**DeepSeek-based Chatbot**

**System Supports Work Management**

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.

To address these challenges while maintaining model stability and efficient use of resources, this paper proposes the application of LoRA (Low-Rank Adaptation) during the fine-tuning of the DeepSeek model. LoRA introduced by [1], allows for a significant reduction in trainable parameters by inserting low-rank matrices into the pre-trained model architecture. As a result, fine-tuning can be conducted with memory and compute costs tens of times lower than full model tuning, while still maintaining high performance. Since the original weights remain untouched, LoRA-fine-tuned models retain foundational knowledge, thereby mitigating the effect of catastrophic forgetting [2].

LoRA also provides high flexibility for enterprise deployment: only the added weight components (known as adapters) need to be stored, rather than the full fine-tuned model. This reduces storage costs and simplifies multi-version deployment across departments. In this project, training data is sourced from platforms like Jira, along with internal log files documenting work progress, employee feedback, and task histories. This data is processed and structured into a standard input format for the model, enabling the training of LoRA adapters for specific tasks: generating consolidated work reports, suggesting role-specific tasks, and guiding new employees on how to handle assigned tasks.

Integrating the fine-tuned DeepSeek model into a work management system not only automates many critical processes, but also acts as an “internal assistant” capable of understanding context, recommending actions, and supporting real-time decision-making. This represents a crucial advancement in enhancing operational capacity, especially as businesses face the growing pressures of digital transformation and resource optimization.

* 1. Literature Review

In recent years, the application of large language models (LLMs) in enterprise support systems has gained significant attention. Models such as GPT, LLaMA, and more recently, DeepSeek, have demonstrated strong potential in processing natural language both flexibly and accurately. These capabilities open new directions for developing intelligent systems such as virtual work assistants, report summarization tools, and smart task suggestion engines. However, to deploy such models effectively in specific enterprise environments, fine-tuning on internal data becomes a critical requirement.

Beyond computational efficiency, another major issue in fine-tuning is catastrophic forgetting, which occurs when a model is fine-tuned multiple times or continuously updated over time. Study [2] explores the relationship between the number of training steps and the degradation of foundational knowledge, showing that even with techniques like LoRA, if there is no proper strategy for knowledge management, the model may still suffer from severe loss of pre-trained knowledge. Reference [6] provides a comprehensive survey of continual learning methods for LLMs, including rehearsal, regularization, parameter isolation, and hybrid approaches such as adapters or LoRA, all aiming to maintain long-term model performance.

One cost-optimization method is proposed in [1], introducing the LoRA (Low-Rank Adaptation) technique. This method preserves all the original weights of the base model and only trains two low-rank matrices within the attention layers, significantly reducing memory and computational costs during fine-tuning. LoRA has been shown to achieve performance comparable to full fine-tuning in various NLP tasks, while also reducing the risk of erasing previously learned foundational knowledge.

However ,studies [13] [14] [15] [16] 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.

Among open-source LLMs, DeepSeek is a high-potential model family designed to support the research community in deploying customizable models efficiently. In their latest technical report, the DeepSeek team[4] trained models ranging from 1.3B to 67B parameters using high-quality multilingual datasets, optimized for logical reasoning tasks. DeepSeek-R1, their fine-tuned reasoning model, has achieved results on par with commercial models like OpenAI GPT-3.5 in many multi-step reasoning tasks [7]. Importantly, DeepSeek is released under a fully open-source license, making it a practical choice for enterprises that cannot afford access to proprietary commercial models.

Experimental documentation has also demonstrated the applicability of DeepSeek to domain-specific tasks. For example, in a tutorial published by DataCamp [8], the authors fine-tuned DeepSeek-R1 Distill (8B) using LoRA to build a medical chatbot capable of chain-of-thought reasoning. Although not directly related to work management, the data preprocessing and fine-tuning strategies used in this study are highly transferable to similar tasks such as report generation, progress analysis, or task recommendation in enterprise settings.

Existing works have laid an essential foundation for integrating LLMs into enterprise assistant systems. However, there remains a lack of research that concretely addresses work management problems, which require integration with internal data (e.g., Jira, chat logs), maintaining knowledge stability, and operating efficiently in resource-constrained environments. This paper builds on the established approaches and extends them by combining DeepSeek, various LoRA-based techniques, and real-world enterprise task data to construct a work management support system tailored to the needs of modern organizations.

* 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.

To ensure efficiency, compactness, and model quality, orthogonalization is applied only to the key attention modules, namely q\_proj (query projection) and v\_proj (value projection). These are core components in the attention mechanism, playing a crucial role in determining query direction and output representation. Applying orthogonalization at these points ensures that new adapters learn completely novel and independent representations from the original model, while preserving the original attention structure.

Additionally, the model enforces that vectors within the same LoRA adapter are learned in mutually independent directions, through an internal orthogonality constraint applied to the row vectors of the low-rank weight matrix A.

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

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

In the first tier, data is collected from three main sources:

* Internal documentation and training manuals
* Work management systems such as Jira or Trello
* Survey and interview data from internal personnel

Once collected, the data is standardized into question–answer pairs following an instruction-tuning format, which is suitable for training large language models (LLMs).

In the model training tier, the system applies the LoRA technique. The base model used is DeepSeek-R1-Distill-Qwen-1.5B, a lightweight and high-performance LLM with Vietnamese language support.

Training is conducted in multiple rounds, where each round adds a new LoRA adapter to learn the residual information not yet captured by previous adapters. During each round, the system computes an orthogonal loss and adds it to the total loss, ensuring that the newly learned directions (vectors) do not overlap with previous ones. This helps prevent catastrophic forgetting and enhances the model's ability to explore diverse learning directions.

In the inference tier, the fine-tuned model is stored and deployed via API using the Hugging Face Transformers platform. This API allows internal systems to query the chatbot and receive contextual responses. Additionally, it stores conversation history, which enhances the chatbot’s understanding of user context in future interactions.

To maintain high performance and context relevance, the system limits each inference prompt to the most recent 5–10 conversation turns, ensuring that prompt length remains efficient while retaining sufficient contextual information.

* 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
* Data obtained from customer support groups

Data from work management systems typically has a high level of cleanliness, requiring only minimal preprocessing steps such as:

* Removing intermediary or redundant log entries
* Adjusting pronoun usage and role-specific references to fit the model's format

Once cleaned, this data can be directly used for AI training.

However, in organizations with multiple levels of management or communication, the data may include noisy comments from intermediate layers, which can disrupt the clarity of conversation flow — as illustrated in Fig 2.

A screenshot of a chat

AI-generated content may be incorrect.

**Fig. 1.** Data taken from Jira

Comments from intermediate management levels often contain repetitive phrases or are system-generated with template-like structures. Common patterns include expressions such as "nhờ.\*hỗ trợ" ("please assist") or "xử lý.\*sớm" ("handle promptly"), which can account for more than half of the comment content. These patterns can be filtered effectively using query-based or regex-based filtering techniques.

For image-based data, although this paper does not cover specific implementation details, there are established techniques to convert images into text form. For example:

* Structured forms or entry forms can be extracted using tools like Tesseract + Layout parsing or LayoutLM, especially when the structure needs to be preserved.
* For images showing error messages or system displays, Tesseract OCR alone is sufficient to extract usable textual content.

A major limitation of data from work management systems is that many operational flows occur outside of Jira. For example, key information may be embedded in code files, manual system operations, or external platforms—while Jira only reflects status updates or summaries. As a result, only around 20% of the total collected data can be directly transformed into usable training samples.

Beyond data from systems like Jira and Trello, companies often maintain Excel-based FAQs used during product or service onboarding—either for customers or internal training. These are typically high-quality data sources, as the questions are already curated and redundant ones are filtered out, and the answers are often very detailed.

However, one limitation of such data is its time-sensitivity—for example, questions like "When will feature A or B be released?" can quickly become outdated. To improve dataset quality, the system should be able to detect and clean time-specific questions (e.g., phrases like "when will it be available", "planned release date", etc.) before training.

As for official documents, such as internal manuals, regulatory documents, and government decrees, these are the most factually accurate sources, especially legal texts. However, the main challenge lies in the generalization requirement, as such content is not naturally formatted as work-related interactions.

A simple and effective solution is to use large language models (e.g., ChatGPT, Claude, or Gemini) to generate structured training data in JSONL format. Based on real-world experimentation, the recommended approach is:

* Let the model read the full document first to build a knowledge foundation.
* Then instruct it with a fixed prompt to generate question–answer pairs in JSONL, where each line includes exactly one instruction and one response.
* Limit each generation session to 25–50 pairs to ensure data quality.
* Focus on a specific section or theme within the document to keep context consistent.

As for internal chat group data, due to privacy and security concerns, this paper does not include or analyze data from these sources.

* 1. Data Formatting for Instruction-Tuning

To Once the question–answer pairs are collected, the data needs to be formatted for training. This paper proposes using the Hugging Face Chat Template format [17]. This format supports multi-turn conversational data, commonly used in assistant-style LLMs. Data is structured as a list of {role, content} pairs, and the chat\_template function defined in the tokenizer is responsible for converting this list into a training prompt. The format helps the model learn natural conversation structures, better context retention, and supports tasks like question answering, data explanation, and function calling.

This format is currently adopted by models such as LLaMA2-chat, Mistral, and DeepSeek, and is natively supported by the Hugging Face Transformers library for 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 collected from work management systems, the task creator is assigned the "user" role, as they typically initiate the request or inquiry. Other participants are labeled as "assistant". Since these dialogues often involve multiple turns, consecutive messages by the same person are merged into a single message. This helps maintain coherent reasoning and allows the model to infer responses in a structured, user–assistant pattern, reducing confusion during training.

In one dataset of 1,000 samples, consisting of 300 from Jira and 700 from Excel-based FAQs gathered during software deployment, the cosine similarity between each sample and the rest was computed. The results showed a left-skewed distribution, with most similarity scores ranging from 0.04 to 0.10, indicating a high degree of diversity across samples and reducing the risk of overfitting. Only ~5–10% of samples had a similarity score above 0.12, the threshold where overlap or paraphrasing might begin to affect training.

A graph of a number of cosine similar to others

AI-generated content may be incorrect.

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

In contrast, data generated by ChatGPT, while structured according to a predefined prompt, exhibited more redundancy. Over 120 pairs had cosine similarity scores above 0.90, implying significant overlap or rewording of content. This is attributed to the model’s limited understanding of internal processes, domain-specific terminology, and operational context, which results in surface-level linguistic mimicry rather than genuine expertise. Despite this, ChatGPT-generated data remains valid for use in PEFT fine-tuning settings such as LoRA or CoLA, especially when used to enhance generalization.

A graph of a number of bars

AI-generated content may be incorrect.

**Fig. 3.** Data generate from Chat GPT

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.

* 1. Modeling Module

The base model used in this work is DeepSeek-R1-Distill-Qwen-1.5B, fine-tuned using LoRA combined with orthogonality constraints according to the following pipeline:

* Tokenization & Data Formatting: The training data-comprising both real and synthetic samples-was preprocessed and normalized into a chat format compatible with the Qwen model. It was then tokenized using the official tokenizer from DeepSeek.
* Pre-training Preparation: Before the fine-tuning process begins, the original weight matrices A from the backbone model-corresponding to the q\_proj, v\_proj, and k\_proj modules-are reloaded and stored as fixed tensors (via detach). These frozen matrices are excluded from gradient backpropagation and are used solely for computing external orthogonality constraints with respect to the current adapter.
* Training Procedure:
  + During training, each A matrix in the LoRA adapters is enforced to satisfy internal orthogonality, by minimizing the error between AATAAT and the identity matrix II. This constraint compels the learned directions within the same adapter to be mutually independent, thus reducing information redundancy.
  + Simultaneously, the newly learned A matrices must also be orthogonal to the corresponding frozen Amatrices from the backbone model (external orthogonality). This ensures that the adapter learns novel directions rather than duplicating existing ones, facilitating better generalization.
* Parameter Control Across Rounds: Adjustment of parameters across training rounds is a critical factor influencing the model's learning performance. In addition to standard hyperparameters like rank, learning rate, and number of epochs, two specific coefficients are introduced to regulate the strength of orthogonality constraints within LoRA modules:
  + Lambda\_internal controls the degree of orthogonality between row vectors within the same A matrix (i.e., intra-module orthogonality). Higher values enforce stronger internal orthogonality, promoting diversity in learned representations.
  + Lambda\_external governs the orthogonality between the current A matrix and those from previous adapters (i.e., inter-round orthogonality). This constraint helps ensure that new representations do not overlap with old ones, enhancing the model's capacity to generalize across tasks.
* Targeted Modules for Orthogonal Constraints: Orthogonalization is selectively applied to the following LoRA target modules:
  + q\_proj: Since this module governs what a token "asks" or attends to, enforcing orthogonality is essential to encourage novel queries.
  + k\_proj: As it produces the keys used in attention weighting, orthogonalization helps prevent overlapping key vectors.
  + v\_proj: Responsible for generating value vectors that are aggregated during attention. Orthogonalization here encourages the model to learn diverse values and preserve information richness.

In summary, fine-tuning the DeepSeek-R1 model using LoRA combined with orthogonality constraints effectively balances representation diversity and generalization, while remaining computationally efficient.

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 Circular 70. 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 data primarily focuses on two departments:

* Developer team: This subset includes materials related to the usage of the system’s base code for extending functionality, as well as documentation intended for third-party integration with the electronic invoice system.
* IT Help Desk: This subset focuses on customer support and procedural guidance regarding invoice management in accordance with Circular 70.

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

|  |  |  |
| --- | --- | --- |
| Training Round | Model | Orthogonal Constraint Modules |
| Round 1 | Lora | None |
| Round 1 | OLora | q\_proj, v\_proj |
| Round 1 | SoLora | q\_proj, v\_proj |
| Round 2 | Lora | None |
| Round 2 | OLora | q\_proj, v\_proj, k\_proj |
| Round 2 | OLora All | q\_proj, v\_proj, k\_proj, o\_proj, gate\_proj |
| Round 2 | SoLora | q\_proj, v\_proj |

To ensure feasibility and cost-efficiency, the training was conducted on Google Colab Pro, which provides access to dedicated GPUs. Thanks to the GPU acceleration and flexible scalability offered by Colab, the DeepSeek-R1-Distill-Qwen-1.5B model could be effectively fine-tuned with appropriate batch sizes and token lengths.

Environment Configuration:

Hardware:

* Platform: Google Colab Pro
* GPU: NVIDIA Tesla T4 or equivalent
* GPU RAM: 15.0 GB (fully available before training)
* System RAM: 51.0 GB

Software:

* Python: Version 3.x (Colab default)
* PyTorch: >= 2.1 with CUDA support
* Transformers: v4.41 or later
* PEFT: v0.9 (or compatible version)

Optimization Settings:

* fp16: Enabled (fp16=True) to leverage GPU acceleration
* Batch Size: Kept small (e.g., 1-2) to stay within the 15 GB GPU VRAM limit
* Token Length: Limited to under 128 tokens per sample to reduce memory load.
  1. Result

Evaluating During training, the three models—LoRA, OLoRA, and SoLoRA—exhibited distinct learning dynamics, reflecting the optimization strategies employed by each method.

A graph with red line

AI-generated content may be incorrect.

**Fig. 4.** Training Loss Comparison

Figure 5. Training Loss Comparison

A graph with orange and pink lines

AI-generated content may be incorrect.

**Fig. 5.** Gradient Norm Over Epochs

The LoRA model began with a relatively low initial loss of 7.8713, significantly lower than the other two. Within less than one epoch, its loss rapidly dropped to around 6.25, indicating extremely fast convergence when no structural learning constraints are applied. However, this rapid descent also reflects a lack of control over the parameter space, potentially leading to overfitting, bias, or catastrophic forgetting in subsequent tasks.

In contrast, OLoRA, which imposes orthogonality constraints on the A matrices within LoRA modules, started with a much higher initial loss of 181.01, as the model was restricted to learning within a narrower subspace. Nevertheless, the loss steadily decreased and reached 6.58 in later stages—comparable to LoRA—while maintaining greater stability. This indicates that orthogonal regularization does not hinder learning; instead, it helps the model avoid redundant parameter directions, preserving task-specific representations from prior learning.

SoLoRA, which combines orthogonalization with inter-adapter collaboration, displayed the most complex learning behavior. Its initial loss peaked at 204.4—the highest among the three—suggesting strong suppression from overlapping constraints. However, notably, SoLoRA’s loss decreased the fastest, reaching 5.92 after just one epoch—lower than both LoRA and OLoRA. This shows that combining collaborative learning and orthogonality not only preserves task separation but also leverages shared knowledge effectively.

In terms of gradient norm, LoRA started at approximately 17.26 and gradually decreased to about 4.5, indicating stable convergence. OLoRA began even higher at 18.42, but similarly exhibited consistent reduction in gradient magnitude, reflecting controlled and stable learning. Meanwhile, SoLoRA had the highest initial gradient norm of 21.31, indicating intense early learning to overcome complex constraints, but eventually converged to similar levels as the other models after 30 epochs—demonstrating high learning efficiency and convergence strength.

The comparison of average accuracy among the LoRA-based training methods revealed that OLoRA (with orthogonalization applied to q\_proj, k\_proj, and v\_proj) achieved the highest mean accuracy of 0.831. This suggests that lightweight orthogonal constraints on critical learning directions enable the model to retain more distinctive task-specific information.

SoLoRA, which adds internal orthogonality among the vectors within A, also performed well with an accuracy of 0.812, highlighting the benefits of refining internal representation structure. In contrast, standard LoRA (without orthogonal constraints) achieved a lower average accuracy of 0.796.

Notably, OLoRA All—which extends orthogonal constraints to o\_proj and gate\_proj—had the lowest performance (accuracy 0.684), indicating that applying orthogonality to modules with limited representational function may introduce noise and negatively affect learning outcomes.

After training, models were evaluated on the original test set using cosine similarity, a common technique in NLP for measuring semantic similarity between two vectors—often used to assess sentence or document-level closeness.

* OLoRA achieved the highest mean cosine similarity (0.8139), indicating that orthogonalization with respect to the base matrix helped preserve prior knowledge while enabling effective learning of new information.
* SoLoRA had the lowest minimum value (0.13) and the largest range (0.87), suggesting that while internal orthogonality encourages diversity in learning directions, it may compromise stability on some samples.
* LoRA exhibited relatively stable but unremarkable results, lacking any mechanism to prevent learning in previously used directions—thereby risking catastrophic forgetting.

**Table 4.** Testing Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Mean | Median | Max | Min | Range |
| LoRA | 0.7866 | 0.82 | 1.0 | 0.22 | 0.78 |
| OLoRA | 0.8139 | 0.84 | 1.0 | 0.33 | 0.67 |
| SoLoRA | 0.7837 | 0.81 | 1.0 | 0.13 | 0.87 |

When tested on a paraphrased dataset—specifically, a set of 20 semantically varied questions related to ERR codes from 1 to 50—SoLoRA (formerly referred to as CoLoRA) demonstrated superior semantic understanding:

* SoLoRA correctly answered 18/20 questions.
* LoRA answered 3/20.
* OLoRA answered 5/20.

Although SoLoRA had a slightly lower average cosine similarity than OLoRA on the original test set, its ability to generalize and interpret rephrased or linguistically diverse queries far surpassed the others. This suggests that sacrificing a small degree of semantic accuracy for greater interpretive flexibility and robustness is a worthwhile trade-off—especially in real-world NLP systems like technical support chatbots or intelligent assistants, where non-standard language is common.

A graph of a number of blue bars

AI-generated content may be incorrect.

**Fig. 5.** Comparison Of Average Accuracy Across LoRA-Based Training Methods

In the second training round, results revealed that adding k\_proj to the orthogonalized modules (in addition to q\_proj and v\_proj) did not significantly alter performance. However, enforcing orthogonality on all modules caused a noticeable drop in quality and increased training difficulty. For instance, OLoRA All answered only 1 question outside the training distribution, while SoLoRA correctly answered 17. This highlights the importance of selectively choosing modules for orthogonalization. Applying constraints indiscriminately can lead to severe degradation in both model quality and generalization ability.

* 1. Conclusion And Future Work

The results from both training rounds highlight the critical importance of strategically selecting which modules should undergo orthogonalization. Specifically, enforcing orthogonality on q\_proj and v\_proj significantly improves model performance. In contrast, applying orthogonality constraints across all LoRA modules—as in OLoRA-Full—leads to notable performance degradation, both in terms of average accuracy and generalization ability when evaluated on novel, out-of-distribution questions. With only 1 out of 20 questions answered correctly, OLoRA-Full demonstrates that over-constraining the model severely limits its flexibility and representational capacity.

Meanwhile, SoLoRA consistently demonstrates effectiveness by maintaining a balance between orthogonal constraints and collaborative knowledge sharing. It achieves 17–18 correct answers out of 20 on the extended evaluation set, validating its ability to generalize while preserving task-specific representations. These findings reinforce the conclusion that orthogonalization should be applied selectively, guiding the model toward learning novel directions while avoiding excessive rigidity that could hinder adaptation to diverse, unseen data.

Moreover, this architecture enables task-level modularization, where different tasks such as dev and support can be trained on separate datasets, each with their own adapter (e.g., using OLoRA), while still leveraging selective orthogonal constraints. This setup allows tasks to specialize more effectively without sacrificing the ability to share transferable knowledge between tasks when necessary.

# 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] | Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, Hao Wang, "Continual Learning of Large Language Models: A Comprehensive Survey," 25 Apr 2024 . [Online]. Available: https://arxiv.org/abs/2404.16789. [Accessed 25 May 2025]. |
| [5] | 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]. |
| [6] | 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]. |
| [7] | 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]. |
| [8] | 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]. |
| [9] | DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi. (100 additional authors not shown), "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," 22 Jan 2025. [Online]. Available: https://arxiv.org/abs/2501.12948. [Accessed 25 May 2025]. |
| [10] | "Fine-Tuning DeepSeek R1 (Reasoning Model)," 27 Jan 2025. [Online]. Available: https://www.datacamp.com/tutorial/fine-tuning-deepseek-r1-reasoning-model. [Accessed 25 May 2025]. |
| [11] | Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," 23 May 2023. [Online]. Available: https://arxiv.org/abs/2305.14314. [Accessed 25 May 2025]. |
| [12] | H. Face, "Chat Templates," [Online]. Available: https://huggingface.co/learn/llm-course/chapter11/2. [Accessed 23 Jun 2025]. |
| [13] | Hengzhi He, Shirong Xu, Guang Cheng, "Golden Ratio Weighting Prevents Model Collapse," 25 Feb 2025. [Online]. Available: https://www.arxiv.org/abs/2502.18049. [Accessed 24 Jun 2025]. |