TableLoRA: Low-rank Adaptation on Table Structure Understanding for Large Language Models

Xinyi He[1]* **Yihao Liu**[2]* **Mengyu Zhou**[3]† **Yeye He**[3]
Haoyu Dong[3] **Shi Han**[3] **Zejian Yuan**[1] **Dongmei Zhang**[3]
[1] State Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2] Peking University [3] Microsoft
hxyhxy@stu.xjtu.edu.cn, haoeliu@stu.pku.edu.cn, yuan.ze.jian@xjtu.edu.cn,
{mezho, yeyehe, haoyu.dong, shihan, dongmeiz}@microsoft.com

Abstract

Tabular data are crucial in many fields and their understanding by large language models (LLMs) under high parameter efficiency paradigm is important. However, directly applying parameter-efficient fine-tuning (PEFT) techniques to tabular tasks presents significant challenges, particularly in terms of better table serialization and the representation of two-dimensional structured information within a one-dimensional sequence. To address this, we propose TableLoRA, a module designed to improve LLMs' understanding of table structure during PEFT. It incorporates special tokens for serializing tables with special token encoder and uses 2D LoRA to encode low-rank information on cell positions. Experiments on four tabular-related datasets demonstrate that TableLoRA consistently outperforms vanilla LoRA and surpasses various table encoding methods tested in control experiments. These findings reveal that TableLoRA, as a table-specific LoRA, enhances the ability of LLMs to process tabular data effectively, especially in low-parameter settings, demonstrating its potential as a robust solution for handling table-related tasks.

1 Introduction

Tabular data are widely used in numerous fields, and LLMs are also widely applied to the understanding and processing of tabular data, such as Table-GPT (Li et al., 2024b), TableLLM (Zhang et al., 2024), *etc.*. Meanwhile, the PEFT (Mangrulkar et al., 2022) (Parameter-Efficient Fine-Tuning) paradigm, with its advantages of high parameter efficiency, is widely used to fine-tune LLMs. Consequently, exploring methods to learn improved table representations under a high parameter efficiency paradigm to address table-related issues more effectively remains a critical and valuable area of research.

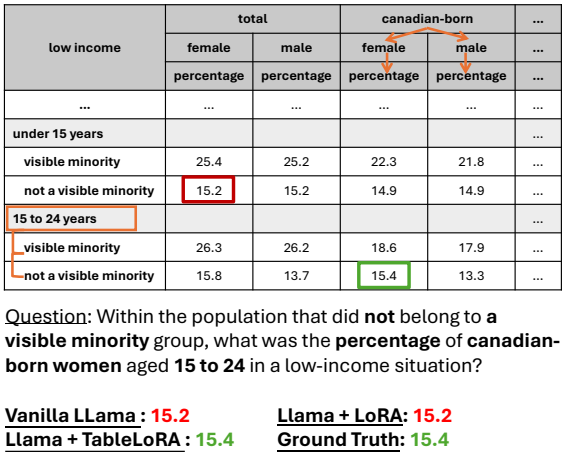

Figure 1: An Example of Common Errors when Directly Using LoRA to Finetune Llama on Tabular Tasks.

However, directly applying PEFT(*e.g.*, Low-rank Adaptation(LoRA)) to table-related tasks reveals several critical challenges:

Challenge 1: How to better serialize tables. Previous studies have shown that different methods of table serialization affect the results (Sui et al., 2024a). However, even with existing serialization techniques, models still struggle to accurately recognize table structures. *E.g.*, in the TableQA example shown in Figure 1, it needs to retrieve cell from the table. The column for retrieval can be easily identified through the same header names in the query. However, the Llama model fine-tuned with LoRA fails to recognize the cell in the same column as the header.

Challenge 2: How to better represent two-dimensional structured information in a one-dimensional sequence. The positional information of rows and columns in a table is crucial for understanding the table structure, row-column correspondence, and so on. *E.g.*, to comprehend the hierarchical left header in Figure 1, one must rec-

* The contributions by Xinyi He and Yihao Liu have been conducted and completed during their internships at Microsoft.

† Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22376–22391

ognize that cells like "15 to 24 years" are in the leftmost column and identify their corresponding row and column content. However, during LoRA fine-tuning, positional information is not explicitly learned and is only implicitly computed through attention, leading to the query in Figure 1 being unable to accurately locate the rows to retrieve through the hierarchical structure of the left header.

To promote the recognition and understanding of table structures during tabular tasks and enhance the ability of Large Language Models (LLMs) to process tables within PEFT, we propose TableLoRA[1], a module compatible with the PEFT framework for LLMs. Existing tabular LLMs attempt to learn table structure relationships using attention mechanisms through additional training. In contrast, we directly inform the model of these relationships through our design, which consists of two key components:

To address *Challenge 1*, the **Special Tokens Encoder** enhances tabular data representation during fine-tuning. The challenge lies in effectively learning and incorporating the special tokens `[tab]`, `[row]`, and `[cell]`, which replace traditional markdown or HTML formats and provide a structured tabular representation for improved model processing. We use a fine-tuning method inspired by p-tuning to ensure effective gradient propagation to special token embeddings, enhancing the model's ability to understand and manipulate tabular data.

To address *Challenge 2*, **2D LoRA** is designed to address the limited information derived from the two-dimensional cell positions compared to the rich semantics each token conveys. To tackle this, we encode row and column indices using low-rank embeddings and upscale these to integrate with the Large Language Model's (LLM) token embeddings. This approach provides precise row and column index identifiers, enabling the LLM to infer whether two cells align along the same row or column. The importance of this structural awareness cannot be overstated for tasks that require the comprehension of tabular data. The 2D LoRA operates in parallel with the original LoRA framework for each layer, enhancing the LLM's ability to effectively incorporate structural information and to generate content based on structured tabular data.

We conducted experiments on three models across four datasets that encompass QA and fact verification tasks on the tables. The results indicate that TableLoRA consistently demonstrates improvements over vanilla LoRA. Specifically, it achieves a 5.9% improvement in HiTab. Furthermore, TableLoRA mitigates 40.56% of the performance gap between LoRA and full fine-tuning. For further validation, we conducted control experiments contrasting various table representation learning methods, highlighting the advantages of TableLoRA's structural design. Finally, we designed and executed a series of exploratory analytical experiments to unravel the principles underlying TableLoRA's efficacy.

The main contributions are as follows:

- TableLoRA is the first to propose a table-specific LoRA, which aids in learning structured information from tables by modifying the model architecture. This innovative approach enhances the model's ability to understand and process tabular data effectively.
- Under PEFT low-parameter settings, TableLoRA demonstrates a superior capability to capture and learn table structures. This makes it highly efficient and effective in scenarios where computational resources and parameter budgets are limited.
- The experimental results of TableLoRA are impressive. It consistently outperforms vanilla LoRA, showing significant improvements in various datasets, thus validating its efficacy and robustness in handling table-related tasks.
- We summarized and designed various methods for table representation learning, and, through controlled experiments, we proved that the current TableLoRA is the optimal solution among them.

2 Related Work

2.1 Tabular Task

Recently, table-related tasks have received considerable attention in the field. Various tasks and datasets have been proposed, such as TableQA (WikiTQ (Pasupat and Liang, 2015), FeTaQA (Nan et al., 2022), HiTab (Cheng et al., 2022), HybridQA (Chen et al., 2020c)), Text2SQL (WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), BIRD (Li et al., 2024a)), Fact Verification (TabFact (Chen et al., 2020b)), and tabular analysis (Table2Analysis (Zhou et al., 2020), Table2Charts (Zhou et al., 2021), AnaMeta (He et al., 2023), DS-1000 (Lai et al., 2022), Text2Analysis (He et al., 2024a)). These datasets

[1]The code will be open-sourced on https://github.com/microsoft/TableLoRA.

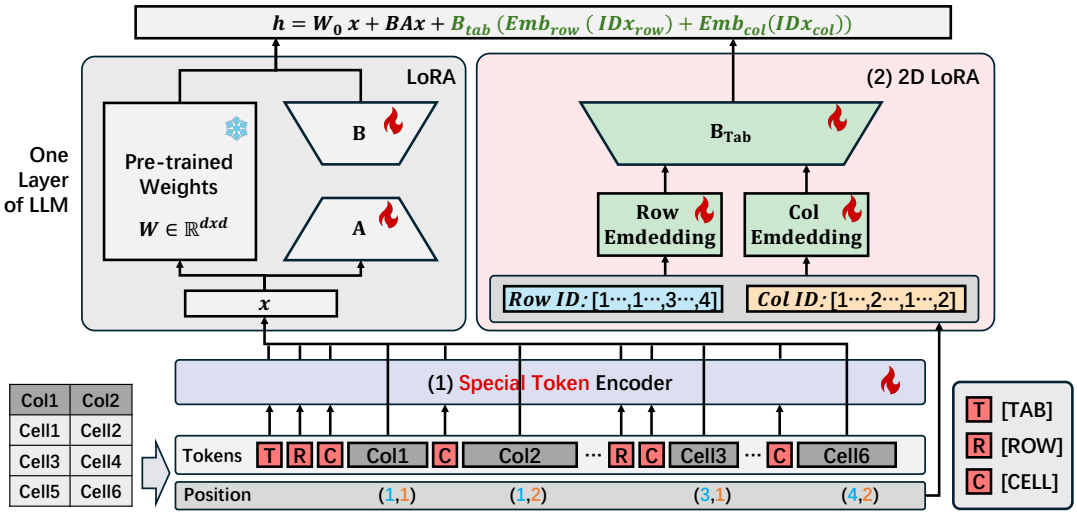

Figure 2: The Framework of TableLoRA. It consists of two main components: Special Tokens Encoder and 2D LoRA. (1) The Special Tokens Encoder (§3.1) incorporates specially defined tokens into the model's input alongside word embeddings just before the transformer layers. (2) The 2D LoRA (§3.2) embeds row and column indices and integrates them into the model at each layer, enabling the processing of tabular data's structure and content.

include various types of tables, such as database tables (*e.g.*, BIRD (Li et al., 2024a), Spider (Yu et al., 2018)), simple tables (tables with the first row as a header and several subsequent rows containing corresponding values, *e.g.*, WikiTQ (Pasupat and Liang, 2015)), and hierarchical tables (tables with tree-structured hierarchy in the top header or the left header, *e.g.*, HiTab (Cheng et al., 2022)). Their common feature is that the input includes tables, requiring models to have the capability to understand and analyze the tables, such as understanding row-column correspondences and positional information within the tables.

2.2 Tabular Representation Learning

In the era preceding Large Language Models, table representation methodologies can be divided into model designs and extra training techniques (Dong et al., 2022). Within model designs, various approaches are employed to effectively capture the structural details of tabular data: (1) Table Serialization: This method linearizes tables for better integration with transformer-based models, *e.g.*, TaPEx (Liu et al., 2021), TABBIE (Iida et al., 2021). (2) Structural Positional Encoding: This technique encodes structural information, such as row and column indices, to preserve spatial relationships within tables *e.g.*, TaPas (Herzig et al., 2020), MATE (Eisenschlos et al., 2021), TABBIE (Iida et al., 2021), and TUTA (Wang et al., 2021). (3) Structure-Based Attention Mechanisms: These mechanisms incorporate structural information into the model through attention, enhancing the focus on relevant table components, *e.g.*, TURL (Deng et al., 2020) and TUTA (Wang et al., 2021). (4) Multiple Encoder Frameworks: This approach uses multiple encoders to process table data more comprehensively, *e.g.*, TABBIE (Iida et al., 2021), DoT (Krichene et al., 2021), and KGPT (Chen et al., 2020a). For the design of our control experiments in §4.3.1, we draw inspiration from these model design strategies. However, it has been observed that directly applying these methods to the era of LLMs does not bring significant improvements.

2.3 Tabular Large Language Model

With the advent of the era of LLMs, a series of works related to large models for tables have followed one after another. To enhance the performance of LLMs in table tasks, existing work primarily focuses on improvements from the following perspectives: (1) Input sequence level: employing different serialization or augmentation methods, *e.g.*, TAP4LLM (Sui et al., 2024b), SpreadsheetLLM (Dong et al., 2024), CoCoST (He et al., 2024b); (2) Reasoning level: increasing task accuracy by breaking down problems into multiple steps to form a reasoning chain, *e.g.*, Chain-of-Table (Wang et al., 2024); (3) Training strategy level: enhancing training effects by constructing large-scale training datasets and synthesizing data, *e.g.*, TableLlama (Zhang et al., 2023), TableLLM (Zhang et al., 2024), StructLM (Zhuang

et al., 2024), Table-GPT (Li et al., 2024b). However, in scenarios with limited data and computational resources, there is a lack of relevant exploration on how to more efficiently improve the model's capabilities in table tasks. Moreover, in the PEFT setting, merely modifying the input sequence does not sufficiently enable the model to learn the structural information of tables. Therefore, in TableLoRA, we have made improvements specifically for tables through model design.

2.4 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) methods efficiently adapt large pretrained models to various downstream tasks by fine-tuning only a small subset of additional parameters, significantly reducing computational and storage costs while maintaining performance. PEFT encompasses techniques like soft prompt methods (*e.g.*, Prompt Tuning (Lester et al., 2021), P-tuning (Liu et al., 2024)), which optimize specific task parameters by adding learnable prompts to the input embeddings. Another approach is low-rank adaptation (*e.g.*, LoRA (Hu et al., 2022)), which inserts smaller trainable matrices into the model. Despite their advantages, these methods lack the ability to understand table structures. TableLoRA addresses this and improve performance on tasks involving structured data.

3 Methodology

Tabular tasks involve generating an answer sequence $output$ given a table T and related text $text$ (such as questions, table captions, etc.). The table consists of n columns and m rows of cells, $T = \{v_{0,0}, v_{0,1}, \ldots, v_{m,n}\}$.

TableLoRA is divided into two primary components as shown in Figure 2: Special Tokens Encoder and 2D Low-Rank Adaptation (2D LoRA). The Special Tokens Encoder is designed to enhance the model's understanding of tabular data by incorporating specially defined tokens that provide a clear and structured representation of tables. These special token encoder is added to the model's input at the same stage as the word embeddings, just before the transformer layers. On the other hand, 2D LoRA is aimed at encoding the structural information of tables by embedding row and column indices and integrating them into the Large Language Model at each layer. This dual approach ensures that the model can effectively process both the content and the structure of tabular data, leading to improved performance in related tasks.

3.1 Special Tokens Encoder

In special tokens encoder, we introduce the incorporation of special tokens to enhance the representation of tabular data during the fine-tuning process.

We define three special tokens: `[tab]`, `[row]`, and `[cell]`, which are used to replace traditional markdown or HTML formatting. Before entering table T and text $text$ into the model, the table needs to be serialized and concatenated with the text to form a single input sequence $input$. During the serialization process, these special tokens are used as delimiters: `[tab]` signifies the beginning of a table, `[row]` indicates the start of a new row, and `[cell]` marks the beginning of a new cell within a row, as shown in Equation (1). These tokens are designed to provide a structured and clear representation of tabular data, facilitating more effective processing and comprehension by the model.

$$\begin{cases} input = concate(table\_string, text) \\ table\_string \\ \quad = serialize(T) \\ \quad = \texttt{[TAB]}\ \texttt{[ROW]}\ \texttt{[CELL]}\ v_{0,0}\ \texttt{[CELL]}\ v_{0,1}\ \ldots \\ \quad\ \ \texttt{[ROW]}\ \texttt{[CELL]}\ v_{1,0}\ \ldots\ \texttt{[CELL]}\ v_{m,n} \end{cases} \tag{1}$$

To ensure effective learning of special token embeddings, we designed a special token encoder inspired by soft prompt methods (Qin and Eisner, 2021), which optimize task-specific parameters by adding learnable prompts to input embeddings. Both approaches require adding new learnable embeddings to the pre-trained model, but the key difference is that soft prompt methods add prompts only at the sequence's beginning, while in TableLoRA, special tokens appear at multiple positions throughout the sequence. Thus, we adopt the encoder from the soft prompt method and extend it to create a position-flexible special token encoder. The formula is as follows:

$$word\_embedding_i = \begin{cases} WordEmbedding(t_i), & t_i \notin S \\ SpecialTokenEncoder(t_i), & t_i \in S \end{cases} \tag{2}$$

where, $t_i \in input$ is one token of input, $S = \{\texttt{[TAB]}, \texttt{[ROW]}, \texttt{[CELL]}\}$ is the set of special token. Each $word\_embedding_i$ is concatenated to form a word embedding sequence that serves as input to the model.

In this paper, the encoder from P-tuning (Liu et al., 2024) is chosen as the special token encoder, which consists of a word embedding layer and a

linear layer. Furthermore, we compare the effects of different soft prompt methods in controlled experiments (§4.3.2).

3.2 2D Low-Rank Adaptation (2D LoRA)

Compared to the rich semantics conveyed by each token, the information that can be derived from the 2D cell positions is relatively limited. To address this, we add index embeddings for the row and column indices. Since the information density of these indices is relatively low, we choose to represent them using low-rank embeddings. These low-rank embeddings are then upscaled and integrated with the token embeddings of the LLM. This approach, termed 2D LoRA, operates in parallel with the original LoRA (Hu et al., 2022) framework for each layer, enabling the LLM to incorporate structural information effectively. Specifically, precise row and column index identifiers are provided, allowing the LLM to infer whether two cells are aligned along the same row or column. This structural awareness is critical for tasks that require an understanding of tabular data.

The mathematical formulation of this integration can be expressed as follows:

$$\begin{aligned} h &= W_0 x + BAx \\ &\quad + B_{\text{tab}} \left(\text{Emb}_{\text{row}}(\text{IDx}_{\text{row}}) + \text{Emb}_{\text{col}}(\text{IDx}_{\text{col}}) \right) \end{aligned} \tag{3}$$

where, h represents the hidden states. W_0, B and A are the weight matrix of the LLM and the original LoRA. B_{tab} is additional parameters introduced by the 2D LoRA. Emb_{row} and Emb_{col} denote the embeddings of the row and column indices, respectively. IDx represents which row/column this token belongs to. Specifically, for tokens that are not part of the table, both the IDx_{col} and IDx_{row} are set to 0. For special tokens, the token `[TAB]` has both IDx_{col} and IDx_{row} set to 0, the token `[ROW]` has IDx_{col} set to 0 and IDx_{row} set to the corresponding row number, and the token `[CELL]` has both IDx_{col} and IDx_{row} corresponding to its respective cell value.

The origin of 2D LoRA lies in the addition of column ID and row ID embeddings to word embeddings, similar to Tapas (Herzig et al., 2020). The uniqueness of 2D LoRA lies in its specific embeddings, which are distinct for each transformer layer. This differentiation is crucial as layers vary in depth and informational needs, as will be demonstrated in subsequent experiments (see Figure 3), showing varying impacts across layers. Additionally, our use of low-dimensional embeddings to represent row/column IDs captures essential positional information efficiently, without requiring high-dimensional semantic embeddings.

4 Experiment

We conducted three parts of the experiment: First, the main experiment, which involved performing TableLoRA and baseline experiments on 3 models in 4 datasets to demonstrate the effectiveness of TableLoRA. Second, the control experiment, in which we designed various variants of encode table methods to compare with TableLoRA, highlighting the advantages of TableLoRA's structural design. Third, further analysis, including an ablation study and in-depth analysis of TableLoRA, to deeply explore the mechanisms of TableLoRA.

4.1 Experiment Setup

Our experiments were conducted on four table-related datasets: HiTab (Cheng et al., 2022), WikiTQ (Pasupat and Liang, 2015), FeTaQA (Nan et al., 2022), and TabFact (Chen et al., 2020b). The first three are Table QA datasets, where the input consists of a table and a related query, and the task is to answer the query based on the table, with the output being the answer to the question. Among them, HiTab involves tables with complex hierarchical structures, such as multi-layered tree structures in the top or left header, as shown in Figure 1. The last dataset, TabFact, is for fact verification, where the input is a table and a related statement, and the task is to determine the truthfulness of the statement based on the table, with the output being the judgment result.

We conducted experiments on three open-source LLMs: Llama 2 (Touvron et al., 2023), Llama 3, and DeepSeek (DeepSeek-AI, 2024). In addition, we include several larger models as baselines for comparison. Due to their closed-source nature or computational constraints, we perform inference only on these models without fine-tuning. Details are provided in §A.2. We use the official metrics for each dataset, as shown in the header of Table 1.

All experiments were run on Linux machines with 4 NVIDIA Tesla A100 80G memory GPUs. The LLaMA Factory framework served as the foundation, which we extensively customized to incorporate TableLoRA-related techniques and methods. In the main experiments, LoRA and 2D LoRA share the same rank (8). PEFT is applied to the k_proj and v_proj layers. For more training de-

Table 1: Main Experiments Results. All metric numbers are in %. Δ represents the difference between the respective metric and TableLoRA, with red indicating negative values, showing how much it decreased.

Model		HiTab		WikiTQ		FeTaQA		TabFact	
		acc	Δ	acc	Δ	bleu	Δ	acc	Δ
Llama 2	Full Finetune	48.61	-0.33	46.20	5.74	32.10	4.10	52.29	-25.77
	Lora	43.00	-5.94	38.76	-1.70	25.13	-2.87	76.93	-1.13
	TableLoRA	48.94	0.00	40.46	0.00	28.00	0.00	78.05	0.00
Llama 3	Full Finetune	61.62	3.05	56.84	3.39	34.86	4.62	84.18	0.17
	Lora	57.06	-1.50	51.98	-1.47	29.09	-1.14	83.49	-0.52
	TableLoRA	58.56	0.00	53.45	0.00	30.23	0.00	84.01	0.00
DeepSeek	Full Finetune	54.92	7.99	47.01	6.59	32.15	4.87	79.44	2.40
	Lora	43.25	-3.69	37.34	-3.08	26.68	-0.61	75.20	-1.85
	TableLoRA	46.94	0.00	40.42	0.00	27.29	0.00	77.05	0.00
Large Models	GPT-4o	55.05	-	58.38	-	14.34	-	44.01	-
	Claude-3.7	56.88	-	66.90	-	13.58	-	80.14	-
	Llama 3.3-70B	22.79	-	41.30	-	7.66	-	15.99	-

tails, see §B.

4.2 Main Results

In the main experiments, we evaluated TableLoRA, the original LoRA (Hu et al., 2022), and full parameter fine-tuning across three models and four datasets. Results are presented in Table 1, and experimental parameters are detailed in §B. Few have attempted to improve PEFT for table tasks, so no additional baselines are available. To validate the model structure's superiority, we summarized common table representation methods and designed variant experiments for comparison in §4.3.1.

The results show that TableLoRA outperforms baseline LoRA fine-tuning, with a 5.9% improvement on the Llama2 model when applied to the HiTab dataset, demonstrating its effectiveness with table-based inputs.

In a low-parameter setting, TableLoRA mitigates LoRA's shortcomings compared to full fine-tuning for table tasks. It reduces the performance gap between LoRA and full fine-tuning by an average of 40.56% (excluding the TabFact outlier on Llama2, as detailed in §C). Specifically, on the HiTab dataset with Llama2, TableLoRA improves LoRA by 5.95%, matching full fine-tuning performance. This demonstrates that TableLoRA can learn complex tabular structures with fewer parameters, addressing LoRA's deficiencies.

Compared with large models, on WikiTQ, Claude 3.7 and GPT-4o generally outperform TableLoRA and related baselines, indicating that tuning-based approaches still require improvement compared to reasoning-focused LLMs, and may need enhancement through incorporation of reinforcement learning for further progress. In contrast, on FeTaQA, TableLoRA and its baselines consistently surpass reasoning LLMs, potentially because the tuning process enables better acquisition of FeTaQA's answer sentence structures, thereby achieving higher BLEU scores.

4.3 Control Experiments

4.3.1 Table Representation learning Methods

To further validate the superiority of the TableLoRA structure, we summarized common table representation learning methods in §2.2 and designed controlled experiments based on those designs. The results are shown in Table 2. Referring to the table pre-training methods before the era of LLMs (Dong et al., 2022), common table representation learning methods that can be applied to LLMs include: describing through strings, enhancing structural information through positional embedding, and enhancing structural information through attention masks. Therefore, we designed the following variant experiments, and more details are shown in §D.

Different format: Table input can be serialized using various formats, such as markdown, HTML, and CSV (the main experimental baseline in §4.2 uses markdown). Compared to the special token encoder in TableLoRA, adding a special token yielded better results than the best-performing markdown format.

Add in string sequence: The positional information in the table (e.g., which row and which column) is added to the string sequence. Compared with the 2D LoRA in TableLoRA, the latter incorporates positional information into the model through embedding, allowing the model to learn positional information more directly, resulting in

Table 2: Control Experiments on the LLama2 of the Hitab. All metric numbers are in %.

Control Exp	HiTab
TableLoRA	**48.94**
LoRA (same rank)	43.00
LoRA (comparable params)	43.06
Special Token Control Exp	
Different format (html)	32.25
Different format (csv)	43.06
Prompt Tuning	42.56
2D LoRA Control Exp	
Add in string sequence	31.06
Add in attention mask (1)	40.50
Add in attention mask (2)	41.00
Add in positional embedding	39.00

greater improvement.

Add in positional embedding: The positional information in the table (e.g., which row and which column) is added to the word embedding through positional embedding. Experiment results show that this does not bring about an effective improvement to the model. Compared with the 2D LoRA in TableLoRA, the latter incorporates positional information into each layer of the model through LoRA, continually emphasizing structural information at different layers during model inference, resulting in greater improvement.

Add in attention mask: This approach improves the model's attention mechanism by incorporating positional information from tables, focusing on two variants: (1) highlighting tokens within the same cell, and (2) highlighting tokens within the same row, column, or cell. In comparison to the 2D LoRA method in TableLoRA, the latter variant learns position vectors in different feature spaces at different layers through embedding, allowing for better integration with the model and resulting in a greater improvement in performance.

4.3.2 Special Token Encoder

We evaluated the impact of different encoders for special tokens. The special token encoder in TableLoRA is inspired by soft prompt methods, as described in §3.1. Common soft prompt methods include P-tuning and prompt tuning (Lester et al., 2021). For the control experiment, we use the **prompt tuning** encoder, which mainly consists of a word embedding layer. We modify it similarly to P-tuning for our experiments. The results in Table 2 show that the prompt tuning encoder fails to effectively learn table-related token embeddings at different positions in the sequence.

Table 3: Ablation Study on the LLama2 of the Hitab. All metric numbers are in %.

Method	HiTab	Δ
TableLoRA	**48.94**	0
w/o Special Token Encoder	47.19	-1.75
w/o 2D LoRA	44.13	-4.81
LoRA	43	-5.94

4.3.3 The Number of Parameters

To verify that the effectiveness of TableLoRA is not due to an increase in the number of parameters, we conducted experiments by increasing the parameter count of the original LoRA to make it comparable with TableLoRA. In Llama2, TableLoRA trains 0.130% of the parameters, while **LoRA (same rank)** (the baseline in §4.2) trains 0.062% of the parameters. By increasing the rank of LoRA from 8 to 16, we obtain **LoRA (comparable params)**, which trains 0.124% of the parameters. The experimental results are shown in Table 2.

The experimental results indicate that the performance improvement of TableLoRA is due to its effective encoding of the table structure, rather than to a small increase in the number of parameters. Even when LoRA's parameter count is increased to a comparable level, it only provides a 0.06% improvement over the same-rank LoRA. In contrast, TableLoRA delivers a 5.88% improvement over the parameter-comparable LoRA.

4.4 Further Analysis

4.4.1 Ablation Study

An ablation study is conducted to further investigate the components that contribute to the enhancements in the performance of the model in Table 3. The results indicate that both the Special Token Encoder and 2D LoRA can bring about improvements to varying degrees, with 2D LoRA in particular contributing to a 4.81% enhancement. Moreover, the improvements are more pronounced when both are used simultaneously.

4.4.2 Tables of Varying Complexity

To deeply explore the improvements brought by TableLoRA to tables of different complexities, we tested the enhancements of TableLoRA with varying upper/left header depths on the HiTab dataset (Cheng et al., 2022). The header depth refers to the hierarchical depth of headers in a

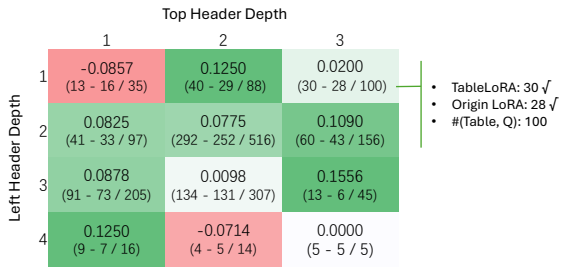

Figure 3: Heatmap of TableLoRA's Enhancement on HiTab Dataset Influenced by Top / Left Header Depth.

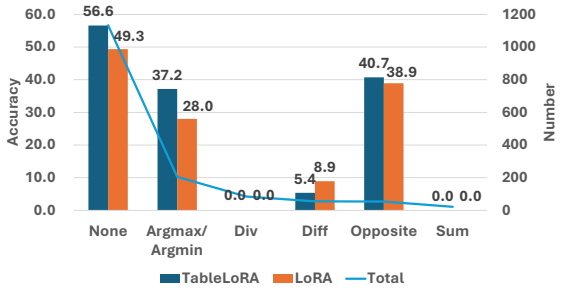

Figure 4: Performance and Data Volumes of Queries with Different Aggregations on HitTab. Accuracy is in %.

hierarchical table, where 1 represents a single-layer header with no hierarchical structure, and 2 or above indicates the corresponding deepest hierarchical depth. The results are shown in Figure 3, where the values indicate the performance of TableLoRA compared to LoRA.

It can be seen that TableLoRA provides greater improvements on more complex tables with deeper hierarchies (i.e., further to the right or lower in the heatmap). This is because TableLoRA enhances the model's understanding of the table structure, especially the relationships between rows and columns and the hierarchical relationships. However, for particularly complex tables, such as those with top/left header depths of 3/4, the improvements brought by TableLoRA are limited, indicating areas needing further improvement.

When both the top and left header depths are 1, a slight drop is observed. (1) The HiTab dataset is imbalanced, containing only 35 samples with one-hierarchy headers. This small sample size limits the reliability of statistical conclusions. Case studies of the three misclassified examples indicate that the errors were caused by perturbations. (2) To further assess the effectiveness of TableLoRA on flat tables—including those with one-hierarchy headers—we conducted experiments on additional flat-structured datasets (*e.g.*, WikiTQ, FeTaQA, TabFact, see Table 1), where TableLoRA consistently achieves stable improvements.

4.4.3 Queries with Different Aggregations

In order to further explore the underlying mechanisms and scope of TableLoRA, we conducted experiments to evaluate the model's performance on various aggregations (functions in HiTab (Cheng et al., 2022)) involved in the queries, as illustrated in Figure 4. To ensure the experiment's reliability, we selected and analyzed aggregation types from the HiTab dataset that had more than 20 samples.

For queries that require precise positioning using table structures, the improvement is more pronounced, such as with Argmax/Argmin and None. TableLoRA showed a 17.2% improvement on Argmax/Argmin, as it needs to identify which elements in the table should be included in the computation. These elements are usually located in the same row or column, and the 2D LoRA in TableLoRA helps the model to position them more accurately. TableLoRA demonstrated a 7.3% improvement on "None" queries, which do not involve aggregation and are typically for retrieval. TableLoRA enhances the model's ability to locate cell positions and retrieve the results.

The model's ability to perform numerical computations still requires improvement. Regardless of whether TableLoRA is applied, the model performs poorly on queries involving arithmetic operations such as Div, Diff, and Sum, sometimes even achieving 0% accuracy. Future research should explore how to enhance the model's computational capabilities when working with tables.

4.4.4 2D LoRA at Different Layers

The impact of different layer 2D LoRA on the results is shown in Figure 5. In the experiment, selected model layers use 2D LoRA, while unselected layers use LoRA. Details are in §E.2.

It can be observed that the earlier the layer, the more conducive it is to learning 2D information. For instance, whether the layers are divided into two or four parts, performance decreases as the number of layers increases. Additionally, the even-numbered layers, being one layer ahead of the odd-numbered layers, exhibit better performance compared to the odd-numbered layers.

4.4.5 TableLoRA with Different Rank

Figure 6 illustrates the performance of TableLoRA and LoRA under different ranks, demonstrating

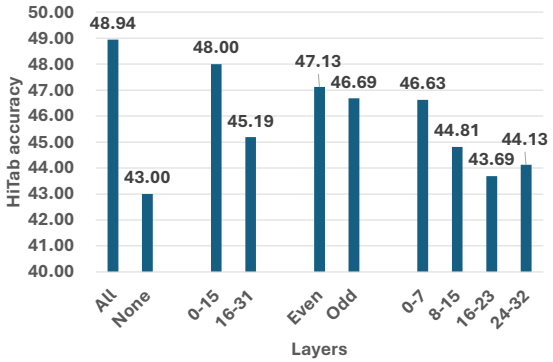

Figure 5: HiTab 2D LoRA Performance Results for Different Layers. All metric numbers are in %.

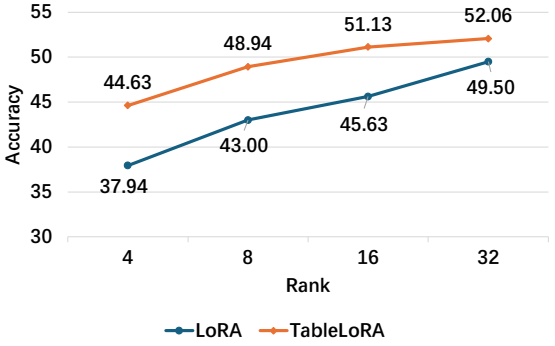

Figure 6: Performance of TableLoRA and LoRA with Different Ranks on HitTab. Accuracy is in %.

that TableLoRA consistently improves with varying ranks, indicating the scalability of the method.

4.4.6 Case Study

The improvements brought by TableLoRA are concretely visualized in Figure 1. As shown, the model is enhanced primarily in two aspects. First, the row-column correspondence has been improved. For instance, in the column selection of the query in Figure 1, the original query can accurately identify the "canadian-born" column. However, due to LoRA's error in detecting which cells belong to the same column, the final retrieved result is incorrect. Second, TableLoRA improves the understanding of the tree-structured header. For example, when performing row selection in the query in Figure 1, LoRA selects the nearest upper row for "15 to 24 years" without recognizing that it is a parent node in a hierarchical structure, with its child nodes corresponding to several rows below. By addressing these two issues, TableLoRA successfully retrieves the correct answers corresponding to the query.

5 Conclusion

In summary, this paper presents TableLoRA, a novel method for enhancing LLMs' understanding of tabular data within the PEFT. By introducing special tokens encoder for table serialization and a 2D LoRA mechanism to encode cell positions, TableLoRA addresses the structural comprehension limitations of existing models. Experiments on multiple datasets show that TableLoRA consistently outperforms vanilla LoRA and the other table representation learning methods, demonstrating significant improvements in handling table-related tasks. This approach is both efficient and effective, marking a significant advancement in the fine-tuning of LLMs for tabular data.

Limitations

One limitation of this paper is that we can only validate our method on open-source models. Unfortunately, we are unable to test its effectiveness on the current state-of-the-art GPT series models due to accessibility constraints. This limitation reduces the generalizability of our findings to the most advanced models available. Additionally, the experiments require substantial GPU computational resources, raising concerns about energy consumption and environmental sustainability.

Another limitation is that, although TableLoRA demonstrates stable improvements compared to LoRA, it still cannot match the results achieved through full finetuning. This discrepancy suggests that while TableLoRA offers notable benefits, it may not yet reach the performance level achieved by fine-tuning techniques.

Ethics Statement

The datasets and other associated resources utilized in this study are publicly available and widely used in various other existing work. All the datasets used in this paper have been reviewed to ensure that they do not contain personally identifiable information or offensive content. However, since these datasets are sourced from the Internet, potential bias may still be present. Furthermore, despite our careful review, the process involving the LLMs may inadvertently introduce inappropriate information into the evolved data.

Acknowledgments

We thank all anonymous reviewers for their valuable comments. Xinyi He and Zejian Yuan were supported in part by the National Key R&D Program of China (2023YFB4704900) and NSFC (61976170, 62088102).

References

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020a. Kgpt: Knowledge-grounded pre-training for data-to-text generation. *arXiv preprint arXiv:2010.02307*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020c. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

DeepSeek-AI. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: table understanding through representation learning. *Proceedings of the VLDB Endowment*, 14(3):307–319.

Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5426–5435. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Haoyu Dong, Yuzhang Tian, Jianbo Zhao, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. 2024. Spreadsheetllm: Encoding spreadsheets for large language models. In *The 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP '24)*.

Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. 2021. Mate: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619.

Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, Zejian Yuan, and Dongmei Zhang. 2024a. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18206–18215.

Xinyi He, Mengyu Zhou, Mingjie Zhou, Jialiang Xu, Xiao Lv, Tianle Li, Yijia Shao, Shi Han, Zejian Yuan, and Dongmei Zhang. 2023. AnaMeta: A table understanding dataset of field metadata knowledge shared by multi-dimensional data analysis tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9471–9492, Toronto, Canada. Association for Computational Linguistics.

Xinyi He, Jiaru Zou, Yun Lin, Mengyu Zhou, Shi Han, Zejian Yuan, and Dongmei Zhang. 2024b. CoCoST: Automatic complex code generation with online searching and correctness testing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19433–19451, Miami, Florida, USA. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.

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

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456.

Syrine Krichene, Thomas Müller, and Julian Martin Eisenschlos. 2021. Dot: An efficient double transformer for nlp tasks with tables. *arXiv preprint arXiv:2106.00479*.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. *ArXiv*, abs/2211.11501.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024a. Can llm already serve as a database interface? a big bench for large-scale

database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024b. Table-gpt: Table fine-tuned gpt for diverse table tasks. In *SIGMOD 2024*.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jianguang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. Gpt understands, too. *AI Open*, 5:208–215.

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`.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024a. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. *Preprint*, arXiv:2305.13062.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2024b. TAP4LLM: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10306–10323, Miami, Florida, USA. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *ICLR*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023. Tablellama: Towards open large generalist models for tables. *Preprint*, arXiv:2311.09206.

Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, et al. 2024. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Mengyu Zhou, Qingtao Li, Xinyi He, Yuejiang Li, Yibo Liu, Wei Ji, Shi Han, Yining Chen, Daxin Jiang, and Dongmei Zhang. 2021. Table2charts: Recommending charts by learning shared table representations.

In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2389–2399.

Mengyu Zhou, Wang Tao, Ji Pengxin, Han Shi, and Zhang Dongmei. 2020. Table2analysis: Modeling and recommendation of common analysis patterns for multi-dimensional data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):320–328.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W. Huang, Jie Fu, Xiang Yue, and Wenhu Chen. 2024. Structlm: Towards building generalist models for structured knowledge grounding. *Preprint*, arXiv:2402.16671.

A Experiment Configuration Details

A.1 Datasets Selection

We selected four datasets (HiTab, WikiTQ, FeTaQA, TabFact) because they are classic, reliable, and cover diverse table structures and tasks. Table 4 summarizes their characteristics. Varying instruction systematically test the model's ability to locate and retrieve information from tables at different difficulty levels, while diverse output rigorously assess its capacity to generate responses with varying linguistic or structural demands.

Table 4: Dataset Selection.

Dataset	Table	Instruct	Output
HiTab	hierarchical	query	words
WikiTQ	flat	query	words
FeTaQA	flat	query	sentences
TabFact	flat	fact/statement	bool

A.2 Models Used

For all experiments conducted in this paper, we employed three pre-trained large language models (LLMs): DeepSeek LLM-7B Chat (deepseek-ai/deepseek-llm-7b-chat), Llama 2-7B Chat (meta-llama/Llama-2-7b-chat-hf), and Meta Llama 3-8B Instruct (meta-llama/Meta-Llama-3-8B-Instruct). Llama 2 is a collection of pre-trained and fine-tuned models optimized for dialogue, outperforming most open-source chat models. Llama 3, the next generation of Llama models, offers enhanced reasoning capabilities and strong performance across various benchmarks. DeepSeek is an advanced model trained on a large English and Chinese dataset, available in both base and chat versions, and open-sourced for the research community. These models were selected based on their relevance in the current landscape of language model research and their suitability for fine-tuning tasks on structured datasets, such as the TabFact dataset.

B Training Configurations

B.1 Hardware and Frameworks

We employed machines with four NVIDIA A100 GPUs for fine-tuning. The LLaMA Factory framework served as the foundation, which we extensively customized to incorporate TableLoRA-related techniques and methods. To enable full-parameter fine-tuning of large-scale models, we used DeepSpeed version 0.14.4. The configuration file employed during fine-tuning was the default DeepSpeed zero-2 stage configuration file from the example directory provided by the LLaMA Factory framework. To ensure consistency and eliminate any framework-induced bias in the results, we applied the same DeepSpeed framework and configuration file for both LoRA fine-tuning and TableLoRA fine-tuning.

B.2 LoRA Fine-Tuning

The LoRA fine-tuning used eight LoRA ranks with an alpha value of 16 and a dropout rate of 0.1. In all cases, training employed a batch size of 8 per device, with gradient accumulation steps of 2, a learning rate of 5e-6, and a cosine scheduler. Training was conducted for three epochs with a maximum sequence length of 4,000 tokens on the TabFact dataset and 1,000 tokens on other datasets. Mixed precision (FP16) and distributed training were enabled using DeepSpeed. TableLoRA, a variant of LoRA, used the same fine-tuning parameters.

B.3 TableLoRA Fine-Tuning

The main implementation of TableLoRA involves both Special Tokens and 2D LoRA. The training of Special Tokens employs p-tuning with its default parameters, while the hyperparameters for 2D LoRA are consistent with those of LoRA. Specifically, for 2D LoRA, the maximum values for columns and rows are set to 40 and 600 on other datasets, and 50 and 600 on the TabFact dataset.

B.4 Full-parameter Tuning

The full fine-tuning process employed a learning rate of 5e-6 with a cosine scheduler and a maximum gradient norm of 1.0. Training was conducted over 3 epochs with a maximum sequence length of 4000 tokens, utilizing a batch size of 8 per device and gradient accumulation steps of 2. The model training leveraged mixed precision (FP16) and distributed training capabilities provided by DeepSpeed. To ensure efficiency and stability, the preprocessing pipeline involved 16 workers, and warmup steps were set to 0. Additionally, the training process included advanced optimization techniques such as AdamW, with careful monitoring of loss curves and model performance metrics throughout.

B.5 Consistency in Training

To ensure comparability across different models and datasets, we applied the same training configurations to all experiments. This uniformity minimized the influence of hyperparameter differences, isolating the effects of model architectures

and dataset characteristics. By maintaining consistent training parameters, we could confidently attribute variations in performance to the intrinsic properties of the models or datasets rather than external factors.

All models demonstrated expected convergence behaviors, with the training process yielding the lowest observed loss values for each model. This result confirmed the stability and reliability of the training procedures. The standardized and well-calibrated configurations enabled us to conduct a robust comparison across different models and fine-tuning techniques, ultimately producing meaningful and consistent insights.

C Details of Main Results

To quantify the average gap reduction between different fine-tuning methods, we calculate the relative improvement of the TableLoRA method over the LoRA method, normalized by the difference between the full finetune method and the LoRA method. Mathematically, this can be expressed as:

$$\text{Gap Reduction} = \frac{1}{n}\sum_{i=1}^{n}\frac{P_{\text{full\_finetune}_i} - P_{\text{tablelora}_i}}{P_{\text{full\_finetune}_i} - P_{\text{lora}_i}}$$

where $P_{\text{tablelora}_i}$ represents the performance metric obtained using the TableLoRA method, P_{lora_i} represents the performance metric obtained using the LoRA method, and $P_{\text{full\_finetune}_i}$ represents the performance metric obtained using the full finetune method for the i-th instance. The total number of instances is denoted by n.

When fine-tuning LLaMA2 on the TabFact dataset, we observed that full-parameter tuning resulted in significantly lower accuracy compared to LoRA, despite both approaches being applied to the same task. One possible reason for this is that LLaMA2's pretraining may not be well-aligned with the task-specific requirements of TabFact, particularly in terms of logical reasoning and table-based data modeling. Full-parameter fine-tuning, which adjusts all weights, might inadvertently interfere with the model's pre-existing knowledge, disrupting its ability to generalize effectively. On the other hand, LoRA's approach, which only adjusts a small set of parameters, focuses more on task-specific patterns, leading to better performance. Furthermore, we encountered issues related to optimization during full-parameter fine-tuning, such as gradient vanishing, which made the optimization process unstable. This instability often led to convergence problems, preventing the model from reaching an optimal solution. LoRA, due to its reduced parameter space, was less prone to such issues and exhibited a more stable convergence. Additionally, TabFact's inherent noise and specific patterns could have been more effectively captured by LoRA, as it is less likely to overfit to irrelevant features. In contrast, other models like LLaMA3 and DeepSeek may have better adapted to the task during pretraining, resulting in higher accuracy when subjected to full-parameter fine-tuning.

D Details of Control Experiment

Different format: When comparing with the special token encoder, three types of table serialization formats are involved: markdown, HTML, and CSV. In the experiment, we maintained consistency with the pandas library for specific serialization methods: `DataFrame.to_markdown()`, `DataFrame.to_html()`, and `DataFrame.to_csv()`.

Add in string sequence: We add the position information of each cell to the string sequence. Specifically, the position string "(row_idx, col_idx)" of each row and column is added to the front of the string of each serialized table cell.

Add in positional embedding: We use Sinusoidal Positional Embedding to encode the row/-column indices separately. The row and column indices are consistent with those in Equation (3). The calculated embeddings are added to the word embeddings, similar to the original positional embeddings in the transformer, for computation.

Add in attention mask: The positional attention mask is combined with the causal mask typically used in LLMs, ensuring that causal constraints are preserved while embedding the structural information of the table. This integration occurs at each layer during the forward pass, enabling the model to consistently emphasize table-specific structural patterns throughout inference. At the core of this method is the concept of weight amplification, which boosts the model's focus on structural information. Tokens within the same cell receive higher attention weight (e.g., adding a mask value of 1), prompting the model to prioritize these tokens. Tokens within the same row or column receive a lower amplification (e.g., mask value of 0.5), highlighting their contextual relevance to a lesser extent, while

Table 5: Ablation Study on each model for each dataset. All metric numbers are in %.

Model		HiTab	WikiTQ	FeTaQA	TabFact
Llama 2	TableLoRA	**48.94**	**40.46**	**28.00**	**78.05**
	w/o Special Token Encoder	47.19	39.73	27.72	78.00
	w/o 2D LoRA	44.13	39.60	25.91	77.37
	LoRA	43.00	38.76	25.13	76.93
Llama 3	TableLoRA	**58.56**	53.45	**30.23**	**84.01**
	w/o Special Token Encoder	58.43	**53.52**	29.81	83.88
	w/o 2D LoRA	57.25	52.75	30.19	83.23
	LoRA	57.06	51.98	29.09	83.49
DeepSeek	TableLoRA	**46.94**	**40.42**	27.29	**77.05**
	w/o Special Token Encoder	45.63	38.87	**27.50**	76.78
	w/o 2D LoRA	44.44	37.71	26.92	75.62
	LoRA	43.25	37.34	26.68	75.20

tokens not sharing a row or column relationship receive no additional weight, maintaining neutral attention scores.

E Details of Further Analysis

E.1 Ablation Study

The ablation results for each dataset are shown in Table 5.

E.2 Performance of 2D LoRA at Different Layers

We conduct the following sets of experiments for different model layers: (1) Halving: The model layers are divided into two halves for the experiment, i.e., layers 0-15 and 16-31. (2) Odd and Even: The model layers are divided into odd and even numbers for the experiment, i.e., layers 0, 2, ..., 30 and layers 1, 3, ..., 31. (3) Quartering: The model layers are divided into four quarters for the experiment, i.e., layers 0-7, 8-15, 16-23, and 24-31.

E.3 Efficiency Comparison

Table 6: Training Efficiency. The experiments were conducted on HiTab LLama2, using the same batch size for comparison. GPU Memory refers to the total GPU memory usage when training is stable.

Method	Duration	GPU Memory
Full Finetune	45min	280G
LoRA	33min	160G
TableLoRA	36min	168G

Compared to Full Finetune, TableLoRA significantly reduces computational resources and time while bridging the performance gap between LoRA and Full Finetune. As shown in Table 6, TableLoRA uses similar time and GPU resources as LoRA, with time being 80% of that of Full Finetune and GPU usage being 57%. Notably, for a fair comparison, the experiments used the same batch size. If the batch size were increased so that TableLoRA and Full Finetune used the same GPU memory, TableLoRA could achieve even better time efficiency.

F Prompt

An example of the prompt is shown below (the example is the sample in Figure 1):

```
<s> [INST] This is a hierarchical table
    question answering task. The goal
    for this task is to answer the given
     question based on the given table.
    The table might be hierarchical.
    Here is the table to answer this
    question. Answer the question.
/*
[TAB] [ROW] [CELL]  low income [CELL]
    total [CELL]  total [CELL]  canadian-
    born [CELL]  canadian-born [CELL]
    immigrant [CELL]  immigrant [ROW]
    [CELL]  low income [CELL]  female
    [CELL]  male [CELL]  female [CELL]
    male [CELL]  female [CELL]  male [ROW]
    [CELL]  low income [CELL]  percentage
    [CELL]  percentage [CELL]  percentage
    [CELL]  percentage [CELL]  percentage
    [CELL]  percentage [ROW] [CELL]  total
     age groups [CELL] [CELL] [CELL] [CELL]
    [CELL] [CELL] [ROW] [CELL]  visible
    minority [CELL]  21.9 [CELL]  21.1
    [CELL]  19.3 [CELL]  18.5 [CELL]  22.0
    [CELL]  21.0 [ROW] [CELL]  not a
    visible minority [CELL]  14.3 [CELL]
    12.2 [CELL]  14.2 [CELL]  12.2 [CELL]
    14.3 [CELL]  12.3 [ROW] [CELL]  under
    15 years [CELL] [CELL] [CELL] [CELL]
```

```
    [CELL] [CELL] [ROW] [CELL]  visible
    minority [CELL]  25.4 [CELL]  25.2
    [CELL]  22.3 [CELL]  21.8 [CELL]  34.3
    [CELL]  36.2 [ROW] [CELL]  not a
    visible minority [CELL]  15.2 [CELL]
    15.2 [CELL]  14.9 [CELL]  14.9 [CELL]
    26.1 [CELL]  25.7 [ROW] [CELL]  15 to
    24 years [CELL] [CELL] [CELL] [CELL]
    [CELL] [CELL] [ROW] [CELL]  visible
    minority [CELL]  26.3 [CELL]  26.2
    [CELL]  18.6 [CELL]  17.9 [CELL]  29.2
    [CELL]  28.6 [ROW] [CELL]  not a
    visible minority [CELL]  15.8 [CELL]
    13.7 [CELL]  15.4 [CELL]  13.3 [CELL]
    20.8 [CELL]  18.7 [ROW] [CELL]  25 to
    54 years [CELL] [CELL] [CELL] [CELL]
    [CELL] [CELL] [ROW] [CELL]  visible
    minority [CELL]  20.7 [CELL]  19.3
    [CELL]  12.6 [CELL]  11.1 [CELL]  21.3
    [CELL]  19.8 [ROW] [CELL]  not a
    visible minority [CELL]  12.7 [CELL]
    11.2 [CELL]  12.5 [CELL]  10.9 [CELL]
    14.3 [CELL]  13.7 [ROW] [CELL]  55 to
    64 years [CELL] [CELL] [CELL] [CELL]
    [CELL] [CELL] [ROW] [CELL]  visible
    minority [CELL]  17.1 [CELL]  16.8
    [CELL]  17.3 [CELL]  16.9 [CELL]  17.0
    [CELL]  16.7 [ROW] [CELL]  not a
    visible minority [CELL]  14.4 [CELL]
    13.2 [CELL]  14.5 [CELL]  13.2 [CELL]
    13.4 [CELL]  13.1 [ROW] [CELL]  65
    years and over [CELL] [CELL] [CELL]
    [CELL] [CELL] [CELL] [ROW] [CELL]
    visible minority [CELL]  17.3 [CELL]
    14.3 [CELL]  15.1 [CELL]  9.8 [CELL]
    17.4 [CELL]  14.4 [ROW] [CELL]  not a
    visible minority [CELL]  16.2 [CELL]
    9.5 [CELL]  17.1 [CELL]  10.0 [CELL]
    12.9 [CELL]  None
*/
Table Caption: The table caption is this
     table displays the results of
    prevalence of low income. the
    information is grouped by low income
     (appearing as row headers), total,
    canadian-born, immigrant, female and
     male, calculated using percentage
    units of measure (appearing as
    column headers).
Question: within the population that did
     not belong to a visible minority
    group, what was the percentage of
    canadian-born women aged 15 to 24 in
     a low-income situation?
The answer is:
[/INST]
```