

Article

Optimization and Application of Generative AI Algorithm Based on Transformer Architecture in Adaptive Learning

Xuan Liu ¹ and Zhi Li ^{2,*} 
¹ School of Education Sciences, Xinyang Normal University, Xinyang 464000, China; 13937673710@163.com

² School of Media and Law, NingboTech University, Ningbo 315000, China

* Correspondence: lizhi@nbt.edu.cn

Abstract

At present, generative AI has problems of insufficient content generation accuracy, weak personalized response, and low reasoning efficiency in adaptive learning scenarios, which limit its in-depth application in intelligent teaching. To solve this problem, this paper proposed a Transformer fine-tuning method based on low-rank adaptation technology, which realized efficient parameter update of pre-trained models through low-rank matrix insertion, and combined the instruction fine-tuning strategy to perform domain adaptation training on the model for the constructed educational scenario dataset. At the same time, a dynamic prompt construction mechanism was introduced to enhance the model's context perception ability of individual learners' behaviors, thereby achieving precise alignment and personalized control of generated content. This paper embeds the "wrong question guidance" and "knowledge graph embedding" mechanisms in the model, provides intelligent feedback based on student errors, and promotes in-depth understanding of subject knowledge through knowledge graphs. Experimental results showed that this method scored higher than 0.9 in BLEU and ROUGE-L. The average response delay was low, which was significantly better than the traditional fine-tuning method. This method showed good adaptability and practicality in the fusion of generative AI and adaptive learning and provided a generalizable optimization path and application solution for intelligent education systems.

Keywords: generative artificial intelligence; Transformer architecture; low-rank adaptation; adaptive learning; dynamic prompt engineering



Academic Editors: Ruobin Gao,
Maohan Liang and Xiaocai Zhang

Received: 3 November 2025

Revised: 8 January 2026

Accepted: 13 January 2026

Published: 13 January 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

In recent years, the rapid development of generative artificial intelligence has promoted the evolution of intelligent systems in various fields. Educational technology, as a key application direction, is facing the challenge of transforming from static content distribution to dynamic personalized teaching [1,2]. Adaptive learning has attracted widespread attention because it can dynamically adjust teaching strategies according to learner differences. Its core lies in accurately understanding learner needs and generating personalized learning resources in real time [3,4]. Generative AI based on the Transformer architecture provides technical support for efficient and intelligent teaching interaction. Exploring its deep integration in adaptive learning has important theoretical value and practical significance for improving teaching efficiency and optimizing learning experience [5,6].

Although existing research has initially applied generative AI to educational scenarios, there are still key problems in its specific implementation [7,8]. First, the pre-trained

large model lacks sufficient adaptation to the language and knowledge structure in the field of education, resulting in low subject accuracy and cognitive fit of the generated content [9,10]. Second, the lack of personalized generation capabilities makes it difficult to provide targeted learning resources based on individual differences among students [11,12]. Third, the high cost of model inference limits its deployment and response efficiency in real-time teaching systems [13,14]. These problems seriously restrict the in-depth application of generative AI in adaptive learning systems [15,16].

By combining generative AI with education systems, we can better respond to diverse student needs, tailor learning content and paths for them, and thus provide a more effective learning experience [17,18]. To improve the performance of generative models in specific tasks, some studies have introduced efficient parameter fine-tuning methods, such as Adapter, Prefix Tuning, and LoRA (Low-Rank Adaptation). Among them, LoRA has attracted widespread attention due to its low computational overhead and high transfer efficiency [19,20]. At the same time, instruction fine-tuning has been proven to significantly improve the task perception ability of the model, and dynamic prompt engineering has performed well in achieving contextual personalized generation [21,22]. However, these methods are mostly focused on general language processing tasks, and their integrated application in educational scenarios is still immature, lacking systematic optimization for learner behavior modeling and targeted generation of teaching content [23,24]. To this end, this paper combines LoRA fine-tuning technology with a dynamic prompt mechanism to construct a generative AI optimization path suitable for adaptive learning scenarios [25,26].

The in-depth application of generative artificial intelligence in adaptive learning faces several fundamental scientific challenges. These challenges revolve around how to accurately adapt general-purpose models to the specialized language and knowledge structures of the educational field, how to achieve real-time personalized content generation based on individual learner differences, and how to maintain high-efficiency reasoning in resource-constrained environments. This research aims to address key research questions, including exploring an efficient fine-tuning path for Transformer models based on low-rank adaptation; designing optimization strategies that integrate teaching instructions to enhance the model's understanding of educational tasks; and constructing a prompting mechanism that can dynamically generate personalized guidance based on learner states. This research is essentially an interdisciplinary field between educational science and artificial intelligence. It goes beyond optimizing model algorithms, involving computational modeling of learning behavior, cognitive patterns, and knowledge transfer processes, embodying an interdisciplinary methodology for empowering educational reform with intelligent technology.

This paper aims to improve the personalized generation quality and response efficiency of generative AI in adaptive learning scenarios and proposes a Transformer model optimization method that integrates low-rank adaptation, instruction fine-tuning, and dynamic prompt construction. First, the low-rank adaptation technology is used to fine-tune the parameters of the pre-trained model by injection, and the effective transfer of teaching language style and knowledge expression is achieved on a large-scale educational scenario dataset. Then, instruction fine-tuning is used to further enhance the model's understanding and generation capabilities for diverse educational tasks (mathematical problem solving, language learning). This paper also optimizes the Transformer architecture itself, adjusting the attention mechanism and position encoding to adapt to the unique contextual requirements of the education field, thereby improving the accuracy and subject fit of the model when generating content. Finally, a dynamic prompt construction mechanism is introduced to optimize the personalized matching ability of generated content by modeling the learner context. The novelty of this method lies in its systematic integration and structural optimization for adaptive learning scenarios. The standard LoRA fine-tuning and

instruction fine-tuning processes have been redesigned to embed semantic role annotation, task embedding vectors, and a dynamic prompt generation mechanism based on knowledge graphs and error types, all specific to the education domain. The collaborative work of these components aims to address the issues of subject-specific accuracy, personalized matching, and cognitive fit in educational content generation, rather than simply combining general techniques.

To systematically address the aforementioned issues, this paper unfolds according to the following structure. Chapter 2 elucidates the Transformer-based generative artificial intelligence optimization method, detailing the construction and preprocessing of educational datasets, laying the foundation for domain adaptability in model training; it introduces a Transformer fine-tuning technique based on low-rank adaptation, aiming to achieve efficient updating of model parameters and knowledge transfer; it describes instruction fine-tuning strategies to enhance the model's understanding and execution capabilities for diverse teaching tasks; and it proposes a dynamic prompt construction mechanism to achieve perception of learners' personalized contexts and control of content generation. Chapter 3 evaluates the adaptive generation performance, verifying the effectiveness of the proposed method through multi-dimensional experiments on generated content quality, personalization matching degree, multi-task adaptability, response time, and resource utilization. Chapter 4 summarizes the entire work, discussing the contributions and limitations of this method and providing an outlook on future research directions. Each chapter is interconnected, aiming to comprehensively present and demonstrate the optimization path of this research, from method construction and implementation to verification.

2. Transformer-Based Generative AI Optimization Method

This methodology partially covers the complete technical process from data preparation to model optimization. The construction and preprocessing of educational datasets are fundamental to subsequent model adaptation, aiming to provide structured, high-quality domain corpora. Low-rank adaptation-based Transformer fine-tuning is the core technology for achieving efficient transfer of pre-trained models. Instruction fine-tuning further endows the model with the ability to understand and execute specific teaching tasks. The dynamic prompting mechanism is responsible for transforming the learner's individual state into contextual guidance for the generation process, ultimately achieving personalized content generation.

2.1. Data Construction and Preprocessing in the Education Field

To achieve effective migration and fine optimization of generative AI models in adaptive learning, this paper first constructs a structured, high-quality educational corpus. The data sources mainly include digital course resource platforms, subject question bank resources, real student question and answer logs, and teacher annotation texts.

The educational corpus constructed by this research institute is sourced from several mainstream digital education platforms in China, covering three subjects: mathematics, physics, and English, and encompassing curriculum content from junior high to senior high school. The total data comprises approximately 150,000 text fragments, including textbook explanations, exercise stems, student answers, and teacher comments. The subject distribution is even; the grade level coverage is complete; the texts are anonymized to preserve the original teaching context and knowledge structure.

First, during the data collection phase, an automated web crawler framework is used in conjunction with API (Application Programming Interface) calls to capture raw text data from a highly trusted educational platform. Content filtering rules are then set to retain only segments related to course instruction, student answers, and teacher explanations, while

excluding non-teaching corpora. Meanwhile, natural language keyword extraction tools are used to roughly classify the original texts into three categories: teaching explanation, question–answer interaction, and task guidance, in order to improve the efficiency of subsequent processing.

Secondly, in the sample screening stage, TF-IDF (Term Frequency-Inverse Document Frequency) and similarity clustering algorithm (using cosine similarity) are used to identify and remove redundant, templated or highly repetitive texts to ensure data diversity and breadth of expression. At the same time, the BERT embedding vector is introduced to perform semantic consistency verification. For text pairs above the similarity threshold (set to 0.92), only those with more complete expressions are retained, as shown in Figure 1.

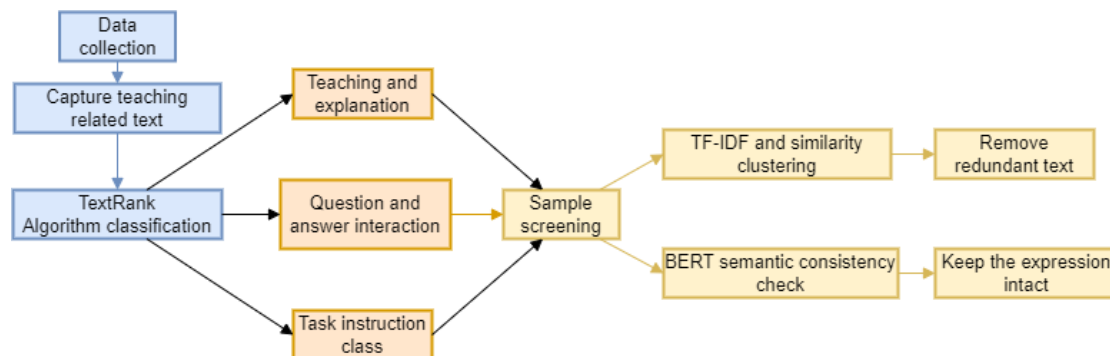


Figure 1. Core steps of data collection and sample screening.

Figure 1 shows the core steps of data collection and sample screening. The data collection stage aims to obtain high-quality teaching-related texts and roughly classify them through the TextRank algorithm. The sample screening stage uses TF-IDF and similarity clustering technology, combined with the BERT (Bidirectional Encoder Representations from Transformers) model, to further screen the data to ensure that useful, non-redundant, and highly semantically consistent texts are retained.

In the semantic hierarchical annotation stage, a weakly supervised annotation mechanism is introduced to divide the teaching text into four semantic roles: knowledge point statement, teaching intention description, task guidance language, and student answer fragments [27,28]. CRF (Conditional Random Field) is combined for sequence labeling task training, and manual review is used to ensure label consistency. In the cleaning and normalization stage, the focus is on dealing with problems such as non-standard expressions, incoherent texts, and colloquial language. First, the language model perplexity index is used to filter low-quality texts (texts with a threshold of Perplexity > 120 can be eliminated) [29,30]. Secondly, a text normalization rule library is introduced to uniformly transcribe common abbreviations, typos, and redundant expressions to ensure that the corpus format is consistent. For formula content, a parser is used to unify the expression form to avoid the model from misjudging mathematical structure information during training.

Finally, the structured coding process is entered to build a unified input format template. Each sample is organized into a multi-field JSON (JavaScript Object Notation) structure, with fields including “instruction” (task instruction), “input” (context information), “target” (reference output), and “metadata” (knowledge point number, grade, and subject category label information). In addition, hash coding is used to construct an anonymous student identity index, linking historical answer records with learning behavior labels, and providing a basic semantic vector for subsequent dynamic prompt construction.

2.2. Transformer Fine-Tuning Based on LoRA (Low-Rank Adaptation)

This study selects LLaMA-2-13B as the backbone network for its methodology. This model is a pre-trained Transformer decoder with an open structure and full parameter access. It has 13 billion parameters, and its architectural details and weights are publicly available, allowing for low-level parameter manipulation and adaptation layer injection. The model was chosen based on its recognized performance in complex instruction compliance and contextual understanding, providing a necessary semantic understanding foundation for subsequent prompting engineering and contextual adaptation in educational scenarios. Fine-tuning is achieved through adaptation layer injection, without directly modifying the weights of the original model. Instead, trainable low-rank adaptation modules are inserted in parallel alongside specific linear layers [31,32].

The first stage is model freezing and target layer selection. The original pre-trained model weights (using FP16 precision) are loaded, and only the query (Q) and value (V) weight matrices of all attention modules in the Transformer are selected as the injection targets. All other parameters are frozen and do not participate in training. This selection is based on the results of previous experiments, which show that the main impact of educational tasks on the quality of model generation is concentrated in the information selection and output construction links, corresponding to the action paths of the Q and V matrices, respectively. The number of model layers here is set to 24, and 2 sets of LoRA modules are injected into each layer.

The second stage is the injection and structure construction of the LoRA module. The original Attention weight matrix $W_{q,v} \in \mathbb{R}^d \times k$ is replaced with

$$W_{q,v}' = W_{q,v} + \Delta W_{q,v}$$

The low-rank matrix $\Delta W_{q,v} = BA$ has a size of $A \in \mathbb{R}^r \times k$, $B \in \mathbb{R}^d \times r$, and a rank of $r = 8$. Parameter initialization: Matrix A is initialized with He, and matrix B is initialized with Xavier to prevent gradient explosion and disappearance. During training, only matrices A and B are updated, and the rest of the weights remain frozen.

The third stage is task adaptation training. Based on the aforementioned educational corpus, the supervised fine-tuning strategy is used to train the inserted LoRA module. The loss function uses standard cross-entropy; the optimizer selects AdamW; the learning rate combines the linear warmup and cosine decay learning rate scheduling strategies. During the training process, a dynamic segmented input window (window size ranges from 512 to 2048 tokens) is used to adapt to teaching task samples of different lengths, thereby ensuring that the model can stably model contextual dependencies. The training is conducted for 30 epochs, each with a batch size of 64, and the training time is about 16 h, as shown in Figure 2.

In the first 5 epochs, the loss value gradually decreases. This is because at this stage, the learning rate is gradually increased from a smaller value, and the model needs to gradually adapt to the task at the beginning, showing a higher training error. In the early stage, as the learning rate gradually increases, the training effect of the model gradually improves, and the loss gradually decreases. After the Warmup phase, the learning rate enters the Cosine Decay phase, which means that the learning rate can gradually decrease. The loss value continues to decrease, but the rate of decrease becomes slower and slower. This is because the model is close to its optimal solution, and further reducing the error requires more refined adjustments. Even so, due to the model's increased stability, the loss tends to stabilize. In the later stages of training (that is, when approaching the 30th epoch), the loss value changes very little, indicating that the model has reached a stable state. In the warmup phase, the learning rate gradually increases from the initial small value until it reaches the predetermined maximum learning rate. The linear growth of

the learning rate enables the model to find suitable parameters better in the early stage and improve the stability of training. After the warmup ends, the learning rate begins to enter the cosine decay phase and gradually decreases. Figure 2 records the evolution of the cross-entropy loss value over 30 training epochs. The loss value is calculated based on the difference between the model's prediction and the true label, obtained through a standard forward and backward propagation process. The loss value for each training step is recalculated by the optimizer after updating parameters based on gradients, and recorded in real-time to the log system. The curves in Figure 2 are constructed based on these continuously recorded loss values. The verification of the curve's correctness relies on the inherent computational correctness of the training framework, including the correct implementation of the loss function, the numerical stability of gradient calculation, and the mathematical consistency of optimizer updates. No gradient explosion or abnormal fluctuations in the loss value occurred during training, indicating that the computation process was effective.

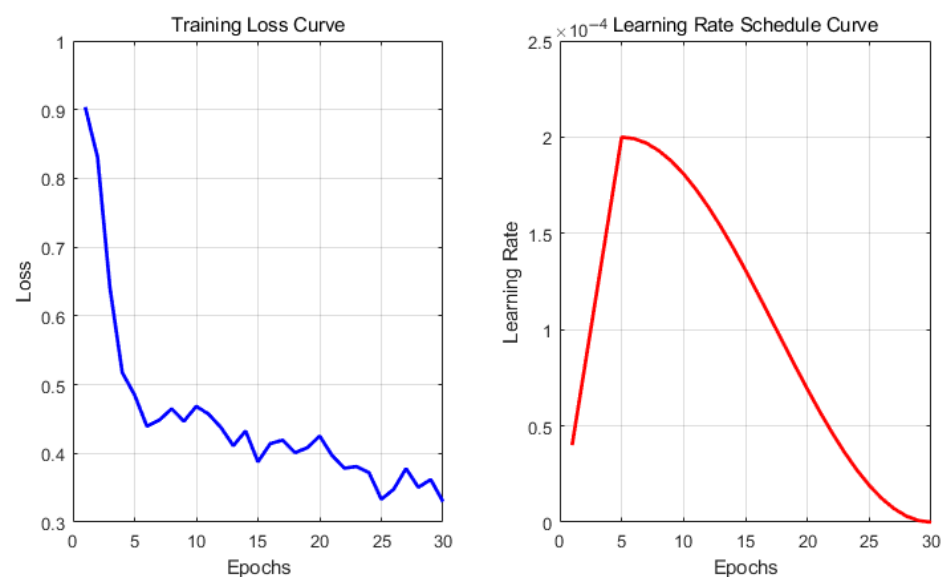


Figure 2. Changes in loss function.

To prevent target drift and expression degradation during fine-tuning, a variety of regularization enhancement mechanisms are introduced in training. Specific measures include (1) L2 regularization is introduced for the inserted LoRA parameter, with a weight set to 5×10^{-5} . (2) A Frozen Layer Activation Check is performed every 500 steps. By calculating the KL divergence between the LoRA output and the frozen backbone output, the fine-tuning direction of the model is monitored to see if it deviates from the pre-trained semantic space. (3) A gradient clipping mechanism is used, and the clipping threshold is set to 1.0 to prevent local gradient explosion. These measures ensure the stability of training and the consistency of generated content, further improving the performance of the model in educational tasks.

To further improve the model's ability to follow instructions in an educational context, this study embeds a lightweight instruction preprocessing module in the LoRA fine-tuning process. This module uses a GPT-tokenizer to structure the input teaching task description and introduces explicit task labels (such as [TASK:EXPLAIN], [TASK:Q&A]) as additional prompt prefixes, thereby significantly improving the LoRA module's ability to identify task boundaries and generate targets. In the specific implementation process, the preprocessing module first parses the input teaching task description and automatically adds corresponding task tags according to the task type (such as explanation, question answering, and

exercise generation). These tags provide clear contextual guidance for the model, enabling it to understand the goals and constraints of the current task more accurately. For example, the tag [TASK:EXPLAIN] is used to prompt the model to generate a detailed explanation of a certain knowledge point, while [TASK:Q&A] is used to guide the model to generate a dialogue form of questions and answers. Through this structured task guidance, the model can more clearly identify task requirements and adjust the style and structure of generated content according to the label. By introducing this instruction preprocessing mechanism, the model can better understand the task objectives when processing educational tasks, thereby generating more accurate and targeted content.

2.3. Instruction Fine-Tuning Improves Task Adaptability

This paper designs and implements a set of structured task instruction sets, combined with instruction fine-tuning strategies, to clearly guide the model to generate content that meets the teaching objectives. The whole process covers the key links of task instruction template construction, instruction–input pair construction, and fine-tuning training implementation.

First, a unified structured task instruction template is constructed based on the type of educational task. The template strictly follows the three elements of “task type, input requirements, and expected output format”. Examples include “Please generate five medium-difficulty math multiple-choice questions based on the following knowledge points”, “Please explain the following physical concepts and provide application examples”, etc. The template text uses concise and clear language to avoid ambiguity and ensure that the model can accurately identify the task objectives. A total of 6 types of task instructions are designed, covering key teaching scenarios such as exercise generation, knowledge point explanation, Q&A, and wrong question analysis.

Secondly, the instruction and input data pairs are constructed using automated script processing that structures the teaching content in the educational corpus into a triple format of “instruction, input, output”. Among them, the “instruction” field corresponds to the above-mentioned task template; the “input” field includes the knowledge point description, context text, and student answer fragments; the “output” field is the corresponding teaching-generated content, including question text, concept explanation or feedback comments, as shown in Table 1. The original text is accurately segmented and matched through regular expressions and semantic matching algorithms to ensure semantic consistency and prevent information confusion. Finally, 124,000 structured training samples are generated.

Table 1. Sample examples.

Sample ID	Task Template	Instruction	Input	Output
1	Concept Explanation	[TASK:EXPLAIN]	Computer Architecture	Computer architecture refers to the hardware structure and working principle of a computer.
2	Question Generation	[TASK:CREATE_Q]	Math problems, Solving Quadratic Equations	Solve the equation $ax^2 + bx + c = 0$.
3	Q&A Response	[TASK:Q&A]	Student asks, “What is an algorithm?”	An algorithm is a series of steps and rules to solve a problem.
4	Concept Explanation	[TASK:EXPLAIN]	Deep Learning, Neural Network Principle	Deep learning is a subfield of machine learning that uses multi-layer neural networks for feature learning.
5	Question Generation	[TASK:CREATE_Q]	Chinese Essay, Describing Spring	Write an essay about spring.
6	Q&A Response	[TASK:Q&A]	Student asks, “How to improve speaking skills?”	Ways to improve speaking skills include listening more, speaking more, and conversing with native speakers.

Table 1 shows how to convert the teaching content in the educational corpus into a structured format of “instruction, input, output” triples. Each triple represents a complete teaching task, and the task template represents the type of task, such as concept explanation, exercise generation, or question-answering feedback. Instructions are specific prompts for task execution, telling the model what to do, such as [TASK:EXPLAIN] for explaining a concept, and [TASK:CREATE_Q] for generating a question. The input contains the background information of the task, including the description of knowledge points and the student’s answer fragments, which provide the background required for the task. The output is the expected result of the task, which is the content generated by the model, such as the detailed explanation of the concept, the questions generated by the exercises, or the specific answers to the Q&A feedback. The development of Table 1 is based on the need for structured data representation required by the instruction fine-tuning strategy. The table’s construction stems from transforming raw teaching texts into a “command–input–output” triple format that the model can process. The sample examples in the table are generated directly from the constructed educational domain corpus. Data collection relied on automated scripts that parsed and aligned the teaching texts in the corpus based on predefined regular expressions and semantic matching algorithms. The accuracy of the data content was recorded and ensured through a two-stage process: the first stage, rule matching executed by the script, ensured the accuracy of field segmentation; the second stage involved manual verification through sampling to confirm semantic consistency, and the verification results were recorded to demonstrate the reliability of the data transformation. The samples presented in Table 1 demonstrate structured examples of different teaching task types. Their scientific purpose is to concretely illustrate the data organization in the instruction fine-tuning stage and reflect the logical relationship between task instructions, contextual inputs, and target outputs, thereby laying a data foundation for the model to understand and execute diverse teaching tasks.

The third step is to load the insertion module based on LoRA fine-tuning into the pre-trained Transformer model and fine-tune it with the constructed instruction dataset. The model training input sequence is composed of “[task instruction] + [input content]”, and the output target is the corresponding teaching text. The cross entropy is used as the objective function; the optimizer is AdamW; the learning rate is set to 1×10^{-4} ; the batch size is 64; the training cycle is 3 epochs. To improve the generalization ability of multi-tasks, samples of different task types are randomly sampled during training to avoid overfitting a single task. The input token length is dynamically adjusted, and a maximum of 1024 tokens is supported.

In addition, in view of the characteristics of multi-task learning, this study introduces a task embedding vector in the Transformer structure. This vector is superimposed with the input token embedding and input into the model together to enhance the model’s perception of the task context. The role of the task embedding vector is to provide an independent identifier for each input task, so that the model can adaptively adjust its generation strategy according to different task types (such as exercise generation, concept explanation, and question answering). The vector is randomly initialized at the beginning of training and is optimized along with the model parameters during training. In this way, the task embedding vector gradually learns how to switch between different tasks, thereby improving the model’s recognition of task instructions and the targeted generation.

2.4. Dynamic Prompt Construction to Achieve Personalized Generation

Aiming at the needs of personalized teaching, this study designs and implements a dynamic prompt construction mechanism based on student behavior data and knowledge mastery, aiming to guide the generative AI model to produce teaching content that is

highly consistent with the individual characteristics of learners through refined contextual prompts. The mechanism covers three parts: data collection and feature extraction, dynamic prompt template design, context splicing and real-time update. It combines deep learning and rule engines to achieve precise control of the adaptive generation environment.

First, a multi-dimensional behavioral feature vector is constructed for individual students. The data sources include students' previous answer records, wrong question types, learning time, learning frequency, and feedback evaluation. Time series preprocessing methods are used for normalization to eliminate abnormal data points. Based on this data, a Transformer-based behavior encoder is applied to extract high-dimensional implicit state vectors. The encoder uses a multi-layer self-attention network to achieve information fusion within the time step, effectively capturing changes in learning behavior and dynamics of knowledge mastery. The encoder parameters adopt a pre-training and fine-tuning strategy, sharing some weights with the overall teaching task model to ensure a close connection between feature expression and generation tasks.

Secondly, a dynamic Prompt template is designed based on the current learning goals and knowledge point mastery. The template consists of multiple structured text segments, including three modules: "Learner status description", "Current knowledge point summary", and "Task instruction prompt". The content of each module is dynamically generated based on the real-time feature vector: the learner status description module maps the output vector of the behavior encoder to a text label (such as "Basic concepts have been mastered", "Need to strengthen application practice"). The knowledge point summary module calls the knowledge graph interface to obtain relevant knowledge units and difficulty levels. The task instruction prompt module selects predefined instruction templates for dynamic adjustment based on the current teaching objectives. The template generation process is completed by combining the rule engine with the neural network generator. The rule engine ensures that the generated content is logically coherent and meets educational standards, and the neural network is responsible for refining natural language expressions.

In the generation phase, the dynamic prompt template is spliced with the student's input questions and context information to form a complete input sequence. A hierarchical tagging strategy is used in sequence encoding, and special separators are used to distinguish different modules to assist the model in accurately interpreting each part of the information. To ensure real-time performance, the system designs a lightweight cache mechanism to incrementally encode frequently updated feature vectors, reduce repeated calculation delays, and meet the needs of complex teaching scenarios. The effect of the hierarchical tagging strategy is shown in Table 2.

Table 2. Effect of hierarchical tagging strategy.

Sample No.	Cache Strategy	Input Processing Time	Repeated Calculation Time	Total Delay Time	Delay Time After Optimization	Delay Reduction Ratio
1	Enable cache mechanism	120 ms	30 ms	150 ms	90 ms	40%
2	Disable cache mechanism	150 ms	50 ms	200 ms	200 ms	0%
3	Enable incremental encoding	100 ms	20 ms	120 ms	80 ms	33%
4	Enable cache + incremental encoding	110 ms	25 ms	135 ms	75 ms	44%

Table 2 shows the changes in latency before and after the optimization strategy, reflecting the effects of the cache mechanism and incremental encoding. The cache mechanism prevents the repeated calculation of frequently occurring identical tasks, significantly reducing the repeated calculation time and thus reducing the total latency. By batch processing multiple tasks, throughput can be significantly improved, especially when the system is heavily loaded. Batch processing can reduce waiting time and improve overall response

speed. The hierarchical labeling strategy refers to using specific delimiters to mark the boundaries of different information modules in the input sequence when constructing the complete input sequence of dynamic prompts. This strategy operates in the sequence encoding stage, aiming to clearly distinguish modules such as learner state descriptions, knowledge point summaries, and task instruction prompts in the dynamic prompt template, as well as the student's input questions and contextual information. Table 2 presents the quantitative effects of this strategy combined with caching mechanisms and incremental encoding on input processing time and overall latency. The importance of this strategy lies in its ability to ensure the model's structured understanding of complex, multi-part inputs, providing a technical prerequisite for the dynamic prompt mechanism to accurately control the generation process. Its role in the research evidence process is to support the feasibility demonstration of the proposed dynamic prompt construction mechanism in real-time teaching scenarios through empirical data that reduces latency, establishing a link between the method design and its expected efficiency advantages through observable indicators.

In the model reasoning stage, based on the Transformer architecture fine-tuned by LoRA mentioned above, combined with instruction fine-tuning parameters, dynamic prompts are used as guides to generate multiple candidate texts through the Beam Search strategy, and the results with the highest semantic consistency and personalized matching are selected. The matching degree calculation uses weighted cosine similarity, and the weights are dynamically adjusted according to the teaching focus and student needs to ensure that the generated content not only meets the task requirements, but also fits the learner's cognitive level and interest preferences.

In addition, the system regularly collects feedback on the generated output and adjusts the dynamic prompt template design and behavior encoder parameters in combination with the online learning mechanism to achieve continuous iterative optimization. The feedback data participates in model fine-tuning through the semi-supervised learning framework to improve the model's sensitivity to personalized differences. This process combines gradient accumulation technology to balance training efficiency and data diversity.

3. Adaptive Generation Performance Evaluation

The educational corpus constructed in the experiment ultimately contains 152,347 text samples, covering three subjects, mathematics, physics, and English, and two educational levels: junior high and senior high. The sample type distribution is as follows: textbook explanations account for 41%, exercise stems for 29%, student answers for 18%, and teacher comments for 12%. All texts are anonymized to remove personally identifiable information.

Data preprocessing follows explicit rules. In the deduplication stage, TF-IDF combined with BERT embedding is used to calculate cosine similarity, with a threshold of 0.92. Texts with semantically repeated meanings exceeding this threshold are retained only for the most complete expression. In the cleaning stage, perplexity-based filtering is applied to remove low-quality text fragments with perplexity values greater than 120. The text normalization rule base defines 187 transformation rules to standardize common subject-specific abbreviations, correct typical spelling errors, and standardize the LaTeX expression format of mathematical formulas.

The key parameters for model training and inference are fixed as follows. The random seed for all training and evaluation procedures was fixed to 42 to ensure reproducibility. The base model is LLaMA-2-13B, using FP16 precision. In LoRA fine-tuning, the rank r is set to 8; the scaling factor α is 32; only the query and value projection matrices of all Transformer layers are injected. The optimizer is AdamW with a weight decay rate of 0.01. Training uses a linear warm-up and cosine decay learning rate scheduler with a maximum learning rate of 3×10^{-4} and 500 warm-up steps. The batch size is set to 64; the maximum

sequence length is set to 1024 words; the gradient clipping threshold is 1.0. The learning rate in the instruction fine-tuning phase is set to 1×10^{-4} , and training is performed for 3 epochs. The behavior encoder in the dynamic cueing mechanism is a 3-layer Transformer with a hidden layer dimension of 768.

The core training is as follows: after model initialization, pre-trained weights are loaded, and all parameters are frozen. Subsequently, LoRA modules are inserted in parallel next to the specified linear layers. During forward propagation, the original weights are added to the output of the low-rank increment matrix. The loss function calculates the cross-entropy between the model output and the target text. Backpropagation only updates the LoRA parameters and the task embedding vector. The KL divergence of the frozen layer activations is calculated every 500 steps to monitor training stability. After training, removing all adapter parameters except the original model weights yields a lightweight model for inference. The code framework, data processing scripts, and model configuration files used in this study are available in an open-source repository. This implementation is based on the PyTorch 2.6 and Hugging Face Transformers libraries. The codebase includes scripts for web crawling, TF-IDF and BERT-based deduplication, text normalization, and building structured instruction–output pairs; it also includes configuration files for the LLaMA-2-13B base model. The repository also includes configuration files specifying all hyperparameters (batch size 64, maximum sequence length 1024, LoRA alpha 32).

To comprehensively verify the effectiveness of the proposed method, performance evaluation was conducted from multiple dimensions. The generated content quality assessment examined the degree to which the model output aligns with educational objectives in terms of language and semantics. The personalization matching assessment measured the suitability of the generated content to the individual characteristics of learners. The multi-task adaptability assessment examined the model’s generalization ability in different teaching scenarios. The response time and resource utilization assessments addressed the feasibility of the method in practical deployment.

3.1. Generated Content Quality

This paper evaluates the impact of the LoRA rank value on the quality of content generated by the instruction fine-tuning model. BLEU (Bilingual Evaluation Understudy) and ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation–Longest Common Subsequence) are used as evaluation indicators to observe their changes within 20 training rounds. Experimental group 1: LoRA rank $r = 4$, with fewer inserted parameters; Experimental group 2: LoRA rank $r = 8$, with stronger expressive ability. Data source: A dataset of “instructions, inputs, and outputs” triplets constructed based on educational tasks, totaling about 124,000 items. Task types include multiple teaching tasks such as exercise generation, concept explanation, and wrong question analysis. Token length: The maximum supported input length is 1024 tokens, with dynamic window adaptation. After each round of training, the BLEU and ROUGE-L scores of the current model are recorded. The performance of the two groups of models is compared. Figure 3 was created using the Matplotlib library in the Python 3.14 programming language. Its generation process begins with recording the evaluation metrics after each epoch during model training. BLEU and ROUGE-L scores are calculated on a reserved validation set. These scores, indexed by training epoch, are stored as a structured data table. The plotting script reads this table, using the training epoch as the x-axis and the score as the y-axis.

Figure 3 shows the changes in model generation quality (BLEU and ROUGE-L scores) after 20 training epochs under different LoRA rank values ($r = 4$ and $r = 8$). The BLEU score measures the degree of overlap between the generated text and the target answer at the n-gram level, reflecting the language fluency and accuracy. The ROUGE-L score measures

the degree of match between the generated text and the reference answer on the longest common subsequence, and focuses more on the text structure and coverage completeness.

$r = 4$ (low-rank configuration): BLEU and ROUGE-L scores rise slowly, eventually tending to 0.85 and 0.89. The curves are stable but converge slowly, reflecting that the model can still learn task instructions with fewer parameter updates, but the accuracy is slightly lower.

$r = 8$ (high rank configuration): Around the 10th epoch, BLEU reaches 0.87, and ROUGE-L reaches 0.88 and then basically stabilizes, higher than 0.9, indicating that higher rank values provide stronger expressiveness and task adaptability. The faster rise in the early stage indicates that training converges more quickly. The configuration of $r = 8$ is better than $r = 4$ in all indicators, indicating that if resources permit, appropriately increasing the LoRA rank can effectively enhance the model's fine-tuning effect. Both configurations have reached saturation in the later stages of training (epoch 15~20), indicating that too many training rounds have little marginal benefit on performance improvement.

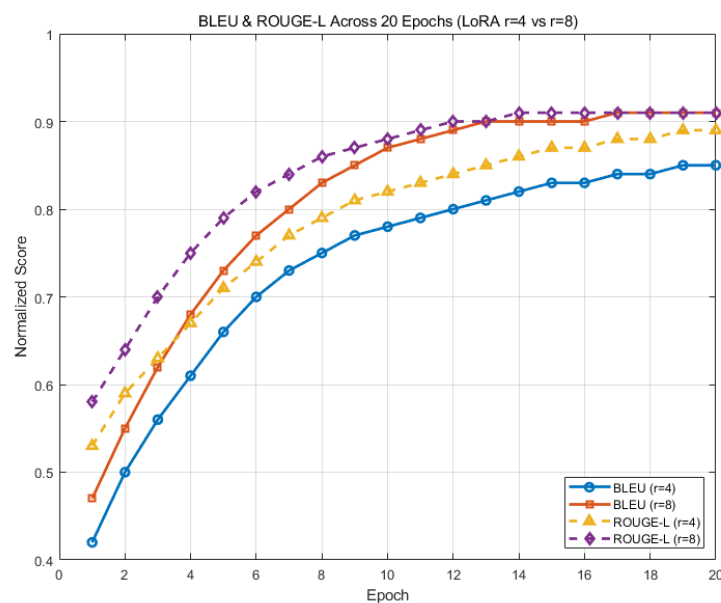


Figure 3. Generated content quality.

For BLEU scores, the mean of the $r = 4$ configuration is 0.846, with a standard deviation of 0.021; the mean of the $r = 8$ configuration is 0.912, with a standard deviation of 0.015. Paired t-tests show a statistically significant difference between the two configurations in the BLEU index ($t(19) = 9.874$, $p < 0.001$, Cohen's $d = 2.21$). For ROUGE-L scores, the mean of the $r = 4$ configuration is 0.887, with a standard deviation of 0.018; the mean of the $r = 8$ configuration is 0.913, with a standard deviation of 0.012. Paired t-tests also report a statistically significant difference ($t(19) = 7.632$, $p < 0.001$, Cohen's $d = 1.71$).

3.2. Matching Degree

Model configuration settings: Configuration A: LoRA rank is set to 4, and dynamic prompt is not introduced; Configuration B: LoRA rank is set to 8, and dynamic prompt construction mechanism is enabled. Both sets of models are fine-tuned based on the same pre-trained model (LLaMA-2-13B architecture) and the same educational corpus, and only differ in fine-tuning strategy and prompt mechanism to control variables. A sample of 80 students was randomly selected, covering different subjects (mathematics, physics, English) and different learning stages (elementary to intermediate). Four types of teaching content were generated for each student: exercise recommendation, concept explanation, Q&A feedback, and learning suggestions. A total of 640 pieces of content were

generated ($80 \times 4 \times 2$ configuration). This paper designed Likert 5-point scoring items in four dimensions and invited five senior teachers with teaching design and educational technology backgrounds as scorers. Each scorer was randomly assigned 160 pieces of content (each set of data was scored by two people to ensure consistency). The scoring stage was anonymous, and the scorers did not know which model configuration the content came from. Figure 4 was created using the Matplotlib and Seaborn libraries in the Python programming language. The generation process begins with the structured organization of the scoring data obtained from manual evaluations. After grouping each rater's scores across the four dimensions according to the model configuration, the mean and distribution statistics of the scores for each dimension are calculated. The plotting script reads the organized data frame and sets the scores as the vertical axis.

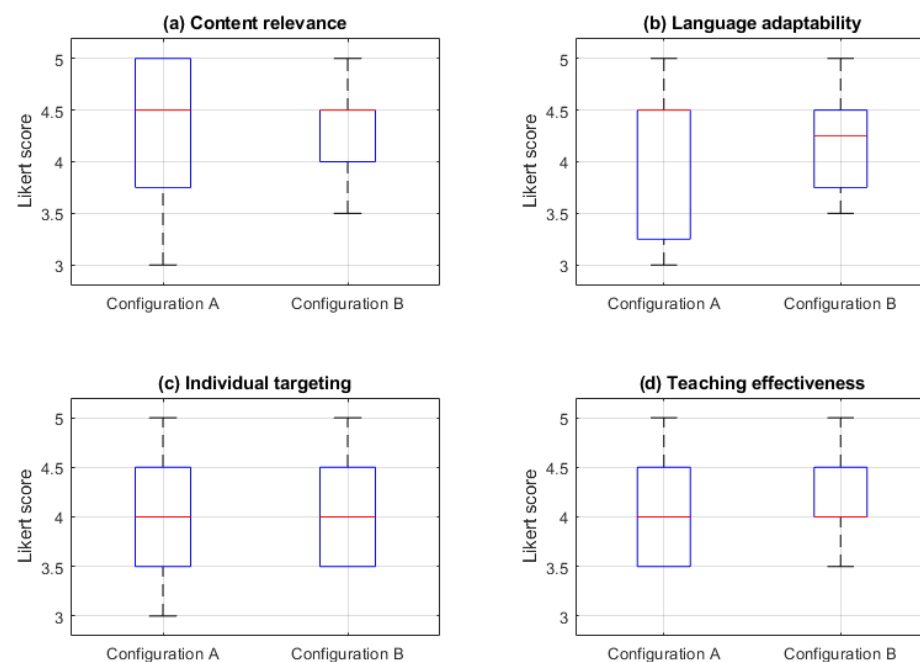


Figure 4. Generated content matching.

Figure 4 shows the distribution of personalized matching scores of generated content with different configurations (configurations A and B) on four key evaluation dimensions. Sub-figure a shows whether the generated content fits the students' current knowledge needs, and sub-figure b evaluates whether the language expression conforms to the students' cognitive style. Sub-graph c examines whether the generated content reflects the adaptation to the individual situation and ability of students, and sub-graph d determines whether the content has teaching value to promote learning. Configuration B shows stable scores in all dimensions, which means that its generated content is more stable in personalized matching. Although configuration A has acceptable scores in some dimensions, the distribution dispersion is large, indicating that its consistency and adaptability are not as good as configuration B.

To assess inter-rater consistency, this study calculated the intra-group correlation coefficient. The results showed an overall ICC value of 0.84, indicating high inter-rater consistency and supporting the effectiveness of subsequent personalized matching assessments. Table 3 shows the ICC values for each assessment dimension.

Table 3. Inter-rater reliability (ICC) across evaluation dimensions.

Evaluation Dimension	ICC Value	95% Confidence Interval
Knowledge Need Fit	0.85	0.81–0.88
Cognitive Style Match	0.82	0.78–0.85
Individual Adaptation	0.83	0.79–0.86
Pedagogical Value	0.86	0.83–0.89
Overall	0.84	0.81–0.87

3.3. Multi-Task Adaptability

Six representative educational generation tasks were selected, and the teaching scenarios included exercise generation, concept explanation, Q&A, error analysis, teaching suggestion generation, and knowledge point summary. 200 structured test samples were selected for each task type, all from a standard teaching corpus or simulated learning data. The sample content ensures coverage of different subject areas and knowledge depth to improve the representativeness of the evaluation. Using the LoRA fine-tuned model, teaching content was generated for each type of task. The average score and standard deviation of each task were calculated to evaluate the performance fluctuation of the model between tasks. Figure 5 was created using the Matplotlib library in the Python programming language. Its generation process begins with the summation and calculation of assessment scores for six categories of instructional tasks. The mean score and standard deviation for each task on the test set were calculated and stored. The plotting script reads this data, using the names of the six task categories as the x-axis and the mean score as the height of the main bars on the y-axis.

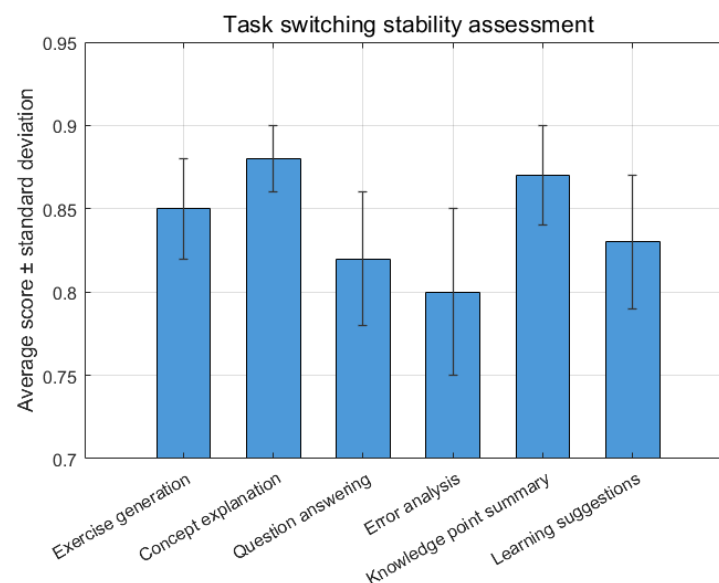
**Figure 5.** Multi-task adaptability.

Figure 5 shows the average scores of the model in all six teaching tasks, which are between 0.80 and 0.88, and the generation quality is good. The highest scores are concept explanation (0.88) and knowledge point summary (0.87), indicating that the model performs best in explanatory tasks. The relatively low scores are error analysis (0.80) and question answering (0.82), which may be because the tasks require stronger semantic understanding and contextual reasoning. The standard deviations of “Concept Explanation” and “Knowledge Point Summary” are relatively small (0.02~0.03). The model’s response to these two types of tasks is stable, with small cross-sample fluctuations and good generalization effects.

The standard deviation of “Error Analysis” is the largest (0.05), indicating that its score fluctuates greatly; the model may adapt unevenly to different input situations; its stability is relatively poor. The “Question Answering” task also shows a certain fluctuation (0.04). This type of task usually contains ambiguous questions or requires contextual memory, which is more challenging. By analyzing the standard deviation range of all tasks (0.02–0.05), it can be seen that the model has no extremely unstable performance in multi-task switching, and the score fluctuation between tasks is within an acceptable range, showing good cross-task adaptability.

3.4. Response Time and Resource Utilization

This paper evaluated the response time and resource utilization of the model under different layer number and batch size configurations. Experimental variables: Four different layer number configurations were selected, each representing a different scale of a Transformer model. Batch size: Four different batch sizes (32, 64, 128, 256) were selected to simulate the impact of different input data volumes on response time and resource utilization. Measurement indicators: Response time measures the time (in seconds) required for the model to output results from receiving input data. Resource utilization indicates the computing resources consumed by the model at runtime, which refers to the proportion of GPU computing resources occupied. Experiments were conducted using Transformer models with different numbers of layers. During each training or inference, one can adjust the number of layers to observe its impact on performance. For each layer configuration, different batch sizes can be used to experiment and simulate different training or inference scenarios. This paper records the response time of each experiment, that is, the time it takes for the model to process one input datum. The resource usage of each experiment can be recorded, that is, the proportion of GPU computing resources used by the model. Figure 6 was created using the Matplotlib library in the Python programming language. The generation process began with the extraction of time-series and resource monitoring data from experimental records under different combinations of layer numbers and batch sizes. Response time and GPU resource utilization percentage were organized into structured data as two sets of observation sequences.

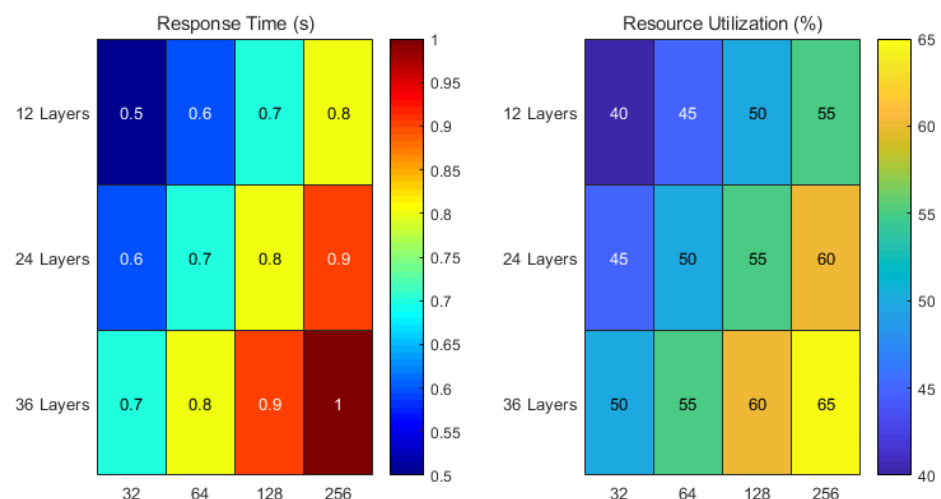


Figure 6. Response time and resource utilization.

Figure 6 shows the changes in response time and resource utilization under different model layer configurations and different batch sizes. The 12-layer model has the shortest response time, especially when the batch size is small. As the number of layers increases, the response time gradually increases. As the number of model layers increases, resource

usage also increases. With each additional layer, the model requires more computing power and video memory. The resource usage of the 12-layer model is low. When the batch size is large, resource demand reaches a peak, which may cause video memory overflow or long computational delays.

3.5. Throughput

Three different depth Transformer model configurations are selected, namely 12 layers, 24 layers, and 36 layers, all based on the same pre-trained architecture, and the model weights and other hyperparameters are kept consistent; only the number of layers is changed. This paper builds an inference test platform under a unified hardware environment to ensure that the test environment is stable and consistent and to eliminate external factors that interfere with performance. The batch size is set to 32, 64, 128, and 256, representing parallel inference requests of different scales, covering small batches for fast response and large batches for high throughput. For each model configuration, multiple rounds of inference tests are performed at each batch size. Each round of testing runs a fixed number of sample inputs and records the time taken for inference. The number of samples processed per second is calculated by “total number of samples/total inference time” as the throughput indicator. Each set of test data is repeated multiple times. The average throughput is calculated, and abnormal data is eliminated to ensure the reliability and stability of the results. Figure 7 was created using the Matplotlib library in the Python programming language. Its generation process begins by processing the raw time-series data from multiple rounds of inference tests under a unified hardware environment. The throughput of each configuration at each batch size was calculated using the formula “total number of samples processed/total inference time”, and the average value was obtained. The plotting script reads the processed data frame, uses batch size as the x-axis and throughput as the y-axis, and plots polylines with different marker styles for the three-layer configurations.

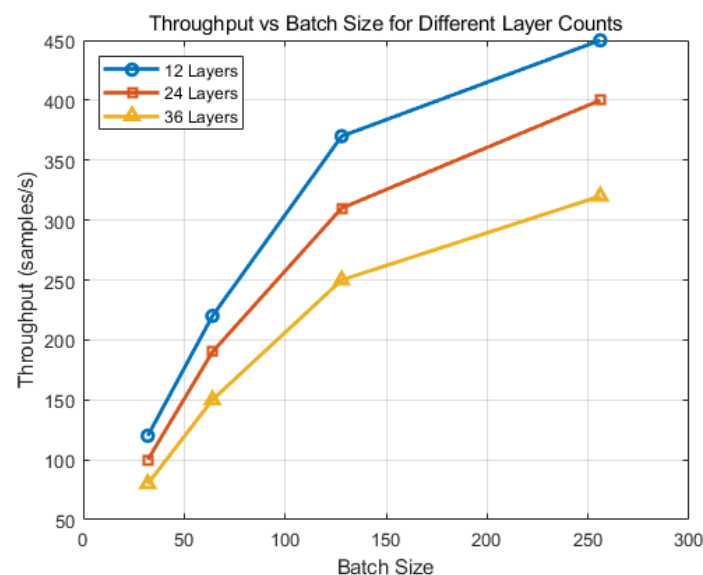


Figure 7. Throughput.

Figure 7 shows the model throughput performance of different Transformer layer configurations (12 layers, 24 layers, 36 layers) under different batch sizes (32, 64, 128, 256). For all layer configurations, the throughput increases with the increase in batch size. This shows that increasing the batch size can improve the parallel processing capability and resource utilization of the model, thereby increasing the number of samples processed per

second. The throughput decreases as the number of layers increases. The 12-layer model has the highest throughput at the same batch size, and the 36-layer model has the lowest. This is because the more layers there are, the higher the model's computational complexity and memory access overhead, resulting in a decrease in inference speed. As the batch size increases, the throughput improvement gradually slows down, especially when the batch size is 256; the growth of the three models tends to be flat, indicating a certain computing resource bottleneck or hardware limitation.

3.6. Validation of the Dynamic Prompting Mechanism for Modeling Students' Knowledge States

To verify the effectiveness of the dynamic prompting mechanism in modeling students' knowledge states, this study designed a longitudinal experiment. Sixty students with continuous learning records were selected as participants, covering mathematics and physics. The experiment collected each student's learning behavior sequence over a one-month period, including correctness of answers, response time, knowledge point access paths, and number of repeated learning sessions. This behavioral data was temporally encoded and input into the behavior encoder of the dynamic prompting mechanism to extract a high-dimensional latent state vector representing the students' knowledge states. To evaluate the vector's ability to represent students' actual knowledge levels, the experiment simultaneously used standard knowledge diagnostic tests for pre- and post-tests to obtain the actual trajectory of changes in their knowledge point mastery. By calculating the correlation between the latent state vector and the measured knowledge mastery, the study analyzed whether the dynamic prompting mechanism could accurately capture students' learning progress.

Table 4 shows that the latent state vector output by the behavior encoder is significantly positively correlated with students' measured knowledge point mastery in most dimensions, with an average correlation coefficient of 0.76. The results confirm that the dynamic prompting mechanism not only enhances the surface-level prompts, but its internal behavioral encoder can extract representations that are highly correlated with the real knowledge state from the learning behavior sequence, thereby achieving continuous modeling and tracking of the learner's knowledge state.

Table 4. Correlation Analysis between Behavioral Latent State Vectors and Measured Knowledge Mastery.

Subject	Number of Knowledge Points Involved in the Analysis	Average Correlation Coefficient (r)	The Proportion of Dimensions That Are Significantly Relevant
Mathematics	12	0.78	91.70%
Physics	9	0.73	88.90%
Overall	21	0.76	90.50%

3.7. Ablation Experiment Compared with Baseline

To verify the independent contributions of each optimization component and establish a rigorous performance baseline, this section designed ablation experiments and multi-baseline comparisons. Five model configurations were set up: the first group was the original LLaMA-2-13B base model without any domain adaptation training; the second group was a model fine-tuned using only LoRA technology; the third group was a model combining LoRA and instruction fine-tuning; the fourth group was the complete model proposed in this paper, integrating LoRA, instruction fine-tuning, and dynamic prompting mechanisms; the fifth group was the LLaMA-7B model. All models were evaluated on the same test set, which contained 2000 samples of structured teaching tasks from mathematics, physics, and English subjects.

Table 5 shows the performance comparison of the five models on the BLEU and ROUGE-L metrics. The base model scored the lowest, with BLEU at 0.73 and ROUGE-L at 0.77, indicating that the unadapted general model is insufficient to meet the accuracy requirements of educational domain generation. Fine-tuning using only LoRA improved both metrics to 0.81 and 0.84, confirming the effectiveness of low-rank adaptation for transferring educational language styles. After fine-tuning with instructions, the model's performance further improved to 0.86 and 0.88, indicating that structured task guidance can enhance the model's understanding of the teaching objectives. LLaMA-7B, after fine-tuning using the same process, scored 0.85 and 0.87, with the complete model achieving optimal performance. BLEU and ROUGE-L reached 0.91 and 0.92, respectively, demonstrating that the dynamic prompting mechanism, by introducing learner context, ultimately achieves gains in personalized matching and cognitive fit of the generated content.

Table 5. Ablation experiments and baseline model performance.

Model Configuration	BLEU	ROUGE-L
LLaMA-2-13B (Basic model)	0.73	0.77
+LoRA fine-tuning	0.81	0.84
+LoRA and instruction fine-tuning	0.86	0.88
LLaMA-7B (Same process, minor adjustments)	0.85	0.87
Complete model (Our method)	0.91	0.92

3.8. Learning Effectiveness Verification

To evaluate the effectiveness of this system in a real-world teaching environment, a control group experiment was conducted. Sixty-eight students from the first year of high school were recruited and randomly divided into an experimental group and a control group, with 34 students in each group. There was no statistically significant difference in pre-test scores between the two groups. The experimental group used an auxiliary learning system integrating the optimized model presented in this paper, which provided personalized exercises, explanations, and feedback based on their learning behavior. The control group used traditional numerical practice materials consisting of the same knowledge points but without personalized arrangement. The experiment lasted four weeks, focusing on the chapter on functions in high school mathematics. Standardized tests on this chapter were administered to all students before and after the experiment.

Test score analysis showed that the experimental group's post-test mean score was 82.4, with a standard deviation of 6.7; the control group's post-test mean score was 76.1, with a standard deviation of 7.9 (Table 6). The adjusted post-test mean estimate for the experimental group was 81.9, and for the control group, it was 76.6. These results demonstrate that the personalized learning path using the optimized model presented in this paper produces a more significant learning gain compared to static materials, providing preliminary evidence for the system's educational effectiveness. To test the significance of the difference in post-test scores between groups, pre-test scores were used as a covariate in an analysis of post-test scores. The results showed that, after controlling for pre-test scores, there was a statistically significant difference in post-test scores between the experimental and control groups ($F(1, 65) = 9.84, p = 0.003$). The partial effect size η^2 was 0.13, indicating that the personalized learning system intervention had moderate practical significance.

Table 6. Analysis of pre- and post-test results in the control group.

Groups	Number of People	Pre-Test Mean Score (Standard Deviation)	Post-Test Mean Score (Standard Deviation)	Adjusted Posttest Mean Estimate
Experimental group	34	75.2 (7.3)	82.4 (6.7)	81.9
Control group	34	74.8 (7.5)	76.1 (7.9)	76.6

4. Conclusions

Based on the Transformer architecture, this paper proposes a generative AI optimization method for adaptive learning by combining low-rank adaptation technology and structured instruction fine-tuning. By constructing multi-dimensional educational data and a dynamic prompt mechanism, the model can accurately perceive students’ individual behaviors and knowledge mastery and generate personalized content. Experimental results show that this method performs well in improving the accuracy of adaptive question-answering and personalized matching, while ensuring the timeliness of response, verifying its practical value in intelligent education. Although this study has made significant progress in model adaptability and generation quality, there are still deficiencies in complex multimodal information fusion and long-term learning trajectory modeling.

The overall added value of this work lies in providing a systematic and generalizable optimization path and application scheme for the deep integration of generative AI and adaptive learning. This methodology transcends simple algorithm optimization, involving computational modeling of learning behavior and knowledge transfer processes, demonstrating the interdisciplinary methodological value of intelligent technology in empowering educational reform. The main challenges in the research implementation process were the construction and annotation of high-quality, structured educational corpora and the collaborative optimization of real-time behavior encoding and low-latency inference in the dynamic prompting mechanism. The scarcity and heterogeneity of domain data increased the complexity of model adaptation. Future research should explore the adaptation of teaching scenarios incorporating multimodal inputs, develop more interpretable long-term learning trajectory modeling methods, and verify the system’s sustained impact on deep educational indicators such as knowledge retention and cognitive load through longer-term longitudinal studies.

Author Contributions: X.L. and Z.L.; Writing, Editing, Software, Review and Editing. Z.L.; Data analysis, X.L. Resources. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the 2024 Henan Higher Education Teaching Reform Research and Practice Project (No. 2024SJGLX0383).

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and the protocol was approved by the Ethics Committee of School of Media and Law, NingboTech University (Project No. 20250721) on 10 July 2024.

Informed Consent Statement: Verbal informed consent was obtained from the participants. Verbal consent was obtained rather than written because this study is a longitudinal experiment that selected 60 sequences of learning behaviors with continuous learning records over a one-month period, without the need to directly collect data from students.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy concerns and the lack of research permission.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Li, H.; Xu, T.; Zhang, C.; Chen, E.; Liang, J.; Fan, X.; Li, H.; Tang, J.; Wen, Q. Bringing generative AI to adaptive learning in education. *arXiv* **2024**, arXiv:2402.14601. [CrossRef]
- Guettala, M.; Bourekache, S.; Kazar, O.; Harous, S. Generative artificial intelligence in education: Advancing adaptive and personalized learning. *Acta Inform. Pragensia* **2024**, *13*, 460–489. [CrossRef]
- Kwak, M.; Jenkins, J.; Kim, J. Adaptive programming language learning system based on generative AI. *Issues Inf. Syst.* **2023**, *24*, 222–231.
- Hess, B.J.; Cupido, N.; Ross, S.; Kvern, B. Becoming adaptive experts in an era of rapid advances in generative artificial intelligence. *Med. Teach.* **2024**, *46*, 300–303. [CrossRef]
- Maity, S.; Deroy, A. Generative ai and its impact on personalized intelligent tutoring systems. *arXiv* **2024**, arXiv:2410.10650. [CrossRef]
- Sachete, A.S.; de Sant'anna de Freitas Loiola, A.V.; Gomes, R.S. AdaptiveGPT: Towards Intelligent Adaptive Learning. *Multimed. Tools Appl.* **2024**, *83*, 89461–89477. [CrossRef]
- Reunanen, T.; Nieminen, N. Artificial Intelligence as a Catalyst: A Case Study on Adaptive Learning in Programming Education. *Hum. Factors Bus. Manag. Soc.* **2024**, *135*, 303–312.
- Pesovski, I.; Santos, R.; Henriques, R.; Trajkovic, V. Generative AI for customizable learning experiences. *Sustainability* **2024**, *16*, 3034. [CrossRef]
- Abbes, F.; Bennani, S.; Maalel, A. Generative AI and gamification for personalized learning: Literature review and future challenges. *SN Comput. Sci.* **2024**, *5*, 1154. [CrossRef]
- Tu, Y.; Chen, J.; Huang, C. Empowering personalized learning with generative artificial intelligence: Mechanisms, challenges and pathways. *Front. Digit. Educ.* **2025**, *2*, 19. [CrossRef]
- Hegade, P.; Shettar, A. Designing Portfolios for Adaptive Learning. In *Adopting Artificial Intelligence Tools in Higher Education: Teaching and Learning*; CRC Press: Boca Raton, FL, USA, 2025; p. 100.
- Yu, H. Reflection on whether Chat GPT should be banned by academia from the perspective of education and teaching. *Front. Psychol.* **2023**, *14*, 1181712. [CrossRef]
- Ni, X. Building an interactive learning space driven by generative artificial intelligence: Personalized English learning experience. *Sci. Innov. Asia* **2024**, *2*, 31.
- Khalil, M.; Wong, J.; Wasson, B.; Paas, F. Adaptive support for self-regulated learning in digital learning environments. *Br. J. Educ. Technol.* **2024**, *55*, 1281–1289. [CrossRef]
- Dong, W.; Pan, D.; Kim, S. Exploring the integration of IoT and Generative AI in English language education: Smart tools for personalized learning experiences. *J. Comput. Sci.* **2024**, *82*, 102397. [CrossRef]
- SH, U.A.; Supriyono, N.M.R.P. Optimizing Learning Through Artificial Intelligence: Evaluating the Impact of Adaptive Learning Technologies on Student Outcomes. *Sinergi Int. J. Educ.* **2023**, *1*, 138–149. [CrossRef]
- ElSayary, A. Integrating generative AI in active learning environments: Enhancing metacognition and technological skills. *J. Syst. Cybern. Inform.* **2024**, *22*, 34–37.
- Kuo, M.; Sarker, S.; Qian, L.; Fu, Y.; Li, X.; Dong, X. Enhancing deep knowledge tracing via diffusion models for personalized adaptive learning. *arXiv* **2024**, arXiv:2405.05134.
- Choi, D.; Im, J.; Sung, Y. LoRA Fusion: Enhancing Image Generation. *Mathematics* **2024**, *12*, 3474. [CrossRef]
- Cosentino, G.; Anton, J.; Sharma, K.; Gelsomini, M.; Giannakos, M.; Abrahamson, D. Generative AI and multimodal data for educational feedback: Insights from embodied math learning. *Br. J. Educ. Technol.* **2025**, *56*, 1686–1709. [CrossRef]
- Liu, J. Generative Artificial Intelligence Drives the Transformation of Learning: Trends and Challenges. *Artif. Intell. Technol. Res.* **2024**, *2*.
- Leon, M. Leveraging Generative AI for On-Demand Tutoring as a New Paradigm in Education. *Int. J. Cybern. Inform. (IJCI)* **2024**, *14*, 17. [CrossRef]
- Asimiyu, Z. Explainable Generative AI in Financial and Real Estate Applications: Achieving High-Fidelity Results Through Adaptive Learning. 2024. Available online: https://www.researchgate.net/profile/Zainab-Asimiyu/publication/386100693_Explainable_Generative_AI_in_Financial_and_Real_Estate_Applications_Achieving_High-Fidelity_Results_Through_Adaptive_Learning/links/67447cbab5bd9d17d606dd5a/Explainable-Generative-AI-in-Financial-and-Real-Estate-Applications-Achieving-High-Fidelity-Results-Through-Adaptive-Learning.pdf (accessed on 2 November 2025).
- Chandrashekar, K.; Jangampet, V.D. Enhancing generative AI precision: Adaptive prompt reinforcement learning for high-fidelity applications. *Int. J. Comput. Eng. Technol. (IJCET)* **2021**, *12*, 81–90.
- Mao, J.; Chen, B.; Liu, J.C. Generative artificial intelligence in education and its implications for assessment. *TechTrends* **2024**, *68*, 58–66. [CrossRef]

26. Nzenwata, U.J.; Barn-Nzekwe, C.L.; Ojelabi, E.O.; Oduware, O.; Atalor, P.E.; Yisau, Y.; Adeyela, A.T.; Nwanguma, E.C.; Emokiniovo, E.; Osisanya, O.A. A systematic review of generative AI in education. *J. Comput. Sci. Appl.* **2024**, *12*, 25–30. [[CrossRef](#)]
27. Yang, H. Harnessing generative AI: Exploring its impact on cognitive engagement, emotional engagement, learning retention, reward sensitivity, and motivation through reinforcement theory. *Learn. Motiv.* **2025**, *90*, 102136. [[CrossRef](#)]
28. Banjade, S.; Patel, H.; Pokhrel, S. Empowering Education by Developing and Evaluating Generative AI-Powered Tutoring System for Enhanced Student Learning. *J. Artif. Intell. Capsul. Netw.* **2024**, *6*, 278–298. [[CrossRef](#)]
29. Veluru, C.S. The Impact of Generative AI on Content Curation and Content Advancements in Education and Training. *Eur. J. Adv. Eng. Technol.* **2024**, *11*, 121–130.
30. Wu, X.Y.; Chiu, T.K.F. Integrating learner characteristics and generative AI affordances to enhance self-regulated learning: A configurational analysis. *J. New Approaches Educ. Res.* **2025**, *14*, 10. [[CrossRef](#)]
31. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *1*, 30–41.
32. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *ICLR* **2022**, *1*, 3–23.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.