**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

DeepSeek-based Chatbot System Supports Work Management

by

Dinh Cong Bang

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

© Copyright by Dinh Cong Bang 2025

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

DeepSeek-based Chatbot System Supports Work Management

by

Dinh Cong Bang

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

Supervisor:

Assoc. Prof. Phan Duy Hung

Dr. Vu Thu Diep

© Copyright by Dinh Cong Bang 2025

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. LoRA keeps the original weights frozen and only trains a 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.

Acknowledgments

I would like to express my deepest gratitude to Assoc. Prof. Phan Duy Hung and Dr. Vu Thu Diep, who guided me from the very beginning in shaping the topic and setting the research direction. This work would not have been possible without his dedicated and persistent support.

I also sincerely thank my colleagues who supported me in providing data and infrastructure for AI training.

Finally, I would like to extend my heartfelt appreciation to my family – those who have always been by my side, loving and supporting me unconditionally throughout this research journey.

# List of Figures

[**Figure 1.** Overall architecture 18](#_Toc202832451)

[**Figure 2.** Data taken from Jira 20](#_Toc202832452)

[**Figure 3.** Data taken from Jira 23](#_Toc202832453)

[**Figure 4.** Data generate from Chat GPT 23](#_Toc202832454)

[**Figure 5.** Training Loss Comparison 29](#_Toc202832455)

[**Figure 6.** Gradient Norm Over Epochs 30](#_Toc202832456)

[**Figure 7.** Comparison Of Average Accuracy Across LoRA-Based Training Methods 32](#_Toc202832457)

# List of Table

[**Table 1.** Data in Hugging Face Chat Template format 22](#_Toc202832459)

[**Table 2.** Average token length and quantity statistics by data source 26](#_Toc202832460)

[**Table 3.** Orthogonal extrusion module 27](#_Toc202832461)

[**Table 4.** Testing Result 31](#_Toc202832462)

Table of Contents

[Acknowledgments 2](#_Toc202832465)

[List of Figures 3](#_Toc202832466)

[List of Table 4](#_Toc202832467)

[Table of Contents 5](#_Toc202832468)

[Chapter 1 Introduction 6](#_Toc202832469)

[1.1. Problem And Motivation 6](#_Toc202832470)

[1.2. Related Works 7](#_Toc202832471)

[1.3. Contribution 9](#_Toc202832472)

[Chapter 2 Background Study 11](#_Toc202832473)

[2.1. Business Workflow in Enterprises 11](#_Toc202832474)

[2.2. DeepSeek model 12](#_Toc202832475)

[2.3. Low-Rank Adaptation 13](#_Toc202832476)

[2.4. Orthogonal in Low-Rank Adaptation 14](#_Toc202832477)

[Chapter 3 System Design 18](#_Toc202832478)

[3.1. System Architecture 18](#_Toc202832479)

[3.2. Data Preprocessing 19](#_Toc202832480)

[3.2.1. Data extraction 19](#_Toc202832481)

[3.2.2. Data Formatting for Instruction-Tuning 21](#_Toc202832482)

[3.3. Modeling Module 24](#_Toc202832483)

[Chapter 4 Experiments And Results 26](#_Toc202832484)

[4.1. Data collection 26](#_Toc202832485)

[4.2. Experiments 27](#_Toc202832486)

[4.3. Results 28](#_Toc202832487)

[4.4. Conclusion And Future Work 32](#_Toc202832488)

[References 34](#_Toc202832489)

# Introduction

## Problem And 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 [2]. 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 [3].

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 [3].

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.

## Related Works

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.

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 [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.

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 [2] 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 [9]. 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 [10], 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.

## 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.

# Background Study

To successfully design and implement an AI-powered work management system tailored to the specific needs of an enterprise, it is essential to establish a solid knowledge foundation—both in terms of existing operational workflows and the latest technical advancements in large language models (LLMs).

First, the structure and characteristics of common work management methodologies will be analyzed to identify integration points where AI models can be effectively embedded. Following that, core technical components such as the DeepSeek model, LoRA (Low-Rank Adaptation) fine-tuning technique, and multi-stage training strategies will be explored. These components form the basis for selecting appropriate technologies to build a system that is efficient, lightweight, and easily adaptable to real-world enterprise environments.

## Business Workflow in Enterprises

In modern enterprise environments, effective work management plays a crucial role in ensuring productivity and collaboration across departments. Organizations often deploy Workflow Management Systems (WfMS) such as Jira or Trello to automate and monitor complex operational processes. This makes it feasible to collect communication and coordination data between different teams.

However, extracting data from workflow systems introduces a key challenge. Tasks are often related to the company’s products or services, but at the same time, the internal communication and contextual understanding of those tasks vary across departments or units. As a result, the data collected tends to have both similarities and differences in the details of each task.

For instance, multiple teams may be referring to the same software using shared domain-specific terminology, yet with different perspectives: the BA (Business Analyst) team focuses on user interaction and functional usage, while the Dev (Developer) team requires in-depth understanding of system functionalities at the code level. This highlights a key requirement for the model — it must be able to precisely capture the shared references when different departments refer to the same feature, while also distinguishing the specific functional needs unique to each department.

## DeepSeek model

The DeepSeek-R1-Distill-Qwen-1.5B is a large language model (LLM) with 1.5 billion parameters, developed based on the Qwen architecture and fine-tuned from the original DeepSeek-R1 version. The model is designed to deliver high performance in logical reasoning, programming, and mathematical tasks, while being optimized for deployment on resource-constrained systems.

It adopts a decoder-only Transformer architecture with the following technical specifications:

* Number of layers (n\_layers): 24
* Model dimension (d\_model): 2048
* Intermediate layer size (d\_intermediate): 5504
* Number of attention heads (n\_heads): 16
* Number of key-value heads (n\_kv\_heads): 16
* Vocabulary size: 102,400 (byte-level BPE)
* Maximum context length: 4096 tokens

The DeepSeek-R1-Distill-Qwen-1.5B is a large language model (LLM) with 1.5 billion parameters, developed based on the Qwen architecture and fine-tuned from the original DeepSeek-R1 version. The model is designed to deliver high performance in logical reasoning, programming, and mathematical tasks, while being optimized for deployment on resource-constrained systems [11]:

* Recommended GPU: NVIDIA RTX 3070 or equivalent, with at least 8GB of VRAM
* Actual VRAM usage: Approximately 3.3GB
* Quantization support: Can be reduced to ~4GB VRAM with 4-bit quantization

These optimizations make it feasible for businesses to deploy the model on standard workstations or even on properly configured personal computers.

DeepSeek-R1-Distill-Qwen-1.5B has received high ratings in reasoning and programming evaluations, achieving performance comparable to much larger models such as Qwen3-235B-thinking. The model leverages chain-of-thought techniques from DeepSeek-R1 to enhance its reasoning ability, while maintaining strong efficiency and relatively low computational requirements.

With its balance between performance and deployability, DeepSeek-R1-Distill-Qwen-1.5B is a highly suitable choice for enterprise applications that require strong reasoning capabilities in a compact and efficient LLM.

## Low-Rank Adaptation

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 [1]. In contrast, LoRA keeps the original weights frozen and inserts additional low-rank matrices, significantly reducing both computation and memory costs during training.

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.

Matrix A projects the input into a lower-dimensional subspace, capturing new learning directions of the original weight W. Matrix B expands this compressed representation back to the output space. As a result, all new information learned during fine-tuning is encoded into ΔW with very low parameter overhead.

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.

However, [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. Kalajdzievski demonstrated that rsLoRA not only improves benchmark task performance, but also reduces computational resource requirements when fine-tuning large language models [15].

Using low-rank matrices allows LoRA-based models to achieve fine-tuning performance comparable to traditional methods, while requiring only 0.1% to 1% of the trainable parameters compared to full fine-tuning [1]. This makes LoRA especially useful in resource-constrained environments, such as training on CPUs or low-memory GPUs, enabling small and medium-sized businesses to train their own AI models affordably and effectively.

## Orthogonal in Low-Rank Adaptation

Orthogonalization in fine-tuning Large Language Models (LLMs) is a technique that forces the rows or columns of a matrix to be orthogonal to each other and normalized to unit length. This constraint compels the model to learn new directions instead of reusing existing ones, helping to prevent catastrophic forgetting [5] [6].

Studies such as Orthogonal Low-Rank Adaptation (O-LoRA) [5] propose splitting the data into separate tasks with distinct themes or objectives, and then training those tasks sequentially under the following principles:

* The matrix A for a new task must be softly orthogonal to the matrix A from previous tasks.
* The matrix A must also be orthogonal within itself, i.e., its row vectors should be mutually orthogonal.

By enforcing the learning directions (i.e., the rows of matrix A) of a new task to be orthogonal to those of previous tasks, O-LoRA ensures that the model does not overwrite previously learned knowledge, maintaining strong performance on older tasks without needing to revisit the original training data. Each task is thereby mapped into a distinct subspace, enabled by the orthogonal nature of the learned vectors. This allows the model to clearly distinguish between tasks, minimizing interference during multi-task training.

Moreover, because the learning space is constrained by orthogonality, the model is essentially forced to explore new representational directions, rather than overfitting by reusing previously adapted weights. This helps reduce the risk of overfitting, particularly for tasks with limited training data.

However, 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. This becomes especially problematic in scenarios where tasks are closely related or share significant feature overlap—in such cases, forcing the model to learn in entirely new directions may prevent it from leveraging efficient, reusable representations.

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.

Selecting which modules to apply orthogonalization to should be done selectively, based on their functional role, representational characteristics, and their impact on learning.

In the Transformer architecture, the key linear modules in the self-attention block include:

* q\_proj (query projection)
* k\_proj (key projection)
* v\_proj (value projection)
* o\_proj (output projection)

These modules are responsible for constructing and coordinating the attention mechanism between tokens in the input sequence. Specifically:

* q\_proj and k\_proj determine how a token evaluates the relevance of other tokens through queries and keys. If the learned vectors are aligned or lose distinctiveness, the attention mechanism may become too blurry to capture meaningful semantic differences.
* v\_proj generates the value vectors that carry extracted information. If these are contaminated by previously learned representations, the model may "forget" new representations it is supposed to learn.
* o\_proj aggregates all the information after attention computation and projects it back into the hidden space. Without orthogonal constraints, newly learned outputs might collapse into existing ones, reducing the model's expressiveness.

Research [16] shows that the projection layers in attention are the most prone to alignment issues after fine-tuning. Applying orthogonal constraints to these projections helps prevent overlapping representations, thereby enhancing the diversity of attention representations. Moreover, since these modules are typically linear and lack intermediate non-linearities (unlike MLP projections), applying orthogonalization here does not degrade the model's capacity for non-linear expression.

In contrast, the MLP components—such as gate\_proj, up\_proj, and down\_proj—exhibit different characteristics:

* They are coupled with non-linear activations like GELU,
* They undergo significant spatial transformations,
* They tend to encode highly non-linear content.
* Therefore, applying orthogonal constraints to MLP modules may eliminate critical non-linear features, negatively impacting the model’s deep representational capacity.

From this analysis, orthogonalization should be applied selectively, primarily to modules like q\_proj, k\_proj, v\_proj, and o\_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.

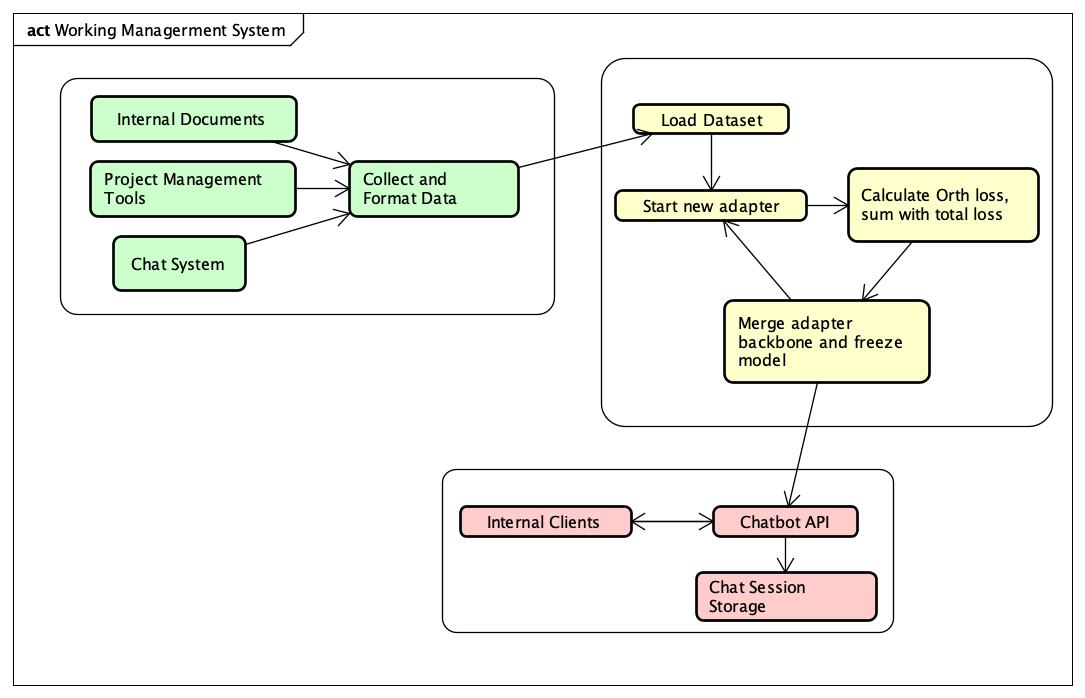
# System Design

## System 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.



**Figure 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.

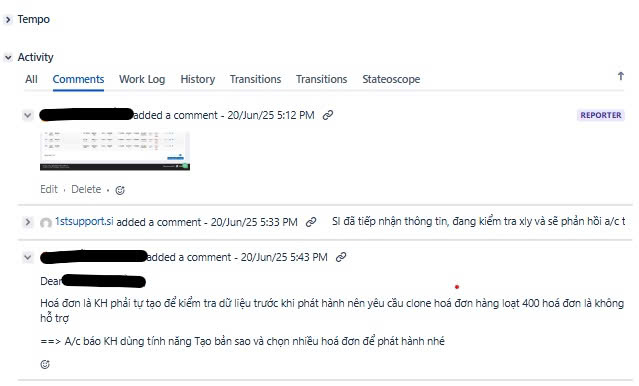
## Data Preprocessing

### 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 Figure 2) and adjusting pronoun usage—to convert it into training data suitable for AI models.



**Figure 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 or 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. However, a common drawback is the presence of time-sensitive information, such as “When will feature A be released?”. To address this, the system should identify and filter out questions with temporal intent, e.g., those starting with “when will”, “expected to launch”, etc., to maintain data quality.

As for instructional documents, such as internal manuals, official notices, and legal regulations, they represent the most accurate sources of knowledge. However, they require strong generalization capabilities and are not formatted in a dialogue structure. The simplest method is to use large language models such as ChatGPT, Claude, or Gemini to generate JSONL-style data. Based on practical experience, it is recommended to allow the model to pre-read the document for knowledge grounding. When prompting the model to generate data, use a fixed prompt with specific requirements: generate JSONL format, with each line containing one question–answer pair, focused on a defined section of the document. Each session should aim to produce 25–50 high-quality pairs to ensure consistency and depth.

### 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.

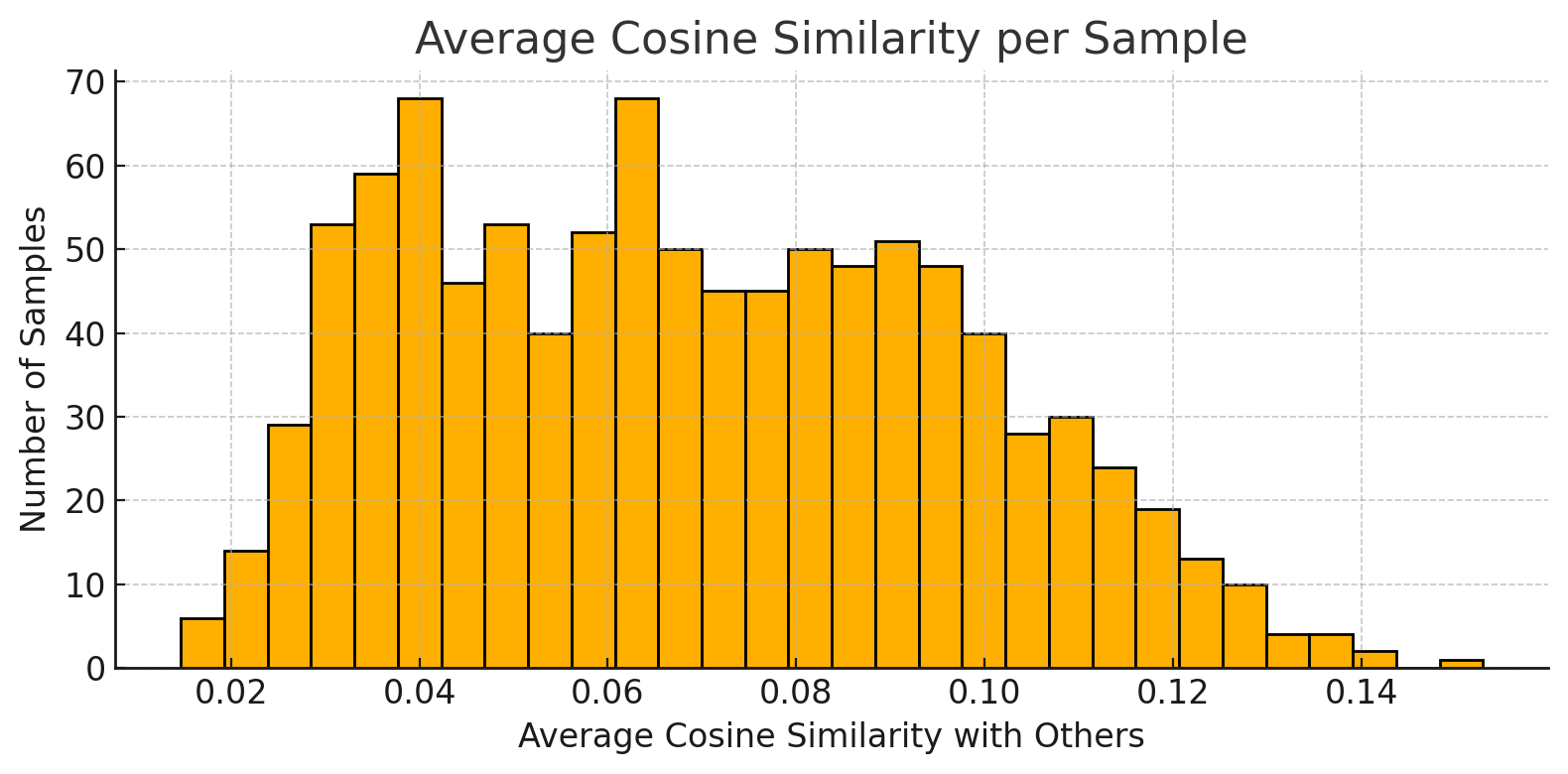
**Table 1.** Data in Hugging Face Chat Template format

|  |
| --- |
| {"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.

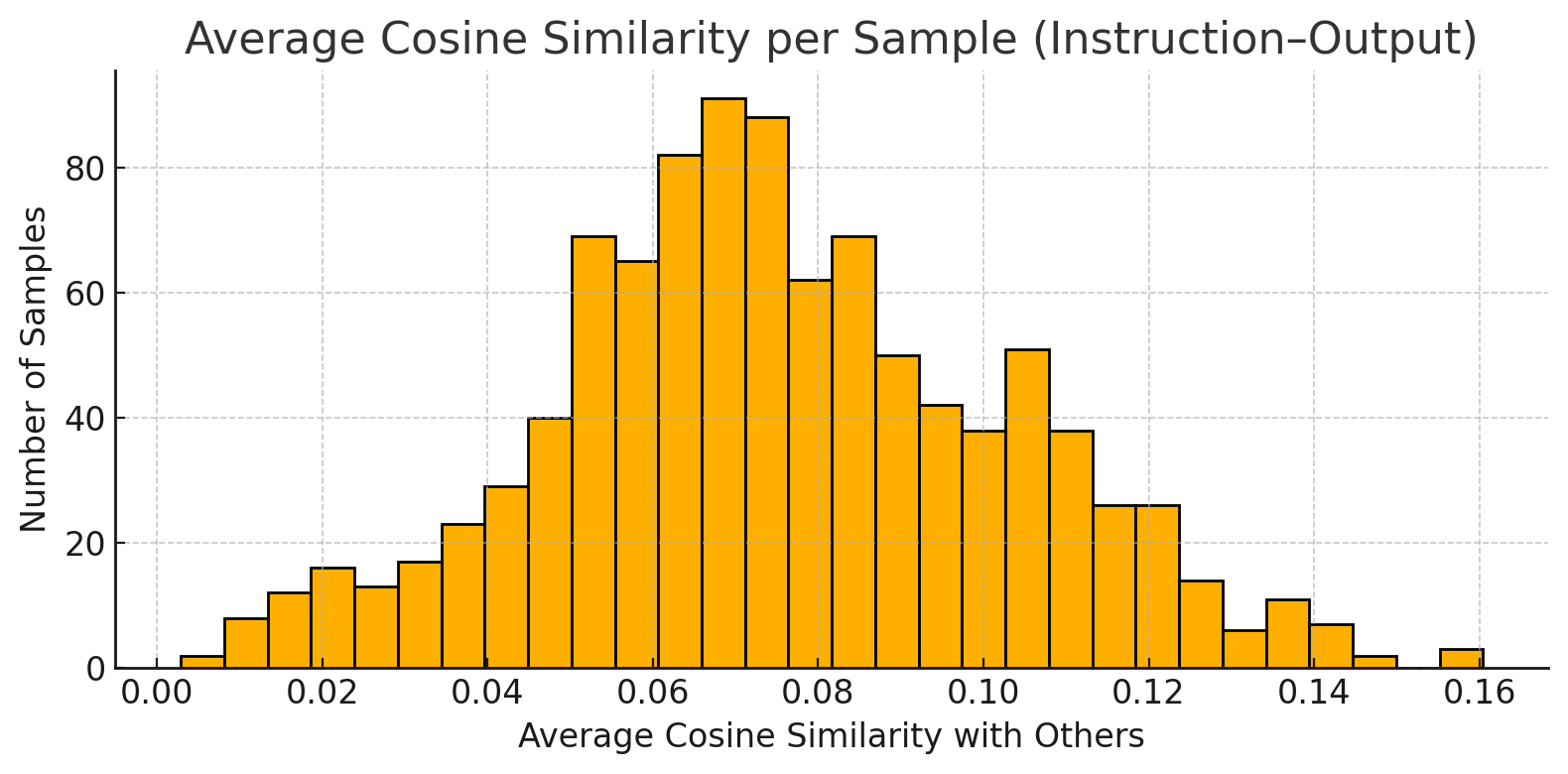
To ensure model diversity, the paper follows a study [18], showing that mixing real and generated data in the golden ratio (~61.8 : 38.2) preserves the accuracy of real samples while enhancing the model's expressive and generalization capabilities via linguistic diversity. In this context, with 1,000 real samples, approximately 618 generated samples are needed to reach a total of ~1,618 samples, ensuring diversity without compromising quality.

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.



**Figure 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.



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

## Modeling Module

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

# Experiments And Results

## Data collection

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 |

## Experiments

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

## Results

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

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.

A graph with red line

AI-generated content may be incorrect.

**Figure 5.** Training Loss Comparison

A graph with orange and pink lines

AI-generated content may be incorrect.

**Figure 6.** Gradient Norm Over Epochs

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.

**Figure 7.** 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.

## 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] | DeepSeek-AI, "DeepSeek-R1-Distill-Qwen-7B," [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B. |
| [12] | 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]. |
| [13] | H. Face, "Seq2Seq Parameters," [Online]. Available: https://huggingface.co/docs/autotrain/en/seq2seq\_params. [Accessed 07 Jun 2025]. |
| [14] | 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]. |
| [15] | 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]. |
| [16] | 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]. |
| [17] | H. Face, "Chat Templates," [Online]. Available: https://huggingface.co/learn/llm-course/chapter11/2. [Accessed 23 Jun 2025]. |
| [18] | 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]. |