**Building an Chatbot for Work Management using DeepSeek and Orthogonal LoRA**

Dinh Cong Bang,   
Phan Duy Hung

FPT University, Hanoi, Vietnam  
bang23mse13147@fsb.edu.vn, hungpd2@fe.edu.vn

**Abstract.** In the current era of digital transformation, integrating AI into enterprise systems has become a necessity. This thesis develops a work management system aimed at automating report generation, onboarding guidance, and point-of-contact lookup. The system enhances productivity by automatically generating reports, suggesting tasks, and guiding new employees. The AI model is trained on data from Jira and internal chat groups, enabling it to understand real-world contexts and business processes.

At the core of the system is the DeepSeek model, fine-tuned using the LoRA (Low-Rank Adaptation) technique combined with Multi-Stage Fine-Tuning. small number of low-rank parameters, reducing the number of trainable parameters by thousands of times compared to full fine-tuning [1]. The model is trained in multiple rounds, with matrices orthogonalized in each round according to specific strategies and objectives. This approach improves the model’s ability to absorb and process information effectively.

**Keywords:** LLM, Orthogonal Regularization, Work Management System, Context-aware AI, LoRA.

1. Introduction
   1. Problem & Motivation

In today's enterprise environment, the demand for effective task tracking, resource allocation, and progress evaluation is growing rapidly, especially with the increasing scale of projects and the complexity of operational workflows. Although many tools such as Jira or Trello have been widely adopted, they primarily serve as manual tracking and storage systems, requiring frequent human intervention. This leads to a time-consuming and inconsistent process when it comes to compiling reports, suggesting tasks, or onboarding new employees.

Simultaneously, the rapid advancement of large language models (LLMs) such as ChatGPT, Grok, and DeepSeek presents new opportunities to automate work management tasks. DeepSeek is an open-source LLM family trained on datasets consisting of trillions of tokens, with enhanced reasoning capabilities achieved through multi-stage fine-tuning strategies [4]. However, to effectively leverage these models in specific enterprise environments, fine-tuning on internal data becomes essential. This introduces challenges in terms of resource efficiency, scalability, and the risk of losing the model's foundational knowledge if not trained properly [2].

In response to these practical needs, this thesis focuses on developing an artificial intelligence system that supports enterprises in tracking, analyzing, and optimizing internal operations. The system is based on DeepSeek, one of the most prominent open-source LLMs today, trained on large-scale datasets, capable of contextual reasoning, and with Vietnamese language support. However, to effectively apply the model within a specific business environment — where internal language, domain-specific terminology, and unique workflows exist — fine-tuning is imperative. Fine-tuning not only allows the model to adapt to organization-specific data but also unlocks the ability to automate a range of processes such as: generating work progress reports, suggesting context-aware tasks, and onboarding new staff through personalized guidance.

* 1. Literature Review

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning technique proposed by [1], aiming to minimize the number of parameters that need to be updated when fine-tuning large language models. In traditional fine-tuning, the entire weight matrix W of the model is updated during training, which requires substantial computational resources and memory—especially when the model has billions of parameters.

Instead of directly updating the original weight matrix W, LoRA keeps W intact and adds a low-rank residual, denoted ΔW. For a weight matrix W∈R*d×k*, meaning d rows and k columns, LoRA introduces a new matrix computed as the product of two smaller matrices.

Where:

* A∈Rr×k
* B∈Rd×r
* r≪min(d,k): the rank of the update, representing the compression level

Finally, the actual weight used in the model after applying LoRA becomes.

Where α is a scaling factor that ensures the overall influence of ΔW is balanced, regardless of the selected rank r. This design allows the rank r to be adjusted without disproportionately impacting the base model’s behavior [1].

In practice, there are several strategies to select an optimal α:

Determined AI recommends starting with α = r, then keeping this value fixed when changing the rank, avoiding the need to retune α [12].

Hugging Face AutoTrain documentation sets a default of α = 32 for rank = 16, and warns that setting α ≫ r can lead to overfitting [13].

Sebastian Raschka suggests a heuristic of α / r = 2 [14] to balance the strength of the update.

[15] found that using γr=α/r can slow down learning and reduce fine-tuning performance, especially with larger values of r. To address this, the author proposed a new approach called rsLoRA (rank-stabilized LoRA), which uses:

This helps stabilize the magnitude of ΔW as r increases, leading to improved convergence and training efficiency, without compromising the model’s reasoning ability.

However, studies [5] [6] [7] [8] have indicated that when applying LoRA to large models or multi-task training scenarios, simply adding low-rank matrices without directional control may lead to overlapping or low-diversity representations. To address this, orthogonalization has been proposed as an important enhancement to improve representation quality and generalization capability. In addition, orthogonalization acts as a form of soft regularization, helping to mitigate overfitting—especially when training on small datasets—and provides better control over the model’s convergence behavior.

While orthogonalization is effective at preserving previous knowledge and reducing task interference through these constraints, it can also unintentionally limit the representational space the model can access. As highlighted by [6] [7], models that are updated only within a restricted representational subspace, though resilient to interference, often exhibit reduced learning flexibility. They cannot fully adjust all weight components, which in turn limits their capacity to generalize or adapt optimally when tasks require overlapping or correlated features.

From this analysis, orthogonalization should be applied selectively, primarily to modules like q\_proj, k\_proj, v\_proj, where linear representations play a central role in retrieving, integrating, and responding to information. This approach leverages the strengths of orthogonalization for knowledge retention and novel representation learning, while avoiding the unintended side effects of overly restricting the model’s expressiveness.

* 1. Contribution

This study focuses on the design and development of a work management system powered by a large language model (LLM), specifically DeepSeek, with components fine-tuned using Low-Rank Adaptation (LoRA) to ensure deployment efficiency in resource-constrained enterprise environments.

The proposed system architecture is tailored for small to medium-sized businesses (SMEs), where task-related data is sourced from platforms such as Trello, Jira, and internal documents (e.g., workflows, guidelines). These data sources are transformed into a JSONL format containing question–answer pairs or structured dialogues suitable for supervised fine-tuning.

To adapt to the specific domain, the DeepSeek-R1-Distill-Qwen-1.5B model is fine-tuned using LoRA in combination with orthogonalization techniques. These constraints are applied both with respect to the model's original weight matrices and internally among the LoRA vectors, in order to enhance generalization and reduce representational redundancy. This setup enables the model to better learn from limited datasets and extract knowledge from complex or underrepresented samples.

This strategy enables lightweight adapter training atop a pre-trained large model without disrupting previously acquired knowledge. It ensures that each adapter is capable of learning new information independently while minimizing overlap with the original model's learned space.

1. Methodology
   1. Model Architecture

The internal chatbot system is designed using a three-tier architecture consisting of:

* Data Collection
* Model Processing & Training
* Inference Deployment via API

The overall architecture is illustrated in Figure 1.

A diagram of a software system

AI-generated content may be incorrect.

**Fig. 1.** Overall architecture

At the first layer, data is collected from three main sources: internal instructional documents, task management systems such as Jira or Trello, and survey/interview responses from organizational staff. Once collected, the data is standardized into question–answer pairs formatted for instruction-tuning, suitable for large language models.

The model training layer applies the LoRA technique combined with orthogonality constraints to mitigate catastrophic forgetting and enhance the model’s ability to learn diverse directions. The base language model used is DeepSeek-R1-Distill-Qwen-1.5B, a lightweight, high-performance model with Vietnamese language support.

Finally, the fine-tuned model is stored and deployed as an API using the Hugging Face Transformers platform. This API enables internal systems to query the chatbot and receive context-aware responses. Conversation histories are retained to improve contextual understanding in future interactions. To ensure efficiency and control context length, the system uses only a fixed number of recent dialogue turns (5–10) to construct the input prompt for each inference session.

* 1. Data Extraction

The dataset is collected from three primary sources:

* Data extracted from work management systems such as Jira or Trello
* Data sourced from internal and publicly available documentation

For data from task management systems, this source is generally very clean. Only minimal preprocessing is required—such as removing intermediary lines (as shown in Fig 2) and adjusting pronoun usage—to convert it into training data suitable for AI models.

A screenshot of a chat

AI-generated content may be incorrect.

**Fig. 2.** Data taken from Jira

Comments from these systems often contain repetitive phrases, system-generated templates, or follow a standardized format. For example, expressions like “please assist”, “process quickly”, etc., account for a significant portion of the content. These patterns can be filtered effectively using keyword queries or regular expressions. A major limitation of data from systems like Jira is that many processing steps occur outside the platform—for instance, in code files or other operational systems—while Jira only serves as a reporting or status-tracking tool. As a result, only around 20% of the total data can be effectively converted into usable training samples.

Beyond Jira and Trello, companies often maintain Excel-based FAQ files during software, service deployments, intended for customers or internal users. These datasets are highly valuable as the questions are often de-duplicated and the answers are detailed.

Instructional documents, official memos, and regulatory texts—especially legal documents, policies, and government decrees—are considered the most accurate sources of information. However, a major challenge lies in the fact that these documents require a high level of abstraction and are not naturally formatted as task-oriented conversations.

This paper proposes allowing large language models such as ChatGPT or Gemini to read the entire document beforehand in order to grasp the overall knowledge. Then, the models can be prompted to generate dialogue-style exchanges based on the content of the document.

* 1. Data Formatting for Instruction-Tuning

After collecting the data in the form of question–answer pairs, it needs to be formatted for training. This paper recommends using the Hugging Face Chat Template format [17]. The Hugging Face Chat Template is a multi-turn dialogue format designed for assistant-style models. Data is structured as a list of {role, content} pairs and converted into a training prompt using a predefined chat\_template function within the tokenizer. This function is responsible for concatenating all dialogue turns into a single input string. The format enables the model to learn natural conversation structures, maintain context more effectively, and support tasks such as query answering, data explanation, and function calling. It is the standard format used by models like LLaMA2-chat, Mistral, DeepSeek, and is natively supported by the Hugging Face Transformers library during both training and inference.

**Table 1.** Data in Hugging Face Chat Template format

|  |
| --- |
| {"messages": [{"role": "user", "content": "How can I reset my password?"}, {"role": "assistant", "content": "To reset your password, click on 'Forgot password' on the login page and follow the instructions."}]}  {"messages": [{"role": "user", "content": "What is the return policy of your store?"}, {"role": "assistant", "content": "Our return policy allows returns within 30 days of purchase, with the original receipt and packaging."}]}  {"messages": [{"role": "user", "content": "Can you explain how invoices work in this system?"}, {"role": "assistant", "content": "Invoices record transactions and are automatically generated when a purchase is made. You can view them under the 'Billing' section."}]}  {"messages": [{"role": "user", "content": "Is there an API to check invoice status?"}, {"role": "assistant", "content": "Yes, you can use the `getInvoiceStatus` API by providing the invoice ID as a parameter."}]}  {"messages": [{"role": "user", "content": "Show me an example of a valid XML invoice payload."}, {"role": "assistant", "content": "<Invoice><ID>12345</ID><Date>2025-06-24</Date><Amount>100.00</Amount></Invoice>"}]} |

For data sourced from task management systems, the task creator is assigned the role of "user", as they typically initiate the inquiry and provide the main context. Other participants in the thread are marked as "assistant." This data often consists of multi-turn exchanges.

To ensure the model retains both factual accuracy and expressive flexibility, this paper adopts the mixing strategy from[18], which recommends the golden ratio (61.8 : 38.2) for real-to-generated data. This ratio preserves the authenticity of real-world data while expanding the linguistic diversity through synthetic samples.

Cosine Similarity analysis confirms high diversity in real data. The average similarity between each sample and all others mainly falls in the 0.04–0.10 range, indicating significantly different content. Only about 5–10% of samples exceed the 0.12 similarity threshold, meaning there is low paraphrasing risk. This demonstrates that the real dataset is highly diverse and well-suited for fine-tuning without overfitting.

A graph of a number of cosine similar to others

AI-generated content may be incorrect.

**Fig. 3.** Data taken from Jira

In contrast, ChatGPT-generated data shows reduced quality compared to Jira and Excel data, with more than 120 near-duplicate pairs (cosine similarity > 0.90), indicating heavy paraphrasing

A graph of a number of bars

AI-generated content may be incorrect.

**Fig. 4.** Data generate from Chat GPT

* 1. Modeling Module

The The base model used is DeepSeek-R1-Distill-Qwen-1.5B, which is fine-tuned using the LoRA technique combined with orthogonality constraints, following the pipeline below:

* Tokenization & Data Formatting: The data is preprocessed and standardized into a chat-style format compatible with the Qwen model, then tokenized using the DeepSeek tokenizer.
* Pre-training Setup: Before training begins, the original LoRA weight matrices A from the base model are loaded and stored as detached tensors. These matrices are later used to compute external orthogonality constraints relative to the current adapter.
* Training Phase: During training, the row vectors of each matrix A are forced to be orthogonal to one another by minimizing the error between the product A·Aᵗ and the identity matrix I. This internal orthogonality constraint ensures that each adapter learns new and independent representations, avoiding redundancy with previously learned directions.

The model introduces two specific hyperparameters to enforce these orthogonality constraints within LoRA modules:

* Lambda\_internal: Controls the degree of internal orthogonality, i.e., the independence among row vectors within a single LoRA matrix A.
* Lambda\_external: Controls the degree of external orthogonality, i.e., the independence between the current LoRA matrix A and the corresponding matrix A from the base model.

The total loss used during training combines the main model loss with two regularization terms, as defined by the following formula:

Orthogonalization is applied only to selected modules, including:

* q\_proj: As this module determines the query direction of each token, enforcing orthogonality here is crucial to avoid repetitive query patterns.
* v\_proj: Since this module creates the value vectors used in attention aggregation, orthogonality ensures the model learns diverse and complementary representations, enhancing overall information diversity.

1. Experiments and Results
   1. Dataset

dataset was collected from an internal Jira-based work management system and documentation related to the electronic invoice processing workflow. Among these, Excel-based FAQ files were gathered during the implementation of the e-invoicing system in accordance with ND70. Additionally, a portion of the data was automatically generated from official instructional documents released after 25/05/2025, using the ChatGPT model to ensure broad coverage and linguistic diversity within the dataset.

The dataset was constructed using the “golden ratio” split between real and generated data, consisting of 66.6% real-world data (1,000 samples) and 33.3% synthetic data (618 samples), totaling 1,618 samples.

After collection, the data underwent preprocessing, including deduplication and normalization, and was formatted into a standardized chat-based structure compatible with language model training. A summary of the dataset distribution and token-level statistics is provided in Table 2:

**Table 2.** Average token length and quantity statistics by data source

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Source | Number of Samples | Median | Longest | Average Length |
| Jira(Real data) | 550 | 96 | 367 | 111.12 |
| Excel | 450 | 101 | 185 | 102.84 |
| GPT Generate | 618 | 88 | 161 | 88.35 |
| Total | 1618 | 101 | 237.67 | 100.1 |

* 1. Implementation Details

In the experiments, the model used was DeepSeek-R1-Distill-Qwen-1.5B, a lightweight yet high-performance variant of the DeepSeek series. This model supports the Vietnamese language and demonstrates strong logical reasoning capabilities. With 1.5 billion parameters and open-source availability, it is well-suited for deployment and fine-tuning in small-to-medium enterprise environments without the need for expensive computational infrastructure.

The fine-tuning process was carried out using the LoRA technique, structured into two training rounds, each involving the training of three different models. The orthogonal constraint settings for each model are detailed in Table 3.

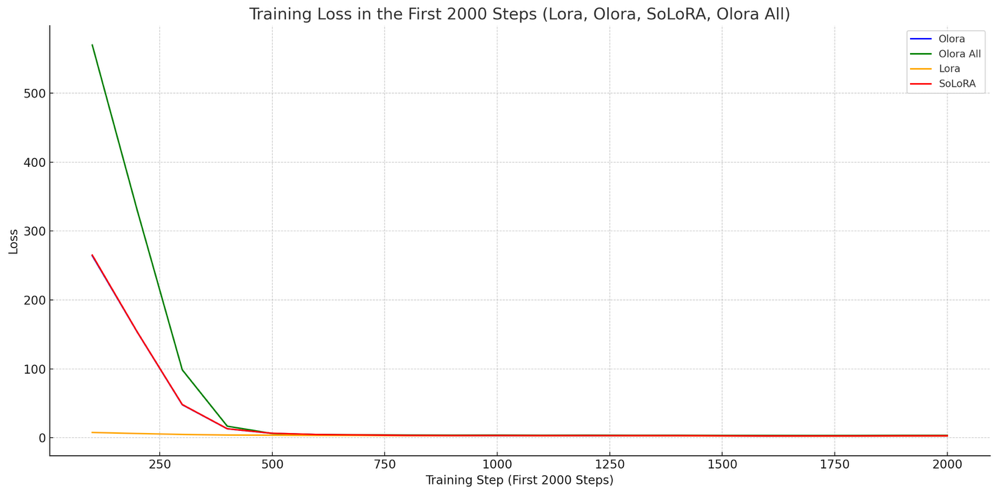
**Table 3.** Orthogonal extrusion module

|  |  |  |
| --- | --- | --- |
| Model | Orthogonality Rule | Orthogonal Constraint Modules |
| Lora | Not orthogonal | None |
| OLora | Orthogonal to the original model | q\_proj, v\_proj |
| SoLora | Orthogonal to the original model | q\_proj, v\_proj |
| Olora All | Orthogonal to the original model and other A vectors | q\_proj, v\_proj, k\_proj, o\_proj, gate\_proj |

* 1. Result

Trong During training, all models exhibited a stable convergence trend, with the loss gradually decreasing over steps. Notably, SoLoRA and OLoRA, despite applying orthogonal constraints to key attention modules, maintained a convergence speed comparable to the original LoRA model without constraints. This indicates that orthogonal regularization can enhance the discriminative capacity of adapters without significantly affecting training time or performance, provided the targeted modules are carefully selected.

In contrast, the OLoRA All model, which applied strong orthogonality constraints across all major modules, experienced significantly slower training per step and required more steps to reach convergence.



**Fig. 5.** Training Loss Comparison

Table 4 is the index of each model after testing with 100 data.

**Table 4.** Testing Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | LoRA | OLoRA | SOLoRA | OLoRA All |
| BLEU-4 | 0.1504 | 0.1607 | 0.0985 | 0.0121 |
| ROUGE-L | 0.3009 | 0.3119 | 0.2549 | 0.1736 |
| BERTScore-F1 | 0.2315 | 0.1987 | 0.1119 | 0.0248 |
| BERTScore-P | 0.0808 | 0.0522 | -0.0097 | -0.0799 |
| BERTScore-R | 0.3917 | 0.3573 | 0.239 | 0.1334 |
| Avg Cosine Similarity | 0.6325 | 0.6137 | 0.5456 | 0.5057 |
| Distinct-1 | 0.2852 | 0.4007 | 0.4438 | 0.4107 |
| Distinct-2 | 0.6719 | 0.7332 | 0.7632 | 0.796 |
| Avg Length | 49.01 | 41.48 | 42.14 | 47.38 |
| Toxicity Score | 0.0907 | 0.0892 | 0.0589 | 0.053 |

Table Based on the evaluation results, both SoLoRA and OLoRA demonstrate strong and competitive performance compared to the LoRA model, even outperforming it in several key metrics. OLoRA achieves the highest scores in BLEU-4 and ROUGE-L, indicating its ability to generate responses that are structurally accurate and closely aligned with reference phrasing. On the other hand, SoLoRA excels in Distinct-1, reflecting greater linguistic diversity and a reduced tendency to produce repetitive outputs. Both models apply orthogonal constraints selectively to critical attention modules, enabling them to learn novel representations without compromising training stability or convergence speed.

In contrast, the OLoRA All model, which enforces orthogonality across all major modules, shows significant performance degradation. It records the lowest scores across semantic similarity (BERTScore-F1), lexical accuracy (BLEU-4), and sequence overlap (ROUGE-L), suggesting that excessive orthogonal regularization can overly constrain the model’s expressive capacity. Although Distinct and Toxicity scores are slightly better, the overall output quality suffers. This highlights the importance of applying orthogonality selectively—targeting essential modules allows the model to retain expressive power while benefiting from improved representation diversity.

* 1. Conclusion And Future Work

The results from the four models underscore the critical importance of strategically selecting which modules should undergo orthogonal constraints. Specifically, selectively applying orthogonality to the q\_proj and v\_proj modules significantly improves model performance and enhances generalization ability when handling novel queries. In contrast, the OLoRA-Full model—which imposes constraints across all LoRA modules—demonstrates a clear degradation in output quality.

SoLoRA stands out for its balanced approach between orthogonal regularization and representational flexibility. Despite applying orthogonal loss to important modules such as q\_proj and v\_proj, the model maintains strong performance across key metrics, including ROUGE-L (0.2549), BLEU-4 (0.0985), and BERTScore-F1 (0.1119), while achieving high linguistic diversity (Distinct-1 up to 0.4438). These results show that SoLoRA is capable of learning novel representations while preserving coherent, diverse responses and avoiding repetitive outputs.

In the context of internal task management—where queries tend to be diverse, context-specific, and span multiple departments—SoLoRA is a suitable solution. Its ability to understand and respond accurately to out-of-distribution (OOD) queries is essential for handling dynamic situations, such as task status updates, employee feedback, or inter-departmental data retrieval. Particularly, SoLoRA’s capacity to learn new representations without breaking contextual structure makes it well-suited for complex and evolving workflows.

Thus, the application of SoLoRA in task management not only ensures effective learning but also increases practical usability, offering an optimal solution for building internal chatbots that support enterprise operations.

Furthermore, this architecture enables modular expansion by training adapters for each task independently. For example, different functions like development and support can use specialized adapters (e.g., OLoRA) with task-specific datasets, while still sharing common modules or selectively applying constraints. This approach enhances adaptability to departmental needs while preserving generalization and knowledge sharing. It forms a solid foundation for scalable, multi-purpose internal AI systems tailored to real-world enterprise requirements.

# References

|  |  |
| --- | --- |
| [1] | Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, "LoRA: Low-Rank Adaptation of Large Language Models," 17 Jun 2021. [Online]. Available: https://arxiv.org/abs/2106.09685. [Accessed 12 05 2025]. |
| [2] | DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang,..100 additional authors not shown), "DeepSeek-V3 Technical Report," 2025 . |
| [3] | D. Kalajdzievski, "Scaling Laws for Forgetting When Fine-Tuning Large Language Models," 11 Jan arxiv. [Online]. Available: https://arxiv.org/abs/2401.05605. [Accessed 13 05 2025]. |
| [4] | Sze Wai Yuen, Liam Li, Kevin Musgrave, "Finding the best LoRA parameters," 11 Sep 2024. [Online]. Available: https://www.determined.ai/blog/lora-parameters. [Accessed 7 Jun 2025]. |
| [5] | H. Face, "Seq2Seq Parameters," [Online]. Available: https://huggingface.co/docs/autotrain/en/seq2seq\_params. [Accessed 07 Jun 2025]. |
| [6] | S. Raschka, "Practical Tips for Finetuning LLMs Using LoRA (Low-Rank Adaptation)," 19 Nov 2023. [Online]. Available: https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms. [Accessed 08 Jun 2025]. |
| [7] | D. Kalajdzievski, "A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA," 28 Nov 2023. [Online]. Available: https://arxiv.org/abs/2312.03732. [Accessed 08 Jun 2025]. |
| [8] | Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, Xuanjing Huang, "Orthogonal Subspace Learning for Language Model Continual Learning," 22 Oct 2023. [Online]. Available: https://arxiv.org/abs/2310.14152. [Accessed 08 Jun 2025]. |
| [9] | Quan Cheng, Yuanyu Wan, Lingyu Wu, Chenping Hou, Lijun Zhang, "Continuous Subspace Optimization for Continual Learning," 17 May 2025. [Online]. Available: https://arxiv.org/abs/2505.11816. [Accessed 17 Jun 2025]. |
| [10] | Ruiyu Wang, Sen Wang, Xinxin Zuo, Qiang Sun, "Lifelong Learning with Task-Specific Adaptation: Addressing the Stability-Plasticity Dilemma," 08 Mar 2025. [Online]. Available: https://arxiv.org/abs/2503.06213. [Accessed 18 Jun 2025]. |
| [11] | Xinyu Ma, Xu Chu, Zhibang Yang, Yang Lin, Xin Gao, Junfeng Zhao, "Parameter Efficient Quasi-Orthogonal Fine-Tuning via Givens Rotation," 05 Apr 2024. [Online]. Available: https://arxiv.org/abs/2404.04316. [Accessed 05 Jul 2025]. |