRA** implementations and their variants demonstrate interesting performance characteristics that warrant careful analysis. The baseline LoRA achieves **78.40%** accuracy on GSM8k and **57.60%** on GSM-Plus, closely tracking zero-shot performance. This suggests that the low-rank adaptation [25] effectively captures task-specific information while preserving the original model's mathematical reasoning capabilities. The addition of dropout to the base LoRA architecture shows minimal improvement (**79.00%** on GSM8k, **58.40%** on GSM-Plus), with confidence intervals significantly overlapping the base implementation. This marginal enhancement suggests that LoRA's inherent low-rank structure might already provide sufficient regularization, making additional dropout-based regularization [14] less impactful.

The poor performance of **AdaLoRA** (43.00%) can be explained through a more technical lens of singular value decomposition (SVD) and intruder dimensions [22]. When AdaLoRA adaptively reduces its rank from 128 to 8 (0.067B to 0.004B parameters), the process can be analyzed through the SVD of the weight updates $\Delta W = AB^T$, where A and B are the LoRA matrices. The aggressive rank reduction creates new singular vectors $\{v_i\}$ that have large singular values σ_i but are nearly orthogonal to the principal components of the original fine-tuned model's weight space, i.e., $\cos(\theta) \approx 0$ between these vectors and the original model's key directions. These intruder dimensions dominate the model's behavior due to their large singular values, yet their orthogonality means they fail to preserve the pre-trained mathematical knowledge encoded in the original weight space [22]. This is particularly problematic because these intruder dimensions effectively create a subspace that is disconnected from the pre-trained model's knowledge manifold, leading to catastrophic interference with the model's ability to access its pre-training distribution. The dramatic reduction in parameters exacerbates this effect by limiting the model's capacity to maintain bridges between the new task-specific knowledge and the original pre-trained mathematical reasoning capabilities.

The strong zero-shot performance of the base model (**80.60%** on GSM8k, **59.80%** on GSM-Plus) provides valuable context for interpreting our fine-tuning results. The high baseline performance suggests robust pre-trained

Table 1. Performance Comparison of Parameter-Efficient Fine-tuning Methods on Mathematical and Logical Reasoning Tasks

Model Variant	Params[1]	$\uparrow$ Acc (%)	95% CI	$\downarrow$ Width[2]
GSM8k Mathematical Reasoning Task				
DoRA[3]	0.8540	**80.80**	**(77.35, 84.25)**	**0.0690**
LoRA[3]	1.7040	78.40	(74.79, 82.01)	0.0721
LoRA-drop[3]	1.7040	79.00	(75.43, 82.57)	0.0714
AdaLoRA[3]	0.0042[4]	43.00	(38.66, 47.34)	0.1189
LoRA-FA[3]	0.0026	38.40	(34.14, 42.66)	0.0853
Zero-shot	8.030	80.60	(77.13, 84.07)	0.0693
GSM-Plus Logical Reasoning Task				
DoRA[3]	0.8540	**60.60**	**(56.31, 64.88)**	**0.0857**
LoRA[3]	1.7040	57.60	(53.26, 61.93)	0.0866
LoRA-drop[3]	1.7040	58.40	(54.08, 62.72)	0.0864
AdaLoRA[3]	0.0042[4]	46.80	(42.42, 51.17)	0.0875
LoRA-FA[3]	0.0026	59.60	(55.29, 63.90)	0.0860
Zero-shot	8.030	59.80	(55.50, 64.09)	0.0895

[1] Parameters shown in billions ($\times 10^9$)
[2] Radius calculated for Confidence Interval from the Given Formula.
[3] Models fine-tuned with different LoRA variant as update schema for weight.
[4] For Adalora, we only report the trianable parameters after **rank reduction** in the table (Initial trainable parameter: 0.067125248×10^9)
Note: GSM8k evaluates mathematical reasoning capabilities, while GSM-Plus tests logical reasoning and bias quotient. Base LLM refers to the original pre-trained language model without fine-tuning.

mathematical abilities in model llama3 8-bit (instruct) [19], while the consistent performance gap between datasets (approximately **20%**) indicates inherent limitations in complex reasoning capabilities. The fact that DoRA matches or exceeds zero-shot performance with fewer parameters (**0.8540M** vs **8.030M**) demonstrates effective knowledge transfer during fine-tuning, while the performance degradation in lightweight variants reinforces the existence of a minimum parameter threshold for effective adaptation.

6.1. Architectural Implications

These findings suggest several key principles for the design of LoRA variants. The success of DoRA's direction-based approach indicates the importance of geometric preservation [19] in parameter updates, suggesting that future architectures should prioritize maintaining geometric relationships in parameter space. Our results also point to the existence of critical parameter thresholds (approximately **0.8-1.7M**) for maintaining performance, highlighting the need for careful balance between parameter efficiency and task complexity. Furthermore, the varying effectiveness of different architectural choices across task types emphasizes the importance of task-specific adaptation mechanisms.

6.2. Limitations and Future Work

Several limitations of our study warrant consideration and suggest directions for future research. The use of **Llama3-8b** [19] for validation, while practical, might introduce systematic biases in our evaluation methodology. The binary correctness metric may not capture partial understanding or reasoning capabilities, and the confidence intervals in our results suggest the need for larger test sets. Future work should explore:

- Hybrid architectures combining DoRA's directional learning with AdaLoRA's adaptive rank mechanism
- Investigation of frequency-aware directional updates
- Development of task-specific downstream adaptation mechanisms

7. Conclusion

In this work, we conducted a comprehensive empirical investigation of various LoRA variants' performance on mathematical and logical reasoning tasks using the Llama3-8B model. Our experiments reveal that direction-based parameter updates, as implemented in DoRA, consistently outperform other variants, achieving 80.80% accuracy on GSM8k and 60.60% on GSM-Plus, compared to baseline LoRA's 78.40% and 57.60% respectively. This superior performance can be attributed to DoRA's preservation of geometric relationships in parameter space during fine-tuning, which proves crucial for maintaining pretrained mathematical knowledge while adapting to specific tasks. Our analysis further identifies a critical parameter threshold (approximately 0.8-1.7M parameters) below which performance significantly degrades, as evidenced by the performance patterns of ultra-lightweight variants like AdaLoRA and LoRA-FA. However, the performance recovery of LoRA-FA on complex reasoning tasks (59.60% on GSM-Plus) suggests that optimal architectural choices may be task-dependent. These findings, combined with the strong zero-shot performance of the base model, indicate that future development of parameter-efficient fine-tuning methods should focus on task-specific adaptation mechanisms while maintaining geometric relationships in parameter space, potentially through hybrid approaches that combine the strengths of different variants.

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 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*, pages 7319–7328, 2021. 2

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[3] Y Chen, S Qian, H Tang, X Lai, Z Liu, S Han, and J Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023. 1

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021. 4, 5, 6

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, May 2019. 1

[6] Ali Edalati, Mohammad S. Tahaei, Ivan Kobyzev, Vahid Partovi Nia, John J. Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022. 2

[7] Naghmeh Farzi and Laura Dietz. Best in tau@llmjudge: Criteria-based relevance evaluation with llama3, 2024. 4

[8] Vlad Fomenko, Han Yu, Jongho Lee, Stanley Hsieh, and Weizhu Chen. A note on lora. *arXiv preprint arXiv:2404.05086*, April 2024. 1, 2

[9] Vlad Fomenko, Han Yu, Jongho Lee, Stanley Hsieh, and Weizhu Chen. A note on lora. *arXiv preprint arXiv:2404.05086*, 2024. 2

[10] Xiaozhong He, Chen Li, Peidong Zhang, Jian Yang, and Xiaoguang E. Wang. Parameter efficient model adaptation for vision transformers. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 817–825, 2023. 2

[11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter efficient transfer learning for nlp. 2019. Unpublished manuscript. 1

[12] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv.org*, January 23 2020. 5

[13] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation. *arXiv*, 2023. 3

[14] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Hongyun zhou, xiangyu lu, wang xu, conghui zhu, tiejun zhao, muyun yang. *arXiv*, 2024. 3, 6

[15] Charles Koutcheme, Nicola Dainese, Arto Hellas, Sami Sarsa, Juho Leinonen, Syed Ashraf, and Paul Denny. Evaluating language models for generating and judging programming feedback, 2024. 4

[16] Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*, 2024. 4, 5, 6

[17] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. 3, 4, 6

[18] Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey on lora of large language models. *arXiv preprint arXiv:2407:11046v3*, 2024. 2

[19] Abhinav Jauhri Abhinav Pandey et.al Meta, Abhimanyu Dubey. The llama 3 herd of models. *arXiv*, 2024. 4, 7

[20] OpenAI, Josh Achiam, Steven Adler, and ... Gpt-4 technical report. 2024. 1

[21] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*, 2024. 3, 4

[22] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. 2024. 5, 6

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia

Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. 1

[24] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy S. Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc., 2021. 1

[25] Y. Zeng and K. Lee. The expressive power of low-rank adaptation. *arXiv.org*, March 18 2024. 2, 6

[26] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv*, 2023. 3

[27] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv*, 2023. 3

Appendix

0.1 Testing Prompt Usage*

Here is the curated prompt we feed in to let llama3 8-bit to do ground truth and generated answer comparison.

```
prompt = f'''
    You are given a predicted answer and a ground truth solution,
    determinant whether the predicted answer match the ground truth answer.
    End your response with <True> or <False> predicted answer:
    {generated_answer}
    ground truth solution:
    {answer_part}
'''
```

0.2 Confidence Interval Chart*

For better visibility, we plot the *Confidence Interval chart* for two tested datasets among different LoRA variants side by side horizontally in the next page.

Acknowledgment

We are giving our special thanks to Professor Kanan who's our supervisor for *CSC277 End-to-End Deep Learning* course. Without his experienced advisory and guidance, we would not come up with ideas to make our comparison using statistical methods. His feedback has led us through great reflection and revision resulted in our final work.

Team Contribution

- **Henry:** Conducted experiments on **Lora-FA**. written *Introduction*, *Background*, *Background*, *Discussion*, and *Conclusion* section and all novel charts included in the report.
- **Kevin:** Conducted experiments on **LoRA**, **LoRA-drop**, and **DoRA**. Written *Experiment Setup*
- **Tianyi:** Conducted experiments on **AdaLoRA**. Written *LoRA variants*, *Background*, *Datasets*, and *Results* sections. Revised *Experiment Setup*.

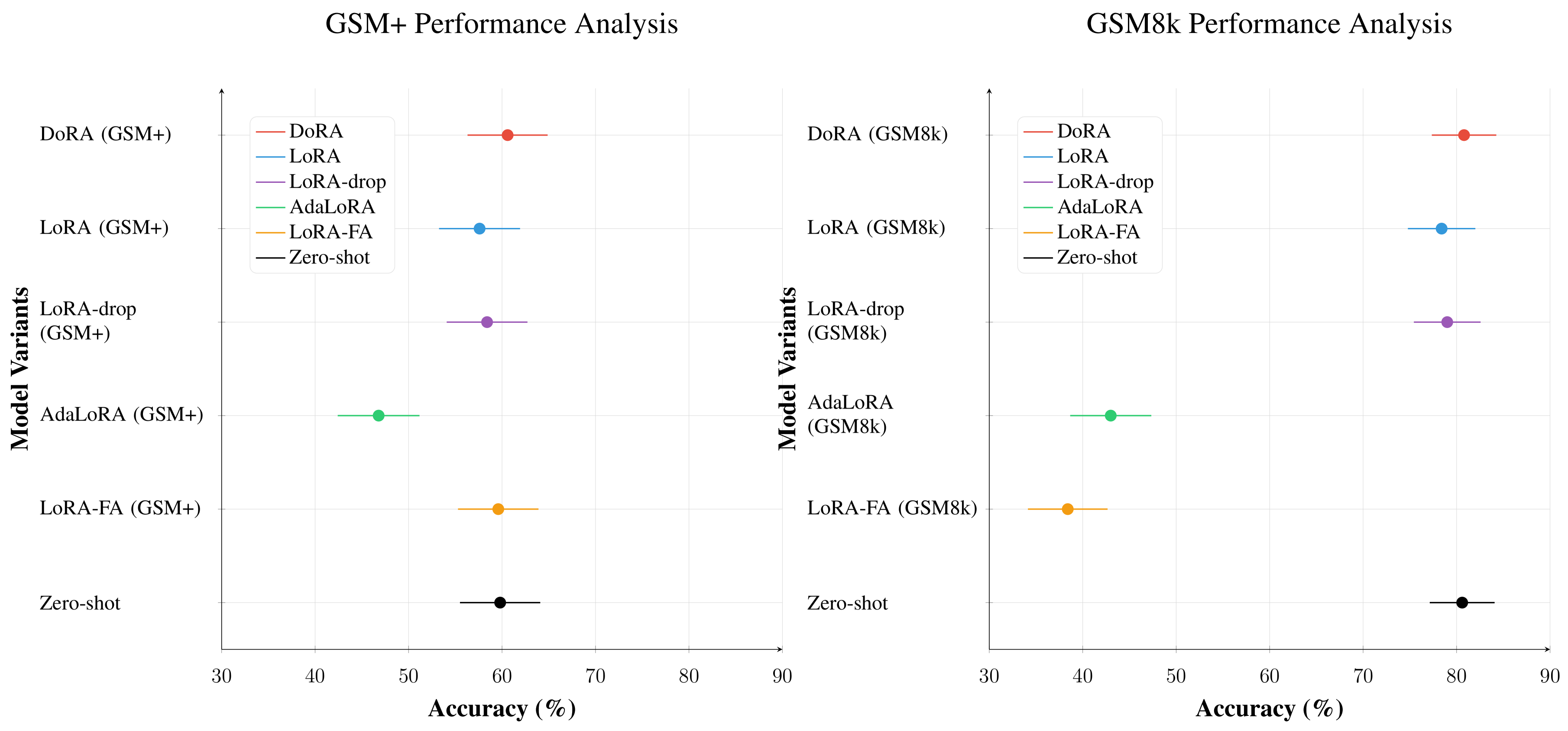


Figure 1: **Comparative Analysis of Parameter-Efficient Fine-tuning Methods:** Performance evaluation on GSM+ (left) and GSM8k (right) benchmarks. Points indicate mean accuracy scores with horizontal lines showing 95% confidence intervals. On both benchmarks