International Core Journal of Engineering Volume 12 Issue 6, 2026
ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

Research on Fine-Tuning Large Language Models for Teaching Assistance in Data Science based on LoRA/QLoRA

Shaobo Xu, Pengqi Duan, Zhang Feng

North China University of Science and Technology, Tangshan 063210, China

Abstract

This paper compares LoRA and QLoRA for structured teaching assistance tasks in data science education under limited computing resources. A dataset with three tasks was built from student questionnaire data. A post-processing pipeline and evaluation metrics were designed for structured output. Experiments show that LoRA is better in training efficiency, while QLoRA performs better in JSON validity, field coverage, structure quality, and citation generation. Overall, QLoRA is more suitable for structured teaching assistance tasks in resource-constrained settings.

Keywords

Parameter-Efficient Fine-Tuning; Teaching Assistance; LoRA; QloRA.

1. Introduction

Large language models have shown strong capabilities in knowledge understanding and content generation[13]. However, they still face many challenges in deep applications within specialized domains. In higher education, students often need personalized learning path planning and diagnosis of ability gaps[11]. Such needs require the model not only to understand domain-specific problems, but also to generate suggestions and explanations with clear structure and good usability. Such needs require the model not only to understand domain-specific problems, but also to generate suggestions and explanations with clear structure and good usability[3,4,9,12].

Fine-tuning is a key step for adapting large language models to specific downstream tasks. As model size continues to grow, traditional full-parameter fine-tuning requires high computational and storage costs. This makes model customization difficult for individuals or teams with limited resources. Parameter-Efficient Fine-Tuning (PEFT) provides a practical solution to this problem. Its core goal is to achieve performance close to full-parameter fine-tuning while adjusting or adding as few parameters as possible. Parameter-Efficient Fine-Tuning (PEFT) provides a practical solution to this problem. Its core goal is to achieve performance close to full-parameter fine-tuning while adjusting or adding as few parameters as possible[1,2,7,8].

Among PEFT methods, LoRA (Low-Rank Adaptation) and QLoRA (Quantized LoRA) are two representative techniques[1,2,5,6,8]. Among PEFT methods, LoRA (Low-Rank Adaptation) and QLoRA (Quantized LoRA) are two representative techniques. This study focuses on the teaching scenario of the Data Science and Big Data Technology major, and builds an assistant model that can generate personalized and structured learning suggestions from multidimensional student profile data. Using Qwen2.5-3B-Instruct as the base model, this study conducts comparative fine-tuning experiments with LoRA and QLoRA to evaluate their differences in training efficiency and output quality.

2. Dataset Construction

(1) Data Source and Preprocessing

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

The dataset used in this study was collected from the Questionnaire on AI-Assisted Teaching and Personalized Training Needs (Student Version), which was designed for students majoring in Data Science and Big Data Technology. The raw questionnaire includes multidimensional information such as basic student information, course enrollment and feedback, self-evaluation of abilities, learning difficulties, development intentions, and open-ended responses.

After data collection, Python scripts were used to read the raw questionnaire data and perform data cleaning according to predefined field mappings, enumeration dictionaries, and desensitization rules. The cleaning process mainly included removing invalid or contradictory records, standardizing structured fields such as grade and direction, and anonymizing personal identity information.

Then, missing values in the data were processed by distinguishing structural missingness from real missingness. For example, evaluation fields corresponding to courses not taken were treated as structural missingness. For real missing values, contextual information was used for completion to obtain a more complete dataset. In addition, to enhance the model's generalization ability for open-ended questions and improve the diversity of training data, extra synthetic samples were generated with a local large language model (Ollama) based on a small number of real open-ended samples.

(2) Dataset Split and Annotation Rules

Based on the cleaned structured and semi-structured data, this study defines three core tasks in the teaching assistance scenario: Task A for personalized training plan generation, Task B for explanation of course–ability gaps, and Task C for learning question answering. The dataset was then split by task type to ensure that each task had representative samples in the training, validation, and evaluation sets.

The training set contains 78 samples in total, including 37 samples for Task A, 33 samples for Task B, and 8 samples for Task C.

The validation set contains 21 samples and is mainly used for hyperparameter tuning and early stopping during training, including 8 samples for Task A, 12 samples for Task B, and 1 sample for Task C.

The evaluation set contains 69 samples and is used for final performance comparison and quantitative analysis, including 30 samples each for Task A and Task B, and 9 samples for Task C.

After cleaning and field mapping, the raw questionnaire data were organized into profile information, a course feedback matrix, and a skills rating matrix. On this basis, instruction–input–output style samples were constructed. In this format, instruction describes the task objective for the three task families, input includes student profiles, ability ratings, course feedback, and summaries of open-ended responses, and output is generated strictly according to a JSON schema.

The output fields for Task A include plan_type, student_summary, course_recommendations, ability_gap_plan, project_practice, weekly_plan, and notes.

The output fields for Task B include task_type and gap_items.

The output fields for Task C include task_type, question, answer, citations, and safety_result.

3. Fine-Tuning

(1) Principles of the Fine-Tuning Methods

LoRA is based on the assumption that parameter changes during task adaptation lie in a low-rank subspace. During fine-tuning, LoRA freezes the original pretrained model parameters and captures task-specific updates only by introducing a small number of trainable low-rank adapter layers. Because this method can adapt the model to specific tasks with relatively low computational and storage cost while preserving most of the model's general capability, it has been widely used in resource-constrained scenarios. Because this method can adapt the model to specific tasks with relatively low computational and storage cost while preserving most of the model's general capability, it has been widely used in resource-constrained scenarios[1,2,5,8].

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

QLoRA further combines LoRA with quantization. It first quantizes the pretrained model weights into a low-precision format, which reduces model size and memory usage. In this study, the NF4 quantization format is adopted. This format compresses parameter representation while preserving the characteristics of the original value distribution as much as possible, and is more suitable for the approximately zero-mean normal distribution of neural network weights. On this basis, QLoRA does not directly update the quantized weights. Instead, like LoRA, it injects low-rank matrices into the model to complete parameter updates, thereby further reducing the number of trainable parameters. Instead, like LoRA, it injects low-rank matrices into the model to complete parameter updates, thereby further reducing the number of trainable parameters[5,8].

(2) Configuration

To ensure a fair comparison between LoRA and QLoRA, this study keeps the base model, number of epochs, learning rate, and optimization strategy consistent, while setting key hyperparameters according to the characteristics of each method. LoRA uses non-quantized loading and emphasizes training speed and parameter expressiveness. QLoRA uses 4-bit quantized loading and emphasizes training feasibility under limited GPU memory and the stability of structured outputs. The detailed training settings are shown in Table 1.

Table 1. Hyperparameter Configuration for the Two Fine-tuning Schemes

Parameter	LoRA	QLoRA
Quantization	None	4bit NF4
Max Length	512	1024
Gradient Accumulation Steps	8	16
Epochs	3	3
Learning Rate	2.0e-4	2.0e-4
Learning Rate Scheduler	Cosine	Cosine
LoRA Rank	16	8
LoRAScaling Parameter	32	16
TargetAdaptation Modules	q/k/v/o+up/down/gate	q/k/v/o

(3) Training Procedure

The training process is driven by a unified Python script that covers data preparation, model adaptation, and training monitoring, in order to ensure reproducibility and robustness.

First, the instruction-format dataset is loaded, and AutoTokenizer is used for dynamic tokenization. For the QLoRA setting, the Xet download path in Hugging Face Hub is disabled by setting the environment variable HF_HUB_DISABLE_XET=1. This avoids silent model weight download failures caused by the unstable hf-xet network library in Windows environments and improves the stability of model loading.

Then, the LoRA adapters are injected into the designated target modules with the PEFT library, and training is conducted with the Trainer API in the Transformers library. To handle rapid API changes across Transformers versions, this study dynamically checks the initialization signature of the TrainingArguments class and automatically adapts to parameter naming differences, which improves code compatibility and robustness.

In addition, validation loss is calculated every 50 steps during training to monitor convergence and potential overfitting, and the checkpoint with the lowest validation loss is saved. Complete training

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

logs, loss curves, learning rate changes, and hardware resource usage are recorded in the output directory for later analysis and reproduction.

(4) Output Pipeline Processing

To ensure usability and standardization of the generated results, this study designs a standardized inference and post-processing pipeline. During inference, the prompt explicitly requires the model to output only one valid JSON object, so as to avoid extra explanatory text. At the same time, the corresponding JSON schema description is injected according to the task type to guide the model to follow the predefined structure as much as possible.

To reduce irrelevant noise in generated text and lower the risk of parsing failure, a multi-stage cleaning and repair process is applied to the outputs. First, the raw generated text is cleaned by removing Markdown code fences, and stop words are set. Once prompt leakage is detected or the model begins to repeat itself, the output is immediately truncated.

Then, a heuristic scanning method is used to extract the text fragment most likely to form a complete JSON object. A lightweight one-pass repair is further applied, such as filling in missing brackets or quotation marks, so that the fragment can be parsed as a valid JSON object by a standard parser.

(5) Output Evalution Metrics

Table 2. Definitions of Output Evaluation Metrics

Metric Category	Metric Name	Meaning	Calculation
Training Efficiency	Training Time	Total time required to complete all training epochs	Measured in seconds, reflecting training cost
Training Efficiency	Training Throughput	Number of training samples processed per unit time	Total training samples / training time
Training Efficiency	Training Step Speed	Number of optimization steps completed per unit time	Total training steps / training time
Structured Output	JSON Pass Rate	Proportion of outputs that can be parsed into valid JSON objects	Number of valid JSON samples / total evaluation samples
Structured Output	Invalid JSON Rate	Proportion of outputs that cannot be parsed into valid JSON	Number of invalid JSON samples / total evaluation samples
Structured Output	Key Field Coverage Rate	Proportion of outputs with complete top-level key fields	Number of samples with complete key fields / total evaluation samples
Structured Output	Structure Pass Rate	Proportion of outputs that pass structural validation when key blocks are non-empty and field types are correct	Number of structurally valid samples / total evaluation samples
Field-Level Alignment	Precision	Proportion of correctly predicted fields among all predicted fields	TP / (TP + FP)
Field-Level Alignment	Recall	Proportion of correctly predicted fields among all labeled fields	TP / (TP + FN)
Field-Level Alignment	F1-score	Harmonic mean of Precision and Recall	2PR / (P + R)
Field-Level Alignment	Macro-F1	Average of F1 scores across fields	Reflects overall balance across fields
Field-Level Alignment	Micro-F1	F1 computed after aggregating TP, FP, and FN across all fields	Better reflects overall prediction performance
Task-Specific Metric	Citation Coverage	Proportion of Task C outputs with non-empty and valid citations fields	Number of valid citation samples / total Task C samples

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

To compare the performance differences between LoRA and QLoRA in structured generation tasks for teaching assistance in a more comprehensive and objective way, this study designs an evaluation system from both the training stage and the inference stage. At the training stage, the main metrics are training time, training throughput, and training step speed, which reflect the efficiency differences between the two fine-tuning methods. At the inference stage, several metrics are designed around the parsability, completeness, and content alignment of structured outputs, including JSON pass rate, key field coverage rate, structure pass rate, and field-level Precision, Recall, and F1. In addition, considering the need for interpretability in the question-answering scenario, citation coverage is further calculated for Task C. This evaluation system reflects not only the training cost of the model but also the reliability of its outputs in actual system calls.

4. Experimental Results and Analysis

(1) Comparison of Experimental Results Between LoRA and QLoRA

To compare the two fine-tuning methods in terms of training efficiency and structured output quality, this study reports training efficiency metrics, structured output metrics, field-level alignment metrics, and task-specific metrics. The detailed results are shown in Table 3.

Table 3. Comparison of Experimental Results Between LoRA and QLoRA

Metric Category	Metric Name	LoRA	QLoRA	Better Performer
Training Efficiency	Training Time (s)	189.63	390.80	LoRA
Training Efficiency	Training Throughput (samples/s)	1.234	0.599	LoRA
Training Efficiency	Training Step Speed (steps/s)	0.158	0.038	LoRA
Structured Output	JSON Pass Rate	0.1304	0.6667	QLoRA
Structured Output	Key Field CoverageRate	0.1304	0.6667	QLoRA
Structured Output	Structure Pass Rate	0.1304	0.6667	QLoRA
Structured Output	Invalid JSON Rate	0.8696	0.3333	QLoRA
Field-Level Alignment	Macro-F1	0.0095	0.2173	QLoRA
Field-Level Alignment	Micro-F1	<0.03	0.2245	QLoRA
Task-Specific Metric	Citation Coverage (TaskC)	0	77.78%	QLoRA

As shown in Table 3, LoRA has a clear advantage in training efficiency. Its training time is shorter, its throughput is higher, and its training step speed is faster. These results indicate that LoRA is more efficient in terms of computational cost and training speed under the current experimental setting.

However, as shown in Fig. 1, QLoRA performs much better on the key metrics related to structured output quality. Its JSON Pass Rate, Key Field Coverage Rate, and Structure Pass Rate all reach 0.6667, while the corresponding values of LoRA are only 0.1304. At the same time, the Invalid JSON Rate

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

of QLoRA is 0.3333, which is much lower than the 0.8696 of LoRA. This shows that QLoRA generates outputs with more stable structure and better format compliance, making the results more suitable for direct use in downstream systems.

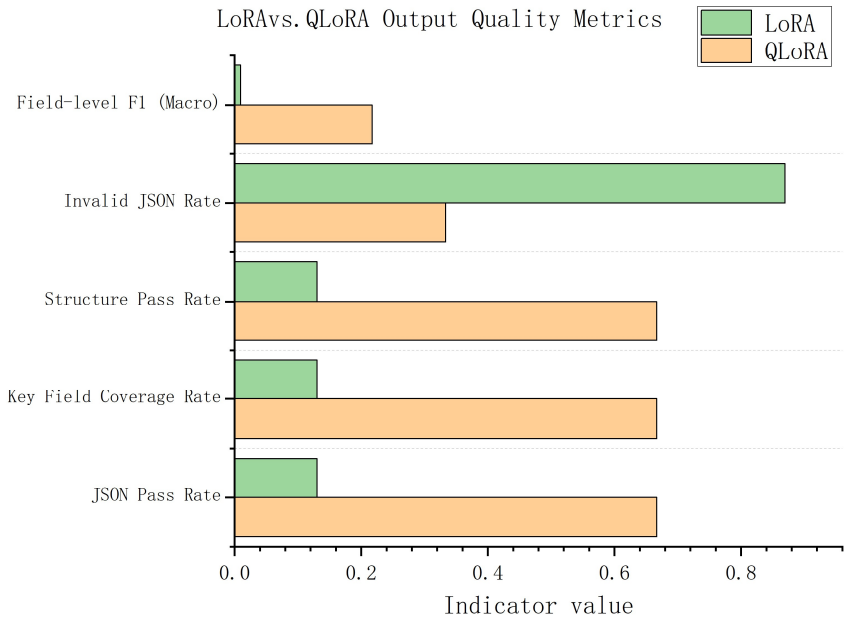

Fig. 1 Comparison of output quality metrics between LoRA and QLoRA.

From the field-level alignment results, QLoRA also outperforms LoRA by a large margin. Its Macro-F1 reaches 0.2173, while LoRA only achieves 0.0095. The Micro-F1 of QLoRA is 0.2245, which is also much higher than that of LoRA. These results indicate that QLoRA not only produces outputs with better structural completeness, but also performs better in key information alignment and content matching.

In Task C, Citation Coverage is used to measure whether the generated answers include valid supporting references. The Citation Coverage of LoRA is 0, which means it fails to generate compliant citation fields under this setting. In contrast, QLoRA reaches 77.78% on this metric, which suggests that its outputs have stronger interpretability and traceability in the learning question-answering scenario.

Overall, the results show a clear trade-off between efficiency and quality. LoRA is more efficient during training, whereas QLoRA is more reliable in structured generation. For the teaching assistance scenario in this study, output parsability, structural completeness, and content reliability are more important than single-run training speed. Therefore, although QLoRA requires more training time, it is the more suitable fine-tuning method for this task.

(2) Typical Case Analysis

To more clearly demonstrate the actual output performance of the model in different tasks, this study selects representative examples from Task A, Task B, and Task C for analysis. Since the original inputs and outputs are all structured JSON data, the key input information and core output results of each task are summarized in Table 4. On this basis, the output characteristics and application value of the model in the three task types are further discussed.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

Table 4. Typical Input and Output Examples of the Three Teaching Assistance Tasks.

Task Type	Key Input	Core Output
Task A: Personalized Training Plan Generation	Sophomore student; preferred directions are "Artificial Intelligence" and "Data Analyst"; learning difficulties include "too many concepts, fragmented system, hard to connect" and "weak mathematical foundation"; relevant courses have been taken	Recommended courses include "Machine Learning Algorithms" and "Data Mining"; ability improvement goals include "understanding machine learning algorithms" and "data mining techniques"; the project practice goal is "complete a full cycle from data preparation to result reporting"; a weekly learning plan is provided
Task B: Explanation of Course–Ability Gaps	Junior student; preferred direction is "Data Analyst"; learning difficulties include "can understand in class but cannot solve problems or write code" and "weak mathematical foundation"; completed courses include "Introduction to Data Science and Engineering," "Mathematical Foundations of Data Science," and "Machine Learning"	Identifies "course–ability gap" as the core problem; provides related courses, evidence descriptions, cause explanations, and improvement suggestions
Task C: Learning Question Answering	Junior student; preferred direction is "Data Analyst"; low ratings in statistics and machine learning; question: "I often cannot understand loss functions and gradient descent in the machine learning course. How should I study them?"	Provides the learning suggestion of "first strengthen linear algebra and calculus basics, then understand the principles, and finally reinforce learning through programming practice"; preserves the citations field

1) Analysis of task A results: As shown in Table 4, the input information of Task A mainly includes grade, development direction, learning difficulties, ability ratings, and course feedback. Correspondingly, the model output provides not only course recommendations but also ability improvement goals, project practice suggestions, and a weekly study plan. In this example, the student has problems such as “too many concepts, fragmented system, hard to connect” and “weak mathematical foundation.” In the output, the model recommends courses such as “Machine Learning Algorithms” and “Data Mining,” and sets the project goal of “completing a full cycle from data preparation to result reporting.”

This shows that the model can generate a relatively complete training plan around the student’s interests and core weaknesses rather than simply repeating the input information. This indicates that the model has good structured integration ability in the task of personalized training plan generation.

2) Analysis of task B results: Task B focuses more on explaining learning problems. The input of this task includes learning difficulties, ability ratings, and course enrollment information, while the output includes gap name, gap level, related courses, evidence descriptions, cause explanations, and improvement suggestions. As shown in the example in Table 4, the model can identify “course–ability gap” as the core problem according to issues such as “can understand in class but cannot solve problems or write code” and “weak mathematical foundation,” and associate this problem with courses such as “Introduction to Data Science and Engineering,” “Mathematical Foundations of Data Science,” and “Machine Learning.”

At the same time, the model further provides cause explanations and improvement suggestions. This indicates that the model is able to summarize ability weaknesses from course performance. Such

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

outputs help teachers and students locate key problems more quickly and can also be directly used by a teaching assistance system.

3) Analysis of task C results: Task C is mainly used for answering learning questions. The input includes basic student profile information, partial ability information, and a specific question, while the output includes the answer content, citation fields, and safety results. As shown in the example in Table 4, for the question "I cannot understand loss functions and gradient descent in the machine learning course," the model provides the learning suggestion of "first strengthen the basics, then understand the principles, and finally consolidate learning through programming practice." The answer is therefore highly targeted.

More importantly, the output preserves the citations field. This indicates that the model retains a certain source basis while giving suggestions. Compared with ordinary open-ended responses, this structured output format is more suitable for teaching assistance scenarios and better reflects the interpretability and traceability of the generated results.

(4) Discussion

Under strict resource constraints, LoRA and QLoRA show a clear trade-off between efficiency and quality. QLoRA uses about twice the training time in exchange for clear advantages in output parsability, structural completeness, and content alignment.

From the results, QLoRA sacrifices training efficiency but gains higher output stability and quality. Although 4-bit quantization reduces training speed, it also significantly reduces GPU memory usage during model loading, which allows the model to use longer context length under the same hardware conditions. This provides the necessary support for handling complex instructions and generating complete JSON structures.

In contrast, LoRA is more efficient in training but weaker in format compliance. The full-precision loading strategy gives it an advantage in training speed, but under the configuration of this study, the shorter context length may limit its ability to understand and execute complex formatting instructions.

For teaching assistance systems, the parsability and structural reliability of generated results are more important than the speed of a single training run. Therefore, although QLoRA is slower in training, its clear advantage in structured generation quality makes it a better and more reliable choice for this scenario. For teaching assistance systems, the parsability and structural reliability of generated results are more important than the speed of a single training run[9,10].

5. Conclusion

This study focuses on professional teaching assistance scenarios and uses Qwen2.5-3B-Instruct as the base model to systematically compare the performance differences between LoRA and QLoRA in structured content generation tasks for large language models. In addition, an output post-processing and evaluation system centered on JSON validity and field alignment quality is designed.

The experimental results show that although LoRA has an advantage in training efficiency, QLoRA achieves clear improvement in structured output quality with the support of longer context enabled by 4-bit quantization. This indicates that QLoRA is more suitable for generating stable, parsable, and directly integrable structured outputs for teaching assistance systems. Therefore, for application scenarios in education and other specialized domains that require output stability and standardization under limited computing resources, QLoRA is the better fine-tuning choice. This study also provides a complete empirical framework for such scenarios, covering dataset construction, fine-tuning practice, and task-specific evaluation.

References

[1] Ding, N., Qin, Y., Yang, G., et al. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, *5*(3), 220–235. https://doi.org/10.1038/s42256-023-00647-9

ISSN: 2414-1895 DOI: 10.6919/ICJE.202606_12(6).0002

[2] Han, Z., Gao, C., Liu, J., et al. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. arXiv preprint arXiv:2403.14608.

[3] Anisuzzaman, D. M., Malins, J. G., Friedman, P. A., et al. (2025). Fine-tuning large language models for specialized use cases. *Mayo Clinic Proceedings: Digital Health*, *3*(1), 100184. https://doi.org/10.1016/j.mcpdig.2024.100184

[4] Tinn, R., Cheng, H., Gu, Y., et al. (2023). Fine-tuning large neural language models for biomedical natural language processing. *Patterns*, *4*(4). https://doi.org/10.1016/j.patter.2023.100700

[5] Patil, R., Khot, P., & Gudivada, V. (2025). Analyzing LLAMA3 performance on classification task using LoRA and QLoRA techniques. *Applied Sciences*, *15*(6), 3087. https://doi.org/10.3390/app15063087

[6] Susilo, A., Christanti, V., & Lauro, M. D. (2024). Fine-Tuning LLaMA-2-Chat untuk ChatBot Penerjemah Bahasa Gaul menggunakan LoRA dan QLoRA. *MIND (Multimedia Artificial Intelligent Networking Database) Journal*, *9*(2), 248–260.

[7] Zhang, D., Feng, T., Xue, L., et al. (2025). Parameter-efficient fine-tuning for foundation models. arXiv preprint arXiv:2501.13787.

[8] Xu, L., Xie, H., Qin, S. J., et al. (2026). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[9] Chu, Z., Wang, S., Xie, J., et al. (2025). Llm agents for education: Advances and applications. arXiv preprint arXiv:2503.11733.

[10] Wang, S., Chen, F., Chen, G., et al. (2025). Using Local LLM Tools to Optimize Chinese High School Chemistry Education: Practice, Challenges, and Future Directions. *Journal of Chemical Education*, *102*(10), 4368–4375. https://doi.org/10.1021/acs.jchemed.5c00359

[11] Hsieh, P. J., Chen, C. C., & Liu, W. (2019). Integrating talent cultivation tools to enact a knowledge-oriented culture and achieve organizational talent cultivation strategies. *Knowledge Management Research & Practice*, *17*(1), 108–124. https://doi.org/10.1080/14778238.2018.1548762

[12] Harry, A., & Sayudin, S. (2023). Role of AI in Education. *Interdisciplinary Journal and Humanity (INJURITY)*, *2*(3), 260–268.

[13] Beck, J., Stern, M., & Haugsjaa, E. (1996). Applications of AI in Education. *XRDS: Crossroads, The ACM Magazine for Students*, *3*(1), 11–15.