**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

© Copyright by Dinh Cong Bang 2025

DeepSeek-based Chatbot system supports work management

Dinh Cong Bang

Degree Master of Software Engineering

FPT University

2025

Abstract

Trong bối cảnh chuyển đổi số hiện nay, việc kết hợp AI vào các hệ thống là một nhu cầu tất yếu. do đó luận văn này phát triển một hệ thống quản lý công việc nhằm tự động hóa việc tạo báo cáo, hướng dẫn người mới, tra cứu đầu mối. Hệ thống nâng cao năng suất thông qua việc tự động tạo báo cáo, đề xuất nhiệm vụ và hướng dẫn nhân viên mới. Mô hình AI được huấn luyện trên dữ liệu từ Jira và dữ liệu các nhóm chat nội bộ, giúp hệ thống nắm bắt ngữ cảnh thực tế và quy trình nghiệp vụ.

Cốt lõi của hệ thống là mô hình DeepSeek, được tinh chỉnh bằng kỹ thuật LoRA (Low-Rank Adaptation) kết hợp Multi-Stage Fine-Tuning. Kỹ thuật LoRA cố định trọng số gốc và chỉ huấn luyện thêm một lượng nhỏ trọng số hạng thấp, giúp giảm đến hàng nghìn lần số tham số cần cập nhật so với tinh chỉnh toàn bộ mô hình [1]. Mô hình sẽ được huấn luyện nhiều vòng, mỗi vòng các ma trận được ép trực giao với nhau tuỳ theo chiến lược và mục tiêu của từng vòng, qua đó tăng khả năng tiếp nhận và xử lý thông tin của mô hình.

Acknowledgments

Tôi xin bày tỏ lòng biết ơn sâu sắc tới PGS. TS. Phan Duy Hùng – người đã hướng dẫn tôi từ những bước đầu trong việc định hình đề tài và định hướng nghiên cứu. Công trình này sẽ không thể hoàn thành nếu thiếu đi sự hỗ trợ tận tình và bền bỉ từ thầy.

Tôi cũng xin gửi lời cảm ơn chân thành tới các đồng nghiệp của mình đã hỗ trợ tôi trong vấn đề cung cấp dữ liệu và hạ tầng training AI.

Cuối cùng, tôi xin dành lời tri ân sâu sắc đến gia đình – những người luôn bên cạnh, yêu thương và ủng hộ tôi vô điều kiện trong suốt chặng đường nghiên cứu này.

# List of Figures

[**Figure 1.** Overall architecture 16](#_Toc201755893)

[**Figure 2.** Data taken from Jira 18](#_Toc201755894)

[**Figure 3.** Data taken from Jira 21](#_Toc201755895)

[**Figure 4.** Data generate from Chat GPT 21](#_Toc201755896)

# List of Table

[**Table 1.** Data in Hugging Face Chat Template format 20](#_Toc201701301)

Table of Contents

[Acknowledgments 2](#_Toc202852253)

[List of Figures 3](#_Toc202852254)

[List of Table 4](#_Toc202852255)

[Table of Contents 5](#_Toc202852256)

[Chapter 1 Introduction 6](#_Toc202852257)

[1.1. Problem And Motivation 6](#_Toc202852258)

[1.2. Related Works 7](#_Toc202852259)

[1.3. Contribution 9](#_Toc202852260)

[Chapter 2 Background Study 10](#_Toc202852261)

[2.1. Business Workflow in Enterprises 10](#_Toc202852262)

[2.2. DeepSeek model 11](#_Toc202852263)

[2.3. Low-Rank Adaptation 12](#_Toc202852264)

[2.4. Orthogonal trong Low-Rank Adaptation 13](#_Toc202852265)

[Chapter 3 System Design 16](#_Toc202852266)

[3.1. System Architecture 16](#_Toc202852267)

[3.2. Data Preprocessing 17](#_Toc202852268)

[3.2.1. Data extraction 17](#_Toc202852269)

[3.2.2. Data Formatting for Instruction-Tuning 19](#_Toc202852270)

[3.3. Modeling Module 22](#_Toc202852271)

[Chapter 4 Experiments And Results 24](#_Toc202852272)

[4.1. Data collection 24](#_Toc202852273)

[4.2. Experiments 25](#_Toc202852274)

[4.3. Results 27](#_Toc202852275)

[4.4. Conclusion And Future Work 31](#_Toc202852276)

[References 32](#_Toc202852277)

# Introduction

## Problem And Motivation

Trong môi trường doanh nghiệp hiện tại, việc theo dõi, nhu cầu phân bổ và đánh giá tiến độ công việc hiệu quả ngày càng lớn, đặc biệt với sự gia tăng về quy mô dự án và độ phức tạp trong quy trình vận hành. Mặc dù nhiều công cụ như Jira hay Trello đã được áp dụng rộng rãi, chúng vẫn chủ yếu mang tính chất lưu trữ và theo dõi thủ công, đòi hỏi sự can thiệp thường xuyên từ người dùng. Điều này khiến quá trình tổng hợp báo cáo, đề xuất công việc hoặc hướng dẫn nhân viên mới trở nên tốn thời gian và thiếu nhất quán.

Song song với đó, sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn (LLMs) như ChatGPT, Grok, DeepSeek mở ra cơ hội mới trong việc tự động hóa các tác vụ quản lý công việc. DeepSeek là một dòng mô hình ngôn ngữ mã nguồn mở được huấn luyện trên tập dữ liệu quy mô hàng nghìn tỷ token, với năng lực suy luận được tăng cường thông qua các chiến lược huấn luyện nhiều giai đoạn [2]. Tuy nhiên, để tận dụng hiệu quả các mô hình này trong môi trường doanh nghiệp cụ thể, việc tinh chỉnh (fine-tuning) dựa trên dữ liệu nội bộ là điều thiết yếu. Điều này đặt ra thách thức về hiệu quả tài nguyên, khả năng mở rộng và nguy cơ làm mất đi tri thức nền của mô hình gốc nếu không được huấn luyện đúng cách [3].

Từ nhu cầu thực tiễn, bài viết này này tập trung phát triển một hệ thống trí tuệ nhân tạo hỗ trợ doanh nghiệp trong việc theo dõi, phân tích và tối ưu hóa hoạt động vận hành nội bộ. Hệ thống dựa trên DeepSeek, một trong những mô hình LLM mã nguồn mở nổi bật hiện nay, được huấn luyện trên dữ liệu quy mô lớn, sở hữu khả năng suy luận ngữ cảnh tốt và có hỗ trợ tiếng việt. Tuy nhiên, để khai thác hiệu quả mô hình trong môi trường doanh nghiệp cụ thể, nơi có ngôn ngữ nội bộ, thuật ngữ chuyên ngành và đặc thù quy trình riêng biệt, việc fine-tuning là yếu tố bắt buộc. Fine-tune không chỉ giúp thích nghi với dữ liệu đặc thù của tổ chức mà còn mở ra khả năng tự động hóa hàng loạt quy trình như: tạo báo cáo tiến độ công việc, đề xuất nhiệm vụ phù hợp theo bối cảnh và định hướng nhân viên mới thông qua các hướng dẫn được cá nhân hóa.

Để giải quyết các bài toán trên mà vẫn đảm bảo hiệu quả tài nguyên và độ ổn định mô hình, bài viết đề xuất kỹ thuật LoRA (Low-Rank Adaptation) trong quá trình tinh chỉnh mô hình DeepSeek. LoRA là một phương pháp được đề xuất bởi [1]. cho phép giảm thiểu đáng kể số lượng trọng số cần cập nhật bằng cách chèn thêm các ma trận hạng thấp vào trong cấu trúc mô hình đã huấn luyện sẵn. Nhờ đó, quá trình fine-tune có thể được thực hiện với chi phí bộ nhớ và tính toán thấp hơn hàng chục lần so với tinh chỉnh toàn bộ mô hình, nhưng vẫn giữ nguyên được hiệu năng ở mức cao. Vì các trọng số gốc được giữ nguyên, mô hình sau tinh chỉnh bằng LoRA có khả năng duy trì tốt các tri thức nền, qua đó hạn chế hiệu ứng catastrophic forgetting [3].

LoRA cũng mang lại tính linh hoạt cao cho doanh nghiệp: chỉ cần lưu trữ phần trọng số được chèn thêm (adapter) thay vì toàn bộ mô hình sau khi huấn luyện, giúp giảm chi phí lưu trữ và dễ dàng triển khai đa phiên bản cho các bộ phận khác nhau. Trong bài viết này, dữ liệu huấn luyện được lấy từ các nền tảng Jira, cùng với các tệp log nội bộ ghi nhận tiến trình làm việc, phản hồi nhân viên và lịch sử nhiệm vụ. Dữ liệu này được xử lý và cấu trúc lại theo định dạng chuẩn đầu vào của mô hình, từ đó phục vụ cho việc huấn luyện các adapter LoRA theo từng tác vụ chuyên biệt: tạo báo cáo tổng hợp công việc, đề xuất nhiệm vụ tương thích với vai trò, và hỗ trợ định hướng và hướng dẫn xử lý task cho nhân viên mới.

Việc tích hợp mô hình DeepSeek đã được tinh chỉnh vào hệ thống quản lý công việc sẽ không chỉ giúp tự động hóa nhiều khâu quan trọng, mà còn đóng vai trò như một “trợ lý nội bộ”, có khả năng hiểu ngữ cảnh, đề xuất hành động và hỗ trợ ra quyết định theo thời gian thực. Đây là bước tiến cần thiết nhằm nâng cao năng lực vận hành trong bối cảnh doanh nghiệp đang đối mặt với áp lực chuyển đổi số và tối ưu nguồn lực.

## Related Works

Trong vài năm gần đây, việc ứng dụng các mô hình ngôn ngữ lớn (LLM) vào hệ thống hỗ trợ doanh nghiệp đã nhận được nhiều sự quan tâm từ. Các mô hình như GPT, LLaMA, và gần đây là DeepSeek đã cho thấy tiềm năng trong việc xử lý ngôn ngữ tự nhiên một cách linh hoạt và chính xác, mở ra hướng phát triển cho các hệ thống tự động như trợ lý công việc, công cụ tổng hợp báo cáo, và gợi ý tác vụ thông minh. Tuy nhiên, để triển khai các mô hình này một cách hiệu quả trong môi trường doanh nghiệp cụ thể, yêu cầu về tinh chỉnh mô hình (fine-tuning) theo dữ liệu nội bộ trở nên cần thiết.

Một trong những cách giúp tối ưu chi phí là nghiên cứu [1], giới thiệu kỹ thuật **LoRA – Low-Rank Adaptation**. Phương pháp này giữ lại toàn bộ trọng số của mô hình gốc và chỉ huấn luyện thêm hai ma trận hạng thấp tại các lớp attention, giúp giảm mạnh chi phí bộ nhớ và tính toán trong quá trình fine-tune. LoRA được chứng minh có thể đạt chất lượng tương đương với fine-tuning toàn phần trong nhiều tác vụ NLP, đồng thời giảm nguy cơ làm mất đi các kiến thức nền đã học trước đó. Tuy nhiên các nghiên cứu [13] [14] [15] [16] đã chỉ ra rằng, khi áp dụng LoRA trong các mô hình lớn hoặc huấn luyện nhiều tác vụ, việc chỉ thêm ma trận hạng thấp mà không có cơ chế điều hướng có thể khiến mô hình học các biểu diễn chồng lặp hoặc kém đa dạng. Để giải quyết vấn đề này, kỹ thuật ép trực giao đã được đề xuất như một bước cải tiến quan trọng, nhằm tăng cường hiệu quả biểu diễn và khả năng tổng quát hóa. Bên cạnh đó, trực giao hóa còn đóng vai trò như một cơ chế regularization mềm, hạn chế hiện tượng overfitting khi huấn luyện trên các tập dữ liệu nhỏ, và giúp kiểm soát tốt hơn quá trình hội tụ của mô hình.

Đối với những model LLM mã nguồn mở, DeepSeek là mô hình có tiềm năng cao, được thiết kế theo hướng hỗ trợ cộng đồng nghiên cứu triển khai mô hình hiệu quả trên dữ liệu tùy biến. Trong báo cáo kỹ thuật mới nhất, nhóm DeepSeek [2] đã huấn luyện các mô hình từ 1.3B đến 67B tham số với dữ liệu chất lượng cao, hỗ trợ đa ngôn ngữ và tối ưu cho khả năng suy luận logic. DeepSeek-R1, phiên bản fine-tuned reasoning model của họ, đạt kết quả tương đương với các mô hình thương mại như OpenAI GPT-3.5 trong nhiều tác vụ đòi hỏi lập luận nhiều bước [6].

Một số tài liệu thực nghiệm cũng đã chứng minh khả năng ứng dụng của DeepSeek vào các bài toán đặc thù. Ví dụ, trong tài liệu hướng dẫn của DataCamp [7] các tác giả đã thực hiện tinh chỉnh DeepSeek-R1 Distill (8B) bằng kỹ thuật LoRA để xây dựng chatbot y khoa có khả năng suy luận theo chuỗi (chain-of-thought). Mặc dù không liên quan trực tiếp đến tác vụ quản lý công việc, nhưng phương pháp tiền xử lý dữ liệu và chiến lược fine-tuning trong nghiên cứu này hoàn toàn có thể áp dụng cho các bài toán tương tự như tạo báo cáo, phân tích tiến độ, hoặc đề xuất nhiệm vụ trong doanh nghiệp.

Các công trình hiện tại đã xây dựng nền móng quan trọng cho việc áp dụng LLM vào các hệ thống trợ lý doanh nghiệp. Tuy nhiên, vẫn còn thiếu những nghiên cứu cụ thể hóa hướng tiếp cận trong bài toán **quản lý công việc**, nơi đòi hỏi khả năng tích hợp dữ liệu nội bộ (như Jira, hệ thống chat), giữ ổn định tri thức nền và hoạt động hiệu quả trên nền tảng tài nguyên hạn chế. Bài viết này kế thừa các hướng tiếp cận đã được xác lập, đồng thời mở rộng bằng cách kết hợp DeepSeek, và các biến thể của kỹ thuật LoRA và dữ liệu công việc thực tế để xây dựng một hệ thống hỗ trợ quản lý phù hợp với nhu cầu của doanh nghiệp hiện đại.

## Contribution

Bài viết tập trung vào việc thiết kế và phát triển một hệ thống quản lý công việc dựa trên mô hình ngôn ngữ lớn DeepSeek, với các thành phần được tinh chỉnh bằng kỹ thuật Low-Rank Adaptation (LoRA) nhằm đảm bảo hiệu quả triển khai trong môi trường doanh nghiệp có tài nguyên hạn chế.

Bài viết đề xuất một kiến trúc hệ thống quản lý công việc AI phù hợp với bối cảnh doanh nghiệp vừa và nhỏ, trong đó dữ liệu được lấy từ những hệ thống quản lý công việc như Trello, Jira, tài liệu nội bộ (quy trình làm việc, hướng dẫn), được chuyển đổi về định dạng JSONL với cặp câu hỏi–trả lời hoặc hội thoại.

Từ dữ liệu được cung cấp, mô hình sẽ fine-tuning deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B bằng kỹ thuật LoRA (Low-Rank Adaptation) kết hợp với kỹ thuật Orthogonalization với các vector gốc của model và nội bộ các vector nhằm tăng khả năng generalization và giảm trùng lặp biểu diễn. Từ đó tăng khả năng học dữ liệu mới trong những tập dữ liệu hạn chế, khai thác được những dữ liệu khó học.

Bằng cách này, hệ thống có thể huấn luyện adapter nhẹ (LoRA) trên nền mô hình lớn mà không phá vỡ cấu trúc đã học trước đó, đồng thời đảm bảo các adapter có khả năng học kiến thức mới một cách độc lập và tránh bị trùng lặp với mô hình gốc.

# Background Study

Để thiết kế và triển khai thành công một hệ thống quản lý công việc sử dụng trí tuệ nhân tạo, phù hợp với nhu cầu đặc thù của doanh nghiệp, cần xây dựng nền tảng kiến thức vững chắc về cả quy trình vận hành hiện tại và các tiến bộ kỹ thuật trong lĩnh vực mô hình ngôn ngữ lớn. Trước tiên, cấu trúc và đặc điểm dữ liệu của các phương pháp quản lý công việc phổ biến sẽ được phân tích nhằm xác định các điểm có thể tích hợp mô hình AI. Sau đó, các nền tảng kỹ thuật cốt lõi như mô hình DeepSeek, kỹ thuật tinh chỉnh LoRA và chiến lược huấn luyện nhiều giai đoạn, đây cơ sở cho việc lựa chọn công nghệ phù hợp để xây dựng hệ thống hiệu quả, nhẹ và dễ cập nhật theo môi trường thực tế.

## Business Workflow in Enterprises

Trong môi trường doanh nghiệp hiện đại, việc quản lý công việc hiệu quả đóng vai trò quan trọng trong việc đảm bảo năng suất và sự phối hợp giữa các bộ phận. Các tổ chức thường triển khai các hệ thống quản lý quy trình công việc (Workflow Management Systems - WfMS) để tự động hóa và giám sát các quy trình vận hành phức tạp như Jira, Trello. Nên việc thu thập thông tin trao đổi giữa các bộ phận là một việc khả thi.

Tuy nhiên việc lấy dữ liệu từ các hệ thống quản lý quy trình công việc phát sinh một vấn đề. Các công việc thường có sự liên quan nhất định đến sản phẩm, dịch vụ mà công ty cung cấp, nhưng đồng thời cũng có sự tách biệt về thông tin trao đổi trong nội bộ từng đơn vị, phòng ban. Dẫn đến việc dữ liệu khi thu thập vừa có tính tương đồng, nhưng cũng có tính khác biệt trong chi tiết từng công việc. Ví dụ cùng nói về 1 phần mềm, cùng là những thuật ngữ chuyên ngành, nhưng bộ phận BA sẽ cần thu thập cách tương tác với phần mềm; nhưng bộ phận Dev lại có những yêu cầu chuyên sâu về việc hiểu từng chức năng của hệ thống qua code. Từ đó đặt ra yêu cầu về việc mô hình phải hiểu rõ được chính xác sự liên quan khi các bộ phận cùng đề cập đến 1 chức năng, nhưng cũng phải hiểu rõ yêu cầu chuyên sâu của chức năng đó với từng bộ phận là gì.

## DeepSeek model

The DeepSeek-R1-Distill-Qwen-1.5B là một mô hình ngôn ngữ lớn (LLM) với 1.5 tỷ tham số, được phát triển dựa trên kiến trúc Qwen và tinh chỉnh từ phiên bản DeepSeek-R1. Mô hình này được thiết kế nhằm đạt hiệu suất cao trong các tác vụ suy luận logic, lập trình và toán học, đồng thời tối ưu hóa cho việc triển khai trên các hệ thống có tài nguyên hạn chế.

Mô hình sử dụng kiến trúc decoder-only Transformer với các đặc điểm kỹ thuật sau:

* Số lớp (n\_layers): 30
* Kích thước mô hình (d\_model): 4096
* Kích thước lớp trung gian (d\_intermediate): 11008
* Số đầu attention (n\_heads): 32
* Số đầu attention cho key-value (n\_kv\_heads): 32
* Kích thước từ vựng: 102,400 (byte-level BPE)
* Chiều dài ngữ cảnh tối đa: 4096 token

DeepSeek-R1-Distill-Qwen-7B có thể được triển khai trên các hệ thống với cấu hình phần cứng vừa phải [8]:

* GPU đề xuất: NVIDIA RTX 3070 với VRAM tối thiểu 8GB
* Yêu cầu VRAM: Khoảng 3.3GB
* Hỗ trợ lượng tử hóa: Có thể giảm yêu cầu VRAM xuống còn khoảng 4GB khi sử dụng lượng tử hóa 4-bit

Điều này cho phép các doanh nghiệp triển khai mô hình trên các máy trạm tiêu chuẩn hoặc thậm chí trên các máy tính cá nhân với cấu hình phù hợp. DeepSeek-R1-Distill-Qwen-1.5B đã được đánh giá cao trong các bài kiểm tra về khả năng suy luận và lập trình, đạt kết quả tương đương với các mô hình lớn hơn như Qwen3-235B-thinking. Mô hình này tận dụng kỹ thuật chain-of-thought từ DeepSeek-R1 để cải thiện khả năng suy luận, đồng thời duy trì hiệu suất cao trong khi giảm thiểu yêu cầu về tài nguyên tính toán. Với sự cân bằng giữa hiệu suất và khả năng triển khai, DeepSeek-R1-Distill-Qwen-1.5B là lựa chọn phù hợp cho các ứng dụng doanh nghiệp yêu cầu mô hình ngôn ngữ lớn có khả năng suy luận mạnh mẽ nhưng vẫn có thể triển khai trên hạ tầng phần cứng hiện có.

## Low-Rank Adaptation

LoRA (Low-Rank Adaptation) là một kỹ thuật parameter efficient fine tuning, được đề xuất bởi [1], nhằm giảm thiểu số lượng tham số cần cập nhật khi tinh chỉnh các mô hình ngôn ngữ lớn. Trong phương pháp tinh chỉnh truyền thống, toàn bộ ma trận trọng số W của mô hình sẽ được cập nhật trong quá trình huấn luyện lại, điều này đòi hỏi rất nhiều tài nguyên tính toán và bộ nhớ – đặc biệt khi mô hình có hàng tỷ tham số [1]. Khác với cách tiếp cận này, LoRA giữ nguyên các trọng số gốc và chỉ thêm vào các ma trận có hạng thấp, giúp giảm đáng kể chi phí tính toán và bộ nhớ cần thiết trong quá trình huấn luyện.

Thay vì việc cập nhật trực tiếp ma trận trọng số gốc *W*, LoRA giữ nguyên *W* và chèn thêm một phần hiệu chỉnh có hạng thấp, ký hiệu là ΔW. Với mô hình có một matrận trọng số W∈R*d×k* – nghĩa là có d hàng và k cột, LoRA sẽ thêm vào một ma trận mới được tính từ tích của hai ma trận nhỏ hơn theo công thức

Trong đó:

* A∈Rr×k
* B∈Rd×r
* r≪min(d,k): độ nén của ma trận hiệu chỉnh

Ma trận A đóng vai trò chiếu thông tin đầu vào xuống ma trận hạng thấp, đại diện cho các hướng học mới của trọng số gốc W. Ma trận B: tái mở rộng thông tin từ không gian thấp lên không gian đầu ra. Do đó, toàn bộ thông tin mới được học trong quá trình fine-tuning được mã hóa vào ΔW với chi phí tham số rất thấp.

Cuối cùng, trọng số được sử dụng trong mô hình sau khi áp dụng LoRA là:

Trong đó, α là hệ số điều chỉnh, sử dụng như một scaling factor. Nhằm đảm bảo rằng tổng mức ảnh hưởng của phần hiệu chỉnh ΔW là phù hợp, bất kể giá trị r được chọn là bao nhiêu. Điều này cho phép việc thay đổi r (độ nén của ma trận hiệu chỉnh) mà không làm mất cân bằng mức độ tác động của phần tinh chỉnh lên mô hình gốc [1].

Từ thực tiễn, có một vài cách để chọn α tối ưu cho mô hình; Determined AI khuyến nghị bắt đầu với α=r, sau đó giữ ổn định giá trị này khi thay đổi rank mà không cần tinh chỉnh lại α [9], theo tài liệu AutoTrain Hugging Face đưa ra giá trị mặc định α=32 với rank = 16, và cảnh báo việc đặt α≫r có thể gây overfitting [10]. Sebastian Raschka cũng đưa ra nguyên tắcα / r = 2 [11] là một tỷ lệ hợp lý để giúp ΔW không quá mạnh cũng không quá yếu so với trọng số gốc. Tuy nhiên, theo nghiên cứu của [12] việc sử dụng γr=α/r có thể gây ra hiện tượng suy giảm tốc độ học và hiệu suất tinh chỉnh, đặc biệt khi sử dụng các giá trị r lớn, tác giả đã đề xuất một phương pháp mới gọi là rsLoRA (rank-stabilized LoRA), trong đó hệ số điều chỉnh được thay bằng γr=α/√r để giúp giữ cho độ lớn của phần hiệu chỉnh ΔW ổn định hơn khi r tăng, từ đó cải thiện độ hội tụ và hiệu quả huấn luyện mà không làm ảnh hưởng đến khả năng suy luận của mô hình. Kalajdzievski chứng minh rằng rsLoRA không chỉ cải thiện hiệu suất trên các bài toán benchmark mà còn giảm yêu cầu về tài nguyên tính toán khi huấn luyện các mô hình ngôn ngữ lớn [12].

Việc sử dụng ma trận hạng thấp giúp mô hình áp dụng kỹ thuật LoRA đạt hiệu suất tinh chỉnh tương đương với các phương pháp truyền thống, nhưng chỉ cần 0.1% đến 1% số thamsố được huấn luyện so với fine-tuning đầy đủ [1]. Điều này khiến LoRA trở nên đặc biệt hữu ích trong các môi trường hạn chế tài nguyên, chẳng hạn như khi huấn luyện trên CPU hoặc GPU dung lượng thấp, giúp các doanh nghiệp vừa và nhỏ cũng có thể dễ dàng huấn luyện các mô hình AI của riêng họ.

## Orthogonal trong Low-Rank Adaptation

Orthogonalized trong fine-tuning trong Large Language Models là một kỹ thuật để ép các vector hàng hoặc cột của ma trận vuông góc góc với nhau và có độ dài bằng 1. Điều này sẽ ép mô hình phải học thêm thông tin mới và tránh việc học chung hướng với những dữ liệu đã có sẵn khiến mô hình bị catastrophic forgetting [13] [14].

Các nghiên cứu như Orthogonal Low-Rank Adaptation [13] chỉ ra bằng cách chia sẵn dữ liệu thành các task với chủ đề hoặc mục tiêu riêng biệt, và huấn luyện tuần tự từng task đó theo nguyên tắc:

* Ma trận A phải trực giao mềm với ma trận A của những task trước đó
* Ma trận A trực giao với các hàng khác trong chính nó.

Bằng cách ép các hướng học (ma trận A) của task mới **t**rực giao với các task trước, O-LoRA đảm bảo mô hình không ghi đè kiến thức đã học, từ đó duy trì hiệu suất tốt cho các tác vụ cũ mà không cần truy cập lại dữ liệu gốc. Mỗi task được ánh xạ vào một không gian con khác biệt nhờ tính chất trực giao của các vector học, giúp mô hình phân biệt rõ ràng giữa các nhiệm vụ khác nhau, tránh nhiễu loạn trong quá trình huấn luyện đa nhiệm. Nhờ không gian học bị giới hạn bởi trực giao, mô hình bị "bắt buộc" phải học các hướng mới thay vì khai thác quá mức vào các trọng số cũ, điều này giúp giảm nguy cơ overfit trên các task ít dữ liệu.

Tuy trực giao mang lại hiệu quả cao trong việc bảo toàn tri thức cũ và tránh nhiễu giữa các tác vụ thông qua ràng buộc trực giao, nhưng việc ép ma trận học A của mỗi task phải vuông góc với các task trước cũng có thể vô tình giới hạn không gian biểu diễn mà mô hình có thể học. Điều này đặc biệt đúng trong bối cảnh các task có liên quan cao hoặc chia sẻ nhiều đặc trưng chung – khi đó, việc bắt buộc học theo các hướng hoàn toàn mới có thể khiến mô hình không tận dụng được các biểu diễn hiệu quả sẵn có. [14] [15] chỉ ra các mô hình chỉ được cập nhật trong một không gian biểu diễn bị giới hạn tuy hiệu quả trong việc giảm nhiễu và tránh interference, nhưng lại làm giảm đáng kể khả năng học linh hoạt của mô hình do không thể điều chỉnh đầy đủ các thành phần trọng số.

Tuy việc áp dụng ràng buộc trực giao giúp tăng cường khả năng bảo toàn tri thức cũ và tách biệt biểu diễn giữa các nhiệm vụ, song thực tế cho thấy không phải tất cả các module trong mô hình ngôn ngữ lớn đều phù hợp để ép trực giao. Việc lựa chọn các module cần áp dụng ràng buộc này nên được thực hiện một cách có chọn lọc, dựa trên vai trò chức năng, đặc điểm biểu diễn, và mức độ ảnh hưởng của từng module đến quá trình học.

Trong kiến trúc Transformer, các module tuyến tính quan trọng trong khối self-attention bao gồm:

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

Các module này chịu trách nhiệm xây dựng và điều phối quá trình attention giữa các token trong chuỗi đầu vào. Cụ thể:

* q\_proj và k\_proj xác định cách một token so sánh và đánh giá độ liên quan với các token khác trong cùng chuỗi, thông qua query và key. Nếu các vector học tại đây bị trùng hướng hoặc suy giảm độ phân biệt, attention sẽ không còn đủ sắc nét để phân tách ngữ nghĩa giữa các phần tử khác nhau.
* v\_proj tạo ra vector biểu diễn thông tin cần trích xuất — nếu bị ảnh hưởng bởi các biểu diễn cũ không liên quan, mô hình có thể "quên" biểu diễn mới cần học.
* o\_proj tổng hợp toàn bộ thông tin sau khi attention được tính toán, đóng vai trò đưa kết quả về không gian ẩn của mô hình. Nếu không kiểm soát trực giao tại đây, các biểu diễn mới có thể bị "nhồi" trùng lặp với các biểu diễn cũ, làm giảm khả năng mở rộng của mô hình.

Nghiên cứu [16] chỉ ra rằng các projection trong attention là những module dễ bị “trùng hướng” nhất sau fine-tuning. Việc ép trực giao các projection này giúp ngăn chặn hiện tượng biểu diễn chồng lấp, từ đó tăng tính đa dạng trong không gian biểu diễn của attention. Ngoài ra, vì các module này thường không có hàm phi tuyến xen giữa (trái ngược với các projection trong MLP), việc áp dụng ràng buộc trực giao tại đây không làm suy giảm khả năng biểu diễn phi tuyến của mô hình.

Ngược lại, các module thuộc phần MLP (feed-forward network) như gate\_proj, up\_proj, down\_proj lại có đặc tính khác biệt:

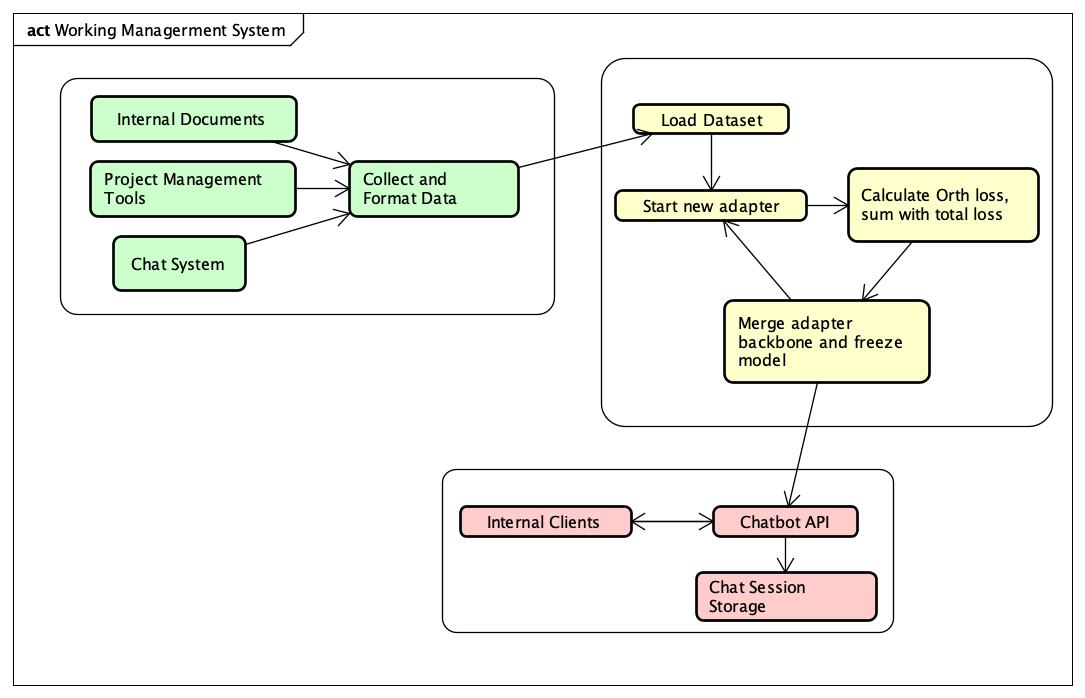
* Chúng đi kèm với các hàm kích hoạt phi tuyến như GELU,
* Chịu ảnh hưởng mạnh của biến đổi không gian,
* Có xu hướng biểu diễn nội dung phi tuyến tính cao.
* Do đó, việc ép trực giao các module MLP có thể làm mất đặc trưng phi tuyến cần thiết, gây ảnh hưởng tiêu cực đến khả năng học biểu diễn sâu của mô hình.

Từ các phân tích trên, việc ép trực giao nên được áp dụng một cách chọn lọc, tập trung vào các module q\_proj, k\_proj, v\_proj, o\_proj, nơi mà biểu diễn tuyến tính đóng vai trò trung tâm trong khả năng truy xuất, tổng hợp và phản hồi thông tin. Cách tiếp cận này vừa giúp khai thác ưu điểm của trực giao trong bảo toàn tri thức và học không gian mới, vừa tránh những hệ quả không mong muốn do giới hạn không gian biểu diễn một cách không cần thiết.

# System Design

## System Architecture

Hệ thống chatbot hỗ trợ hỏi đáp nội bộ được thiết kế theo kiến trúc ba tầng chính: Thu thập dữ liệu, Xử lý & Huấn luyện mô hình, và Triển khai inference thông qua API. Kiến trúc tổng thể của hệ thống được minh họa trong Hình 1.



**Figure 1.** Overall architecture

Ở tầng đầu tiên, dữ liệu được thu thập từ ba nguồn chính: tài liệu hướng dẫn nội bộ, các hệ thống quản lý công việc như Jira hoặc Trello, và thông tin khảo sát/phỏng vấn từ nhân sự trong tổ chức. Sau khi được thu thập, dữ liệu được chuẩn hóa thành các cặp câu hỏi–trả lời theo định dạng instruction-tuning phù hợp cho các mô hình ngôn ngữ lớn.

Tầng huấn luyện mô hình áp dụng kỹ thuật LoRA kết hợp ràng buộc trực giao giúp hạn chế hiện tượng catastrophic forgetting và gia tăng khả năng học nhiều hướng khác. Mô hình ngôn ngữ gốc được sử dụng là DeepSeek-R1-Distill-Qwen-1.5B, một phiên bản nhẹ có hiệu suất cao và hỗ trợ tiếng Việt.

Cuối cùng, mô hình sau huấn luyện được lưu trữ và triển khai dưới dạng API thông qua nền tảng Hugging Face Transformers. API này cho phép các hệ thống nội bộ gọi đến mô hình chatbot để nhận phản hồi theo ngữ cảnh, đồng thời lưu trữ lịch sử phiên trò chuyện để tăng khả năng hiểu ngữ cảnh trong các tương tác tiếp theo.

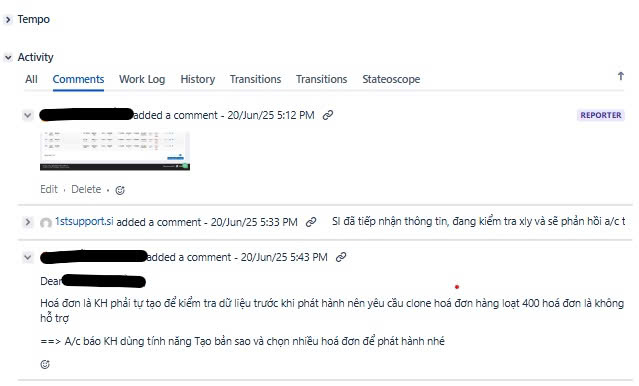
## Data Preprocessing

### Data extraction

Dữ liệu được thu thập dựa trên 2 nguồn chính:

* Dữ liệu lấy từ các hệ thống quản lý công việc trên hệ thống Jira
* Dữ liệu dựa trên các tài liệu hướng dẫn nội bộ và public

Với dữ liệu của những hệ thống quản lý công việc, đây là những dữ liệu có độ clean rất cao, chỉ cần các bước clean những dòng dữ liệu trung gian như trong Figure 2 và thay đổi lại cách xưng hô cho phù hợp với mô hình là có thể trở thành dữ liệu train trực tiếp cho AI.



**Figure 2.** Data taken from Jira

Comment của các cấp này thường có các từ lặp đi lặp lại, nhiều comment do hệ thống tự tạo hoặc có 1 format chung trong đó các từ dạng: nhờ.\*hỗ trợ", xử lý.\*sớm,… chiếm quá nửa nội dung của comment, những dữ liệu này hoàn toàn có thể lọc qua query hoặc regex. Một hạn chế lớn của dữ liệu lấy từ hệ thống quản lý công việc là nhiều luồng xử lý nằm ngoài Jira, như dữ liệu của các file code, các thao tác trực tiếp trên hệ thống khác, Jira chỉ là nơi truyền và thông báo kết quả, vì vậy lượng dữ liệu thực sự có thể đưa thành dữ liệu train chỉ được 20% trong tổng số dữ liệu.

Ngoài những dữ liệu từ các hệ thống như Jira, Trello. Các công ty khi triển khai một sản phẩm dịch vụ thường có các file excel dạng FAQ với khách hàng hoặc người được hướng dẫn. Đây là những dữ liệu chất lượng nhất, khi các câu hỏi được trả lời thường đã được lọc sẵn những câu hỏi tương tự nhau, và câu trả lời cũng rất chi tiết. Tuy nhiên hạn chế của những dữ liệu này là sẽ có những dữ liệu mang tính thời điểm, như bao giờ triển khai chức năng A hay B, để hạn chế việc này, khi đưa vào làm dữ liệu training, hệ thống có thể tìm và nhận biết những câu hỏi có dạng: khi nào có, bao giờ triển khai,… Để dữ liệu được clean hơn.

Với dữ liệu dạng tài liệu hướng dẫn, các công văn văn bản, đây là tài liệu có tính chuẩn xác cao nhất, nhất là những tài liệu dạng luật, văn bản, nghị định. Tuy nhiên thách thức là các tài liệu này cần khả năng tổng quát cao và chưa phải là một dạng format trao đổi công việc. Bài viết đề xuất cho các mô hình như Chat GPT, Gemimi đọc trước file tài liệu để tổng quát được lượng kiến thức. Rồi yêu cầu các mô hình tạo ra các đoạn hội thoiajt dựa trên tài liệu đó.

### Data Formatting for Instruction-Tuning

Sau khi đã lấy được dữ liệu theo dạng cặp câu hỏi và câu trả lời, dữ liệu cần được format để huấn luyện. Bài viết đề xuất format dữ liệu theo chuẩn Hugging Face Chat Template [17]. Hugging Face Chat Template Là định dạng dữ liệu hội thoại nhiều lượt (multi-turn) được sử dụng bởi các mô hình dạng "assistant". Dữ liệu được tổ chức theo danh sách các cặp {role, content}, sau đó được biến đổi thành prompt huấn luyện thông qua hàm template được định nghĩa sẵn trong tokenizer. Hàm chat\_template sẽ chịu trách nhiệm ghép các lượt hội thoại thành một chuỗi đầu vào duy nhất. Định dạng này giúp mô hình học được cấu trúc cuộc trò chuyện tự nhiên, giữ ngữ cảnh tốt hơn và hỗ trợ các bài toán như trả lời truy vấn, giải thích dữ liệu, function calling. Đây là định dạng chuẩn hiện được sử dụng bởi các mô hình như LLaMA2-chat, Mistral, DeepSeek, và được hỗ trợ trực tiếp bởi thư viện transformers của Hugging Face khi huấn luyện hoặc sinh dữ liệu đầu vào.

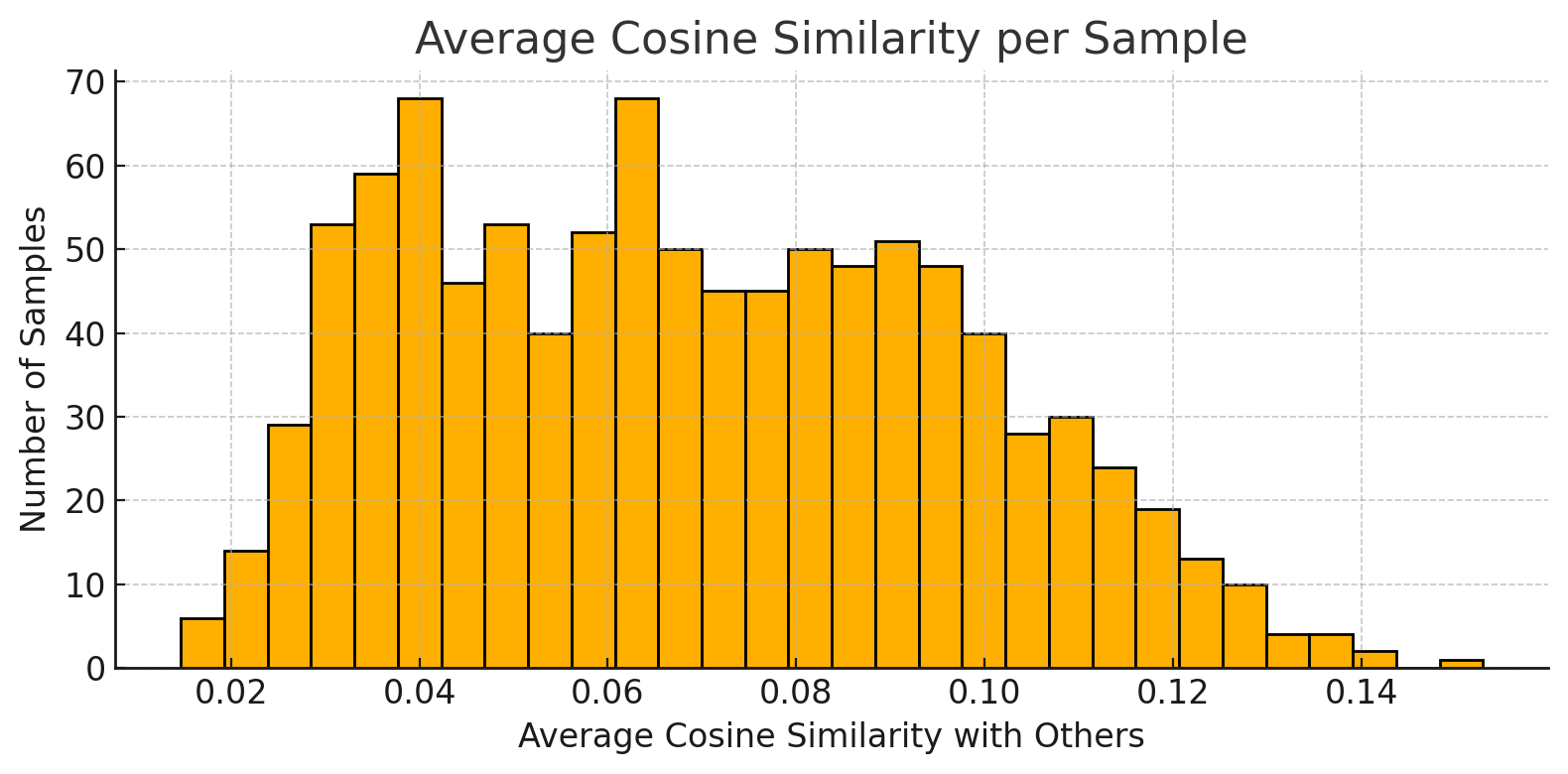
**Table 1.** Data in Hugging Face Chat Template format

|  |
| --- |
| {"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>"}]} |

Với dữ liệu lấy từ các hệ thống quản lý công việc, người tạo task để role là "user", vì thường đây là người đưa ra thắc mắc chính và là bên cung cấp thông tin, những người khác trong task sẽ là đặt role là "assistant", những dữ liệu này thường chia ra nhiều lượt nói/trao đổi thông tin.

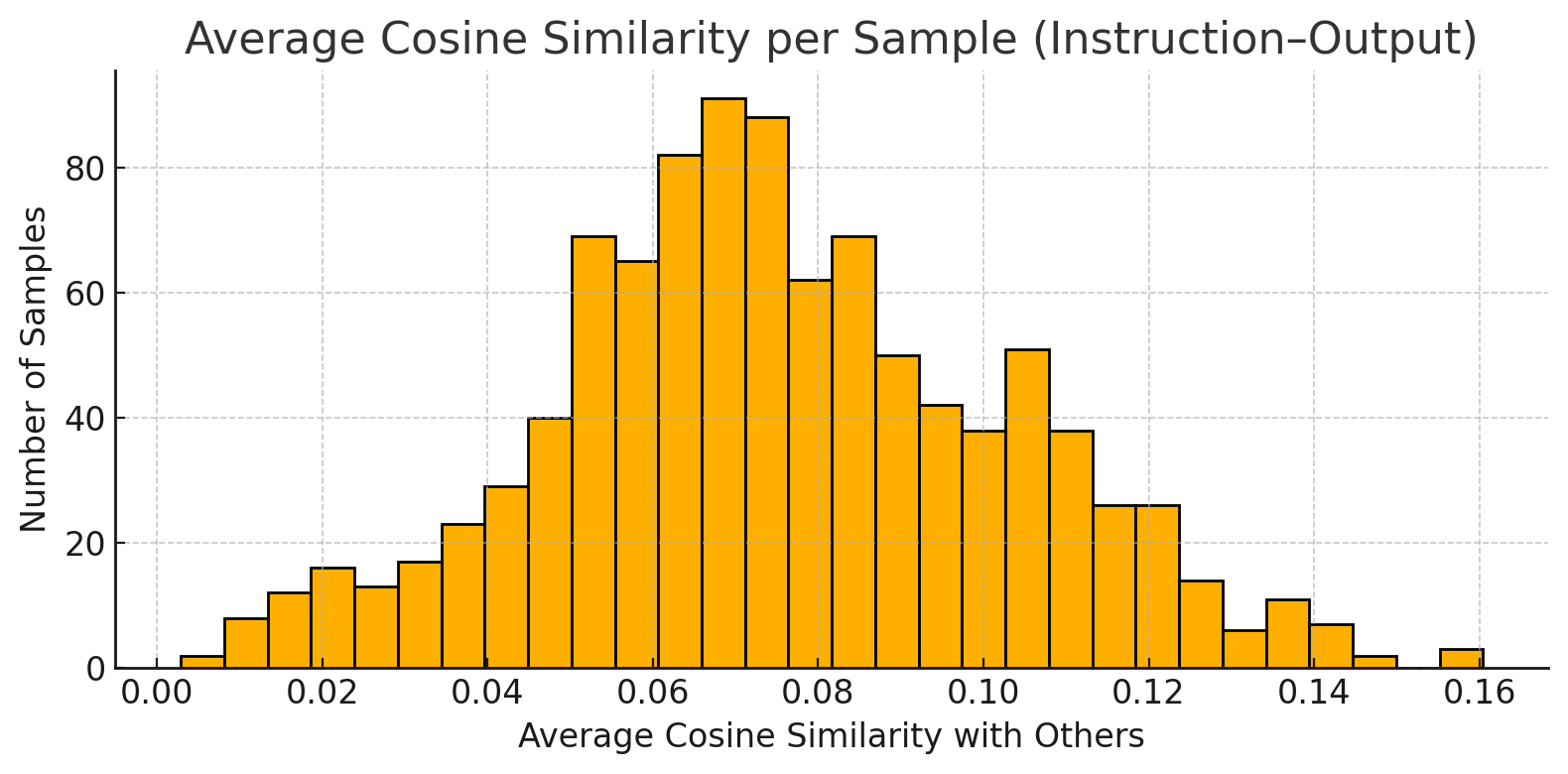
Để đảm bảo được tính đa dạng của mô hình, bài viết dựa theo nghiên cứu của [18] chứng minh rằng việc pha trộn dữ liệu thực và dữ liệu sinh (GPT-generate) theo tỷ lệ vàng ~61.8 : 38.2 sẽ giúp mô hình giữ được độ chính xác từ dữ liệu thật, vừa mở rộng được khả năng biểu đạt, tổng quát hóa nhờ vào sự đa dạng ngôn ngữ của dữ liệu sinh. Trong ngữ cảnh của bài viết, với 1.000 mẫu dữ liệu thật đang có, tỷ lệ này đồng nghĩa với việc cần bổ sung thêm khoảng 618 mẫu dữ liệu generate để đạt tổng ~1.618 mẫu nhằm đảm bảo tính đa dạng mà không giảm chất lượng của mô hình.

Phân tích Cosine Similarity cho thấy mức độ đa dạng cao trong dữ liệu thực. Độ tương đồng trung bình giữa từng mẫu với phần còn lại chủ yếu nằm trong khoảng 0.04–0.10, phản ánh nội dung khác biệt rõ rệt giữa các mẫu. Chỉ khoảng 5–10% mẫu vượt ngưỡng similarity 0.12, cho thấy nguy cơ trùng lặp (paraphrase) là rất thấp. Điều này chứng tỏ tập dữ liệu thực mang tính đa dạng cao, phù hợp để fine-tuning mô hình ngôn ngữ mà không lo bị overfit.



**Figure 3.** Data taken from Jira

Với những dữ liệu Generate từ Chat GPT, chất lượng của bộ dữ liệu không được bằng dữ liệu lấy từ Jira, Excel khi có >120 cặp gần giống nhau (cosine similarity > 0.90), nội dung có thể bị viết lại lặp lại (paraphrase) nhiều lần.



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

## Modeling Module

Model gốc là DeepSeek-R1-Distill-Qwen-1.5B, được fine-tuning bằng kỹ thuật LoRA kết hợp ràng buộc trực giao theo pipeline như sau:

* Tokenization & Data Formatting: Dữ được tiền xử lý và chuẩn hóa theo định dạng chat format phù hợp với mô hình Qwen, sau đó được mã hóa thành token thông qua tokenizer của DeepSeek
* Trước khi bắt đầu quá trình huấn luyện, các ma trận trọng số gốc A của mô hình nền được load lại và lưu trữ dưới dạng tensor cố định (detach). Các ma trận này được dùng để tính toán external orthogonality đối với adapter hiện tại.
* Trong quá trình huấn luyện, các vector của ma trận A Ép các vector hàng trong chính một ma trận A phải trực giao với nhau, bằng cách tối thiểu hóa sai số giữa *AAT* và ma trận đơn vị *I*. Ràng buộc này đảm bảo adapter học được thông tin mới mẻ, tránh việc “chép lại” hướng biểu diễn cũ.

Mô hình bổ sung hai tham số đặc biệt nhằm đảm bảo ràng buộc trực giao giữa các vector trong các module LoRA, bao gồm:

* Lambda internal: đại diện cho mức độ kiểm soát tính trực giao giữa các vector hàng trong cùng một ma trận A (nội bộ từng module LoRA).
* Lambda external: đại diện cho mức độ kiểm soát trực giao giữa ma trận A hiện tại và các ma trận A của model gốc.

Tổng loss được sử dụng trong quá trình huấn luyện kết hợp giữa loss chính của mô hình và hai hàm tham số nói trên theo công thức:

Các ma trận ép trực giao với 1 số module nhất định, bao gồm các module:

* q\_proj: do vector quyết định trực tiếp việc token muốn hỏi gì, nên ép trực giao là điều cần thiết
* v\_proj: tạo các vector value để được tổng hợp trong attention, vì vậy ép trực giao là cần thiết để giúp mô hình học các giá trị bổ sung khác nhau đảm bảo sự đa dạng thông tin.

# Experiments And Results

## Data collection

Dữ liệu của được thu thập từ hệ thống quản lý Jira nội bộ và các tài liệu liên quan đến quy trình xử lý hóa đơn điện tử. Trong đó, các tệp Excel dạng FAQ là Trong quá trình triển hệ thống hoá đơn điện tử theo thông tư 70. Bên cạnh đó, một phần dữ liệu được sinh tự động từ nội dung các tài liệu hướng dẫn chính thức được ban hành từ 25/05/2025, thông qua mô hình ChatGPT để đảm bảo độ phủ và tính đa dạng của tập dữ liệu. Dữ liệu tập trung vào 2 bộ phận:

Bộ phận developer: tập trung vào hướng dẫn dử dụng base của hệ thống để code các chức năng mở rộng và các tài liệu tích hợp hệ thống hoá đơn điện tử được publish cho bên thứ 3.

Bộ phận it help desk: tập trung vào hỗ trợ khách hàng và hướng dẫn các quy trình, các quy định về hoá đơn theo thông tư 70.

Tập dữ liệu được phân chia theo tỷ lệ "vàng" giữa dữ liệu thật và dữ liệu sinh, với 66,6% là dữ liệu thật (tương đương 1.000 mẫu), và 33,3% là dữ liệu được generate (tương đương 618 mẫu), trên tổng số 1.618 mẫu.

Dữ liệu sau khi tổng hợp được tiền xử lý, lọc trùng lặp, và định dạng lại theo chuẩn chat-format đầu vào cho mô hình ngôn ngữ. Bảng tổng hợp phân bố dữ liệu và đặc điểm thống kê được trình bày trong Bảng 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

Quá Trong thực nghiệm, mô hình được sử dụng là DeepSeek-R1-Distill-Qwen-1.5B, một phiên bản nhẹ nhưng hiệu suất cao của dòng DeepSeek, với khả năng hỗ trợ tiếng Việt và suy luận logic tốt. Đây là mô hình có 1.5 tỷ tham số, được phát hành mã nguồn mở, giúp dễ dàng triển khai và tinh chỉnh trong các môi trường doanh nghiệp vừa và nhỏ mà không yêu cầu hạ tầng tính toán quá mạnh.

Quá trình huấn luyện được thực hiện theo kỹ thuật LoRA, được chia thành 2 lần huấn luyện với mỗi lần 3 model.  Bảng 3 là thông tin cụ thể về việc ép trực giao của các model.

**Table 3.** Orthogonal extrusion module

|  |  |  |
| --- | --- | --- |
| Lần huấn luyện | Model | Ép trực giao |
| Lần 1 | Lora | Không |
| Lần 1 | OLora | q\_proj, v\_proj |
| Lần 1 | SoLora | q\_proj, v\_proj |
| Lần 2 | Lora | Không |
| Lần 2 | OLora | q\_proj, v\_proj, k\_proj |
| Lần 2 | OLora All | q\_proj, v\_proj, k\_proj, o\_proj, gate\_proj |
| Lần 2 | SoLora | q\_proj, v\_proj |

Các model được huấn luyện với cùng 1 cấu hình, cụ thể là:

* Rank: 16
* Learning Rate: 5e-4
* Number of epochs: 30
* Batch size: 2
* Tokenizer length: 128
* Warmup ratio: 0.1
* Lambda external: 0.0001
* Lambda internal: 1.0

Do Để đảm bảo tính khả thi và tiết kiệm chi phí, quá trình huấn luyện được triển khai trên nền tảng Google Colab Pro với phần cứng hỗ trợ GPU chuyên dụng. Nhờ vào tài nguyên GPU mạnh mẽ và khả năng mở rộng linh hoạt của Colab, mô hình DeepSeek-R1-Distill-Qwen-1.5B có thể được huấn luyện hiệu quả với batch size và độ dài token phù hợp.

Cấu hình môi trường cụ thể:

Phần cứng:

* Nền tảng: Google Colab Pro
* GPU: NVIDIA Tesla T4 hoặc tương đương
* (15.0 GB GPU RAM – khả dụng đầy đủ trước khi bắt đầu training)
* RAM hệ thống: 51.0 GB
* Dung lượng ổ đĩa khả dụng: ~198 GB (đã sử dụng 37.7 GB / 235.7 GB)

Phần mềm:

* Python: 3.x (mặc định của Colab)
* PyTorch: >= 2.1 với hỗ trợ CUDA
* Transformers: v4.41 trở lên
* PEFT: v0.9 (hoặc tương thích)

Thiết lập tối ưu:

* fp16: Bật (fp16=True) để tận dụng tính năng tăng tốc trên GPU.
* Batch size: nhỏ (ví dụ: 1–2) để tránh vượt ngưỡng 15GB GPU VRAM.
* Chiều dài token: giới hạn dưới 128 tokens/mẫu để giảm tải bộ nhớ.

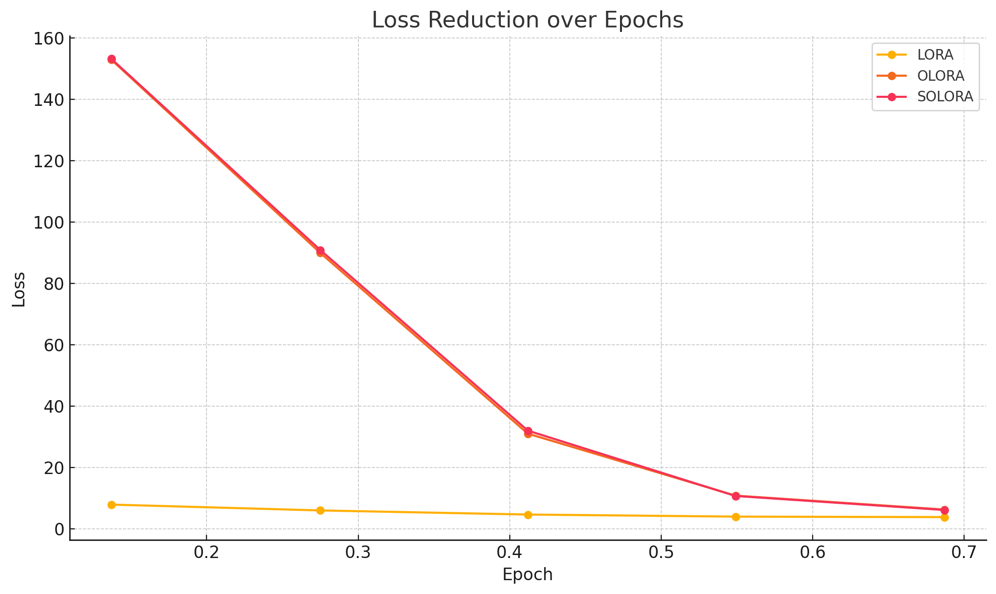
## Results

Trong quá trình huấn luyện, ba mô hình LoRA, OLoRA và SoLoRA thể hiện những đặc điểm học rất khác biệt, phản ánh rõ chiến lược tối ưu mà mỗi phương pháp áp dụng. Mô hình LoRA khởi đầu với giá trị loss là 7.8713 – thấp so với hai mô hình còn lại. Ngay trong chưa tới một epoch, loss của LoRA giảm nhanh chóng xuống chỉ còn khoảng 6.25, cho thấy khả năng hội tụ cực kỳ nhanh khi không bị giới hạn bởi cấu trúc học nào. Tuy nhiên, chính điều này cũng phản ánh sự thiếu kiểm soát trong không gian tham số, có thể dẫn đến việc học lệch, overfit hoặc ghi đè lên kiến thức cũ trong các tác vụ nối tiếp.

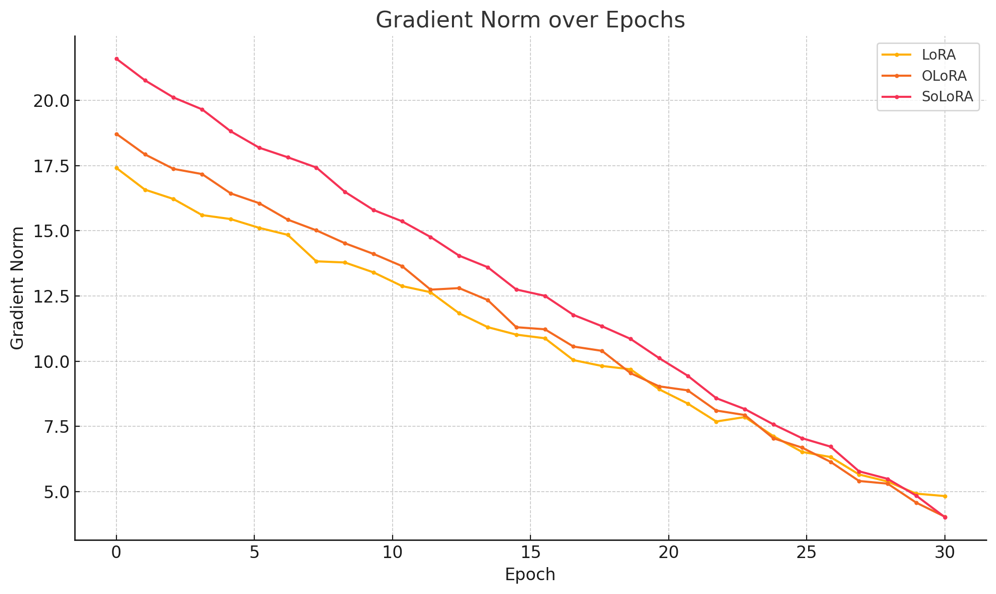
Ngược lại, OLoRA, với cơ chế ép trực giao lên các ma trận A trong module LoRA, bắt đầu với loss cao hơn đáng kể (181.01), do mô hình phải học trong một không gian hẹp hơn. Mặc dù vậy, loss của OLoRA cũng giảm đều đặn và xuống mức 6.58 ở giai đoạn sau, tiệm cận hiệu quả với LoRA mà vẫn đảm bảo được tính ổn định. Điều này chứng tỏ rằng việc ràng buộc trực giao không làm suy giảm quá trình học, mà còn giúp mô hình tránh việc học trùng lặp với các hướng tham số đã tồn tại từ các tác vụ trước đó.

Mô hình SoLoRA, là sự kết hợp giữa trực giao hóa và hợp tác giữa các adapter, thể hiện độ phức tạp lớn nhất trong quá trình học. Với loss khởi đầu lên tới 204.4 – cao nhất trong cả ba – SoLoRA dường như bị “kìm hãm” mạnh bởi các ràng buộc chồng chéo. Tuy nhiên, đáng chú ý là loss của SoLoRA lại giảm nhanh nhất, chỉ còn 5.92 sau một chu kỳ học, thấp hơn cả LoRA và OLoRA. Điều này cho thấy chiến lược kết hợp học phối hợp và trực giao không chỉ giữ được sự phân biệt giữa các tác vụ mà còn khai thác hiệu quả tri thức chia sẻ.

Xét về gradient norm, LoRA bắt đầu với giá trị khoảng 17.26 và giảm dần về khoảng 4.5, thể hiện quá trình hội tụ đều đặn. OLoRA khởi đầu cao hơn (18.42), nhưng gradient cũng giảm ổn định, phản ánh tính học ổn định và có kiểm soát. Trong khi đó, SoLoRA có gradient norm ban đầu lên đến 21.31 – lớn nhất – cho thấy cường độ học rất mạnh từ đầu để vượt qua các ràng buộc phức tạp, nhưng cũng giảm về mức tương đương với hai mô hình còn lại sau 30 epoch, chứng minh tính hội tụ cao và hiệu quả của mô hình. Tóm lại, quá trình học của ba mô hình cho thấy SoLoRA là mô hình hiệu quả nhất dù bị ràng buộc mạnh, OLoRA là lựa chọn cân bằng giữa độ chính xác và khả năng duy trì tri thức, trong khi LoRA phù hợp với các bài toán đơn giản hoặc không yêu cầu học nối tiếp.



**Figure 5.** Training Loss Comparison



**Figure 6.** Gradient Norm Over Epochs

Kết quả So sánh độ chính xác trung bình giữa các phương pháp huấn luyện dựa trên LoRA cho thấy rằng OLoRA (áp dụng ràng buộc trực giao lên các ma trận q\_proj, k\_proj, và v\_proj) đạt hiệu suất cao nhất với độ chính xác trung bình 0.831. Điều này cho thấy việc ép trực giao nhẹ lên các hướng học quan trọng giúp mô hình giữ lại thông tin đặc trưng cho từng tác vụ tốt hơn. SoLoRA – phương pháp bổ sung thêm ràng buộc trực giao giữa các vector nội bộ – cũng đạt kết quả cao với độ chính xác 0.812, cho thấy lợi ích từ việc điều chỉnh cấu trúc không gian biểu diễn nội bộ. Trong khi đó, LoRA tiêu chuẩn (không có ép trực giao) đạt 0.796, thấp hơn so với hai phương pháp trên. Cuối cùng, OLoRA AL – phương pháp mở rộng ép trực giao lên cả o\_proj và gate\_proj – lại có hiệu suất kém nhất (0.684), cho thấy việc mở rộng ép trực giao không phù hợp lên các module ít đóng vai trò biểu diễn có thể gây nhiễu và ảnh hưởng tiêu cực đến kết quả huấn luyện.

Sau khi huấn luyện xong, các mô hình được test trên tập dữ liệu cũ và được so sánh dựa trên Cosine similarity, một kỹ thuật phổ biến trong xử lý ngôn ngữ tự nhiên (NLP) để đo mức độ tương đồng giữa hai vector, thường được dùng để đánh giá sự giống nhau về ngữ nghĩa giữa hai câu hoặc văn bản.

OLoRA có độ chính xác trung bình cao nhất (0.8139), cho thấy việc ép trực giao với ma trận gốc đã giúp mô hình giữ được kiến thức cũ và học được thông tin mới hiệu quả hơn.

SoLoRA có min thấp nhất (0.13) và khoảng dao động lớn nhất (0.87), điều này có thể cho thấy việc ép trực giao cả nội bộ vector giúp mô hình đa dạng hóa hướng học, nhưng lại làm mất sự ổn định ở một số mẫu.

LoRA thường có kết quả khá ổn định, nhưng không vượt trội, do không có cơ chế nào ngăn mô hình học lại theo hướng cũ – có thể dẫn đến hiện tượng catastrophic forgetting.

**Table 4.** Orthogonal extrusion module

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

So Tuy nhiên, khi đánh giá trên tập dữ liệu đã được chỉnh sửa câu chữ – cụ thể là các câu hỏi liên quan đến lỗi ERR từ 1 đến 50 với nhiều cách diễn đạt khác nhau – kết quả cho thấy SoLoRA (trước là CoLoRA) thể hiện vượt trội về khả năng hiểu ngữ nghĩa:

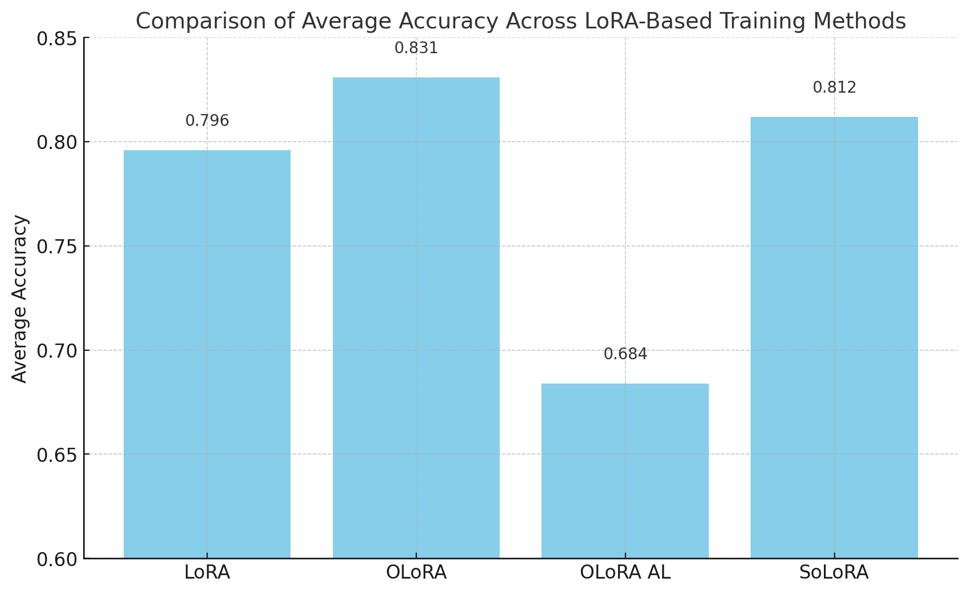
o SoLoRA trả lời chính xác 18/20 câu hỏi,

o Trong khi LoRA chỉ đúng 3/20,

o Và OLoRA đạt 5/20.

Điều này cho thấy rằng, mặc dù SoLoRA có độ chính xác trung bình (cosine similarity) thấp hơn một chút so với OLoRA trong tập kiểm thử ban đầu, nhưng khả năng tổng quát hóa và hiểu các câu hỏi có ngôn ngữ biến đổi của SoLoRA lại vượt trội hơn hẳn.

Từ đó, có thể rút ra rằng việc đánh đổi một phần độ chính xác ngữ nghĩa để đạt được năng lực hiểu sâu hơn và linh hoạt hơn về mặt diễn đạt là hoàn toàn xứng đáng – đặc biệt trong các hệ thống cần xử lý ngôn ngữ tự nhiên đa dạng và không chính tắc như chatbot hỗ trợ kỹ thuật hay hệ thống phản hồi thông minh.



**Figure 7.** Comparison Of Average Accuracy Across LoRA-Based Training Methods

Ở vòng 2, cho thấy việc ép trực giao thêm module k\_proj không mang lại kết quả quá khác biệt so với việc ép trực giao q\_proj, v\_proj. Tuy nhiên việc ép trực giao toàn bộ các module lại khiến model không chỉ phải huấn luyện năng thêm mà còn khiến chất lượng bị suy giảm rõ rệt. Chất lượng trong các câu hỏi không nằm trong tập dữ liệu được train cũng cho thấy Olora Full chỉ trả lời được 1 câu, trong khi với SoLora là 17. Điều này cho thấy rằng, việc lựa chọn module phù hợp để áp dụng trực giao hóa là yếu tố then chốt trong việc cân bằng giữa hiệu suất và khả năng tổng quát, trong khi ép toàn bộ một cách không chọn lọc có thể dẫn đến suy giảm nghiêm trọng cả về chất lượng lẫn khả năng mở rộng mô hình.

## Conclusion And Future Work

Mô Kết quả từ hai vòng huấn luyện cho thấy tầm quan trọng của việc lựa chọn có chiến lược các module để áp dụng trực giao hóa. Việc bổ sung q\_proj và v\_proj giúp tăng đáng kể chất lượng của mô hình. Ngược lại, việc ép trực giao toàn bộ các module LoRA (OLoRA-Full) lại dẫn đến hiệu suất suy giảm rõ rệt – không chỉ ở độ chính xác trung bình mà còn ở khả năng tổng quát hóa khi đánh giá trên các câu hỏi mới ngoài tập huấn luyện. Với chỉ 1/20 câu trả lời chính xác, OLoRA-Full cho thấy rằng ràng buộc quá mức có thể triệt tiêu tính linh hoạt và khả năng biểu diễn của mô hình. Trong khi đó, SoLoRA tiếp tục khẳng định tính hiệu quả khi giữ được cân bằng giữa ràng buộc trực giao và khả năng khai thác tri thức phối hợp, đạt 17-18/20 câu trả lời đúng trên tập kiểm thử mở rộng. Điều này khẳng định rằng ép trực giao cần được thiết kế có chọn lọc, vừa đủ để điều hướng mô hình học các hướng mới, nhưng không nên cực đoan đến mức làm mất khả năng thích ứng với dữ liệu ngoài phân phối.

Mô hình có thể tách riêng các task như dev/support thay vì huấn luyện trên một tập dữ liệu chung, đồng thời có thể huấn luyện thành nhiều adapter như Olora, trong khi ép trực giao một cách có chọn lọc. Để các task có khả năng chuyên biệt hoá tốt hơn nhưng vẫn giữ được khả năng chia sẻ tri thức giữa các task.

# 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] | 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]. |
| [6] | 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]. |
| [7] | "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]. |
| [8] | DeepSeek-AI, "DeepSeek-R1-Distill-Qwen-7B," [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B. |
| [9] | 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]. |
| [10] | H. Face, "Seq2Seq Parameters," [Online]. Available: https://huggingface.co/docs/autotrain/en/seq2seq\_params. [Accessed 07 Jun 2025]. |
| [11] | 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]. |
| [12] | 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]. |
| [13] | 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]. |
| [14] | 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]. |
| [15] | 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]. |
| [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]. |