



ARTICLE

# An Intelligent Lesson Preparation Assistant System Using a Fine-Tuned Large Language Model for Computer Science Education

Hongzhi Li<sup>1,\*</sup>, Jiale Wu<sup>1</sup>, Dun Li<sup>2,\*</sup>, Kezhong Lu<sup>1</sup> and Qishou Xia<sup>1</sup>

<sup>1</sup>School of Big Data and Artificial Intelligence, Chizhou University, Chizhou, China

<sup>2</sup>Department of Industrial Engineering, Tsinghua University, Beijing, China

\*Corresponding Authors: Hongzhi Li. Email: pickpickup@sohu.com; Dun Li. Email: lidun.2023@tsinghua.org.cn

Received: 03 June 2026; Accepted: 24 August 2026; Published: 22 October 2026

**ABSTRACT:** The application of large language models (LLMs) to computer science (CS) education has great potential for intelligent lesson preparation, yet existing systems remain limited by weak textbook alignment, inconsistent pedagogical structures, and extensive manual post-editing. This paper presents the Fine-Tuned DeepSeek-driven Intelligent Lesson Preparation Assistant System (FT-DILPAS), a textbook-driven lesson preparation framework that integrates efficient model adaptation, structured textbook understanding, curriculum-aware prompting, and retrieval-augmented generation into a unified pipeline. Evaluations on 60 CS textbooks show that FT-DILPAS achieves parsing accuracies of 96.5% for standard layouts and 88.2% for complex layouts, significantly outperforming traditional parsing methods. Human evaluations further demonstrate improvements of 50.8% in format standardization and 30.6% in textbook alignment over GPT-4o, with consistently superior pedagogical quality across all evaluated general-purpose LLMs. Overall, FT-DILPAS provides a practical and scalable paradigm for curriculum-aligned, AI-assisted lesson preparation in computer science education.

**KEYWORDS:** Intelligent lesson preparation; fine-tuned LLMs; retrieval-augmented generation (RAG); computer science education

## 1 Introduction

The rapid digital transformation of education has accelerated the modernization of computer science (CS) curricula, creating increasing demands for efficient and high-quality instructional design. As curriculum standards continue to evolve alongside emerging technologies such as artificial intelligence and cloud computing, conventional lesson preparation based on manual editing and static textbooks has become increasingly inefficient. Empirical studies indicate that instructors, particularly in primary, secondary, and vocational education, spend more than 60% of their working time on repetitive tasks including lesson planning and teaching material preparation, leaving limited opportunities for instructional innovation and personalized guidance [1–3].

Recent advances in large language models (LLMs), represented by GPT-4 and DeepSeek, have demonstrated remarkable capabilities in educational content generation and intelligent tutoring [4–6]. Nevertheless, directly adopting general-purpose LLMs for CS education remains problematic. Their tendency to generate hallucinated explanations or logically incorrect code compromises the rigor required in programming education [7,8]. More fundamentally, existing LLMs lack awareness of curriculum-specific pedagogical

structures, textbook organizations, and institutional teaching standards. Consequently, the generated materials often require extensive manual revision, substantially reducing the practical efficiency gains expected from AI-assisted lesson preparation [9]. Existing educational LLM applications are predominantly designed as open-ended conversational tutors, where prolonged multi-turn interactions introduce unpredictable computational costs and inconsistent pedagogical outputs. In contrast, lesson preparation is inherently a structured instructional design task rather than a dialogue-intensive activity. This mismatch motivates a task-driven generation paradigm in which instructors provide structured pedagogical requests (e.g., textbook, chapter, knowledge point, and instructional constraints), enabling deterministic, bounded-cost generation of complete teaching materials.

To address these challenges, we propose the Fine-Tuned DeepSeek-driven Intelligent Lesson Preparation Assistant System (FT-DILPAS) for computer science education. Built upon the open-source DeepSeek-LLM-7B and following a Textbook-Driven paradigm, FT-DILPAS treats official textbooks as the authoritative knowledge source and integrates QLoRA-based efficient fine-tuning, a hybrid textbook parsing algorithm (pdfplumber + TF-IDF + KMeans), Retrieval-Augmented Generation (RAG), and the CO-STAR prompting framework into a unified lesson preparation pipeline. By preserving the pedagogical structure of textbooks throughout the generation process, the proposed system enables automated generation of standardized multimodal teaching resources—including lesson plans, code examples, practical exercises, and presentation slides—while effectively reducing hallucinations, improving curriculum alignment, and maintaining predictable computational costs suitable for local deployment [10].

The main contributions of this work are summarized as follows:

1. We propose FT-DILPAS, a textbook-driven lesson preparation framework that integrates efficient LLM fine-tuning with a hybrid textbook parsing algorithm, enabling structured understanding of computer science textbooks and curriculum knowledge.
2. We develop a unified generation pipeline combining Retrieval-Augmented Generation (RAG) and the CO-STAR prompting framework to suppress hallucinations, enforce pedagogical constraints, and produce standardized, curriculum-aligned teaching materials.
3. We conduct comprehensive experiments on 60 computer science textbooks spanning three subdisciplines. FT-DILPAS achieves 96.5% textbook parsing accuracy and consistently outperforms generic LLMs in textbook alignment, structural completeness, and format standardization.
4. We demonstrate that FT-DILPAS substantially reduces lesson preparation effort while generating classroom-ready multimodal teaching resources, providing an efficient and cost-effective solution for AI-assisted computer science education.

The primary contribution of FT-DILPAS does not stem from introducing new individual techniques, but from systematically integrating QLoRA, RAG, and CO-STAR into a pedagogically coherent instructional generation framework. Specifically, the proposed framework (1) establishes an iterative feedback loop between hybrid document parsing and prompt engineering, enabling automatic prompt refinement based on parsing outcomes; (2) enhances retrieval quality through a curriculum-aware weighting strategy that prioritizes pedagogically important textbook content using TF-IDF-derived importance scores; and (3) ensures semantic consistency across multimodal instructional resources via a unified dispatch mechanism, thereby improving the coherence and quality of the generated lesson plans, exercises, and presentation slides.

## 2 Related Work

### 2.1 From Rule-Based Tutors to Generative AI

Intelligent tutoring systems (ITS) have evolved from rule-based expert systems to LLM-driven generative assistants, reflecting the rapid advancement of artificial intelligence in education [11]. Early systems, such as Cognitive Tutors, relied on manually constructed production rules to provide structured learning guidance but lacked the flexibility to support open-ended interaction and natural language understanding [12]. Recent advances in large language models (LLMs) have enabled ITS to generate personalized explanations, provide real-time programming assistance, and adapt instructional content to diverse learning needs [13–15]. Despite these advances, current LLM-based educational systems remain largely general-purpose and often struggle to maintain curriculum consistency, pedagogical structure, and textbook alignment, motivating the development of domain-specific instructional generation frameworks.

### 2.2 The Challenge of Domain Adaptation

Although general-purpose LLMs have demonstrated remarkable generative capabilities, their direct application to computer science (CS) education remains limited by hallucinations, unreliable code generation, and insufficient pedagogical consistency [16,17]. More fundamentally, these models lack explicit awareness of curriculum structures and textbook organizations, making it difficult to generate teaching materials that conform to institution-specific instructional standards [18]. Recent research indicates that domain-specific adaptation, particularly efficient fine-tuning combined with retrieval-enhanced knowledge grounding, consistently outperforms prompt engineering alone in improving instructional accuracy and curriculum alignment [19–22]. This trend motivates the development of specialized educational LLMs capable of generating reliable, textbook-aligned teaching resources.

### 2.3 Making LLMs Feasible for Education

The widespread adoption of LLMs in education is constrained by the high computational cost of conventional full fine-tuning, which requires substantial GPU memory and computing resources [23]. Recent advances in Parameter-Efficient Fine-Tuning (PEFT), particularly Quantized Low-Rank Adaptation (QLoRA), have substantially reduced these resource requirements by combining lightweight parameter updates with low-bit quantization [24,25]. As a result, domain-specific LLMs can be efficiently adapted and deployed on commodity hardware, making localized, privacy-preserving, and cost-effective educational applications increasingly feasible [26].

### 2.4 Breaking the Memory Barrier with Distributed Training

Although parameter-efficient fine-tuning substantially lowers training costs, memory constraints remain a major challenge when adapting large language models to educational tasks. Fully Sharded Data Parallelism (FSDP) improves scalability by distributing model parameters, gradients, and optimizer states across multiple GPUs, thereby overcoming single-device memory limitations [27,28]. Recent studies have shown that the combination of FSDP and QLoRA provides an effective balance between computational efficiency and model capacity, enabling cost-effective training of domain-specific LLMs on both workstation-class hardware and large-scale GPU clusters [29,30]. These advances make scalable educational LLM deployment increasingly practical.

## 2.5 Summary of Related Work

Table 1 summarizes representative studies related to this work, highlighting their contributions and the key limitations addressed by the proposed “Textbook-Driven” paradigm.

**Table 1:** Summary of related work.

| Category             | Representative Work  | Key Contributions                | Limitations                                |
|----------------------|----------------------|----------------------------------|--------------------------------------------|
| Generative ITS       | GPT-4, Claude [11]   | Human-like dialogue capabilities | Prone to hallucinations.                   |
| Domain Adaptation    | LoRA, QLoRA [4,5]    | Parameter-efficient adaptation   | Not validated for complex code generation. |
| Distributed Training | FSDP, ZeRO [6,17]    | Memory-efficient sharding        | Not combined with curriculum data.         |
| Intelligent Tutoring | Cognitive Tutor [10] | Structured step-by-step guidance | Cannot handle open-ended student queries.  |

## 3 Methodology

This section presents the methodology of the proposed FT-DILPAS framework for textbook-driven lesson preparation. The framework integrates four complementary components: (1) a resource-efficient fine-tuning pipeline based on QLoRA and sequence packing; (2) textbook-structure-aware knowledge extraction; (3) a CO-STAR-based controllable prompting framework; and (4) a retrieval-augmented generation (RAG) module for knowledge grounding. Together, these components enable curriculum-aligned, standardized, and multimodal teaching material generation. The overall three-layer architecture is illustrated in Fig. 1.

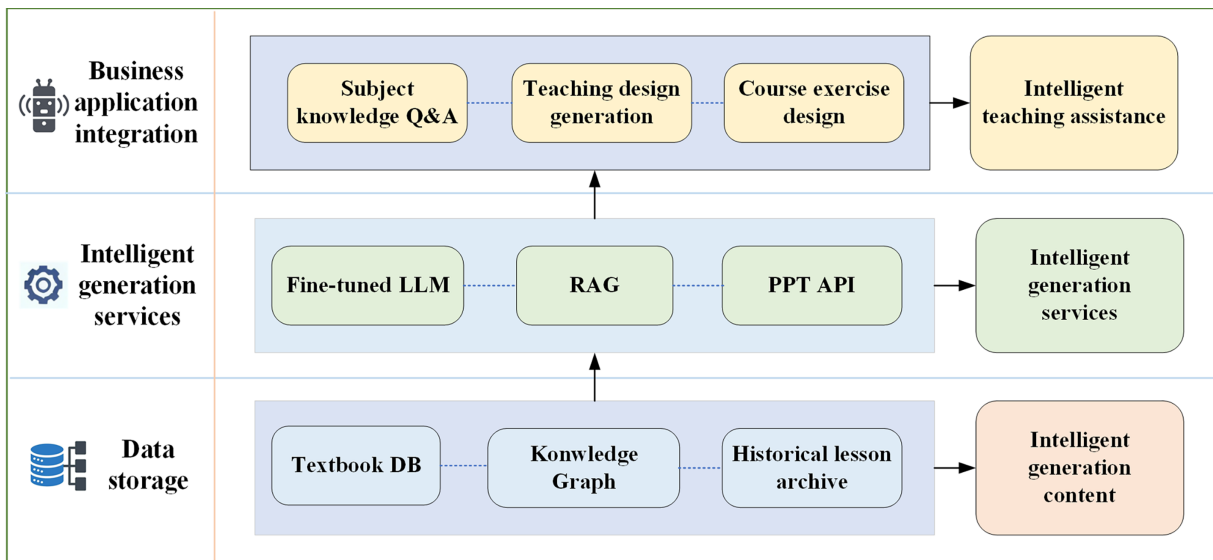


**Figure 1:** Three-layer system architecture.

Building upon the three-layer pipeline illustrated in Fig. 1, we further elaborate on the data flow and synergy among core modules. Standard RAG ranks chunks solely by cosine similarity, ignoring the textbook's hierarchical structure. We address this by incorporating the knowledge tree from our hybrid parser into retrieval scoring. For each chunk  $d_i$ , the retrieval score is

$$\text{Score}(d_i) = \alpha \cdot \text{sim}(q, d_i) + \beta \cdot \text{HierWeight}(d_i), \quad (1)$$

where  $\text{sim}(q, d_i) = \frac{e_q \cdot e_{d_i}}{\|e_q\| \|e_{d_i}\|}$  is the cosine similarity between the query embedding and the chunk embedding. The structural weight term  $\text{HierWeight}(d_i)$  is directly derived from the unified knowledge tree constructed by the hybrid parser (see Section 3.2), and is computed as:

$$\text{HierWeight}(d_i) = \gamma \cdot \text{Depth}(d_i) + (1 - \gamma) \cdot \text{TF-IDF}(d_i), \quad (2)$$

with  $\gamma \in [0, 1]$ . In all experiments, the hyperparameter  $\gamma$  is fixed at 0.6 to balance textbook hierarchy and term importance. Here,  $\text{Depth}(d_i)$  denotes the normalised hierarchical depth of the knowledge point associated with  $d_i$ , while  $\text{TF-IDF}(d_i)$  represents the normalised TF-IDF score of its core terminology computed over the textbook corpus.

The hyper-parameters  $\alpha$  and  $\beta$  control the trade-off between semantic relevance and structural fidelity. A larger  $\beta$  is preferred for well-structured textbooks with standardized headings and explicit chapter hierarchies, allowing retrieval to better preserve the intended pedagogical sequence. In contrast, a larger  $\alpha$  is adopted for noisy or irregular documents (e.g., scanned or multi-column PDFs), where semantic similarity compensates for structural parsing errors and improves retrieval robustness.

### 3.1 Fine-Tuning for Educational Domain Adaptation

Although general-purpose LLMs such as DeepSeek provide strong generative capabilities, their direct application to computer science education is constrained by limited domain adaptation and high computational overhead. We therefore employ a resource-efficient fine-tuning strategy that integrates QLoRA, sequence packing, and frozen backbone parameters for cost-effective domain adaptation.

#### 3.1.1 QLoRA for Parameter-Efficient Adaptation

Full fine-tuning a 4B-parameter model with FP16 precision and the Adam optimizer requires approximately 48 GB of VRAM, which exceeds the memory capacity of most GPUs commonly deployed in educational environments. Accordingly, we employ QLoRA [25] to quantize the base model to 4-bit NF4 (Normal Float 4-bit) and inject lightweight low-rank adapter layers to achieve efficient domain adaptation. Let  $W \in \mathbb{R}^{d \times k}$  be a pre-trained weight matrix. Following the standard LoRA formulation, the weight update is approximated as:

$$W_{\text{adapted}} = W_{\text{frozen}} + \Delta W, \quad \Delta W = s \cdot AB^T, \quad (3)$$

where  $A \in \mathbb{R}^{d \times r}$ ,  $B \in \mathbb{R}^{k \times r}$ ,  $\text{rank } r \ll \min(d, k)$ , and  $s \in \mathbb{Z}_q^*$  is a scaling factor. With  $AB^T \in \mathbb{R}^{d \times k}$ , the update is dimensionally consistent with the original weight matrix. As shown in Table 2, this reduces trainable parameters to 0.21% of the base model and total VRAM usage from 48 to 20 GB.

**Table 2:** Memory footprint comparison for DeepSeek-LLM-7B model.

| Component          | Full Fine-Tuning | QLoRA         |
|--------------------|------------------|---------------|
| Base Model Weights | 8 GB (FP16)      | 2 GB (NF4)    |
| Trainable Params   | 4 B (100%)       | 8.4 M (0.21%) |
| Optimizer States   | 32 GB            | 67 MB         |
| Total VRAM         | 48 GB            | 20 GB         |

### 3.1.2 Sequence Packing for Training Throughput

Educational QA datasets (e.g., textbook question–answer pairs) usually exhibit highly variable sequence lengths, ranging from 30 to 150 tokens. Direct padding to a fixed 2048-token context window leads to over 70% of computational waste caused by redundant padding tokens. To eliminate such inefficiency, we adopt sequence packing [7] to concatenate multiple samples into each training batch until the context limit  $L_{max}$  is reached. The packing efficiency is defined as:

$$Efficiency = \frac{\sum_{i=1}^M L_{real,i}}{M \cdot L_{max}}, \quad (4)$$

where  $M$  denotes the number of training samples packed into a batch,  $L_{real,i}$  is the actual token length of sample  $i$ , and  $L_{max} = 2048$  is the fixed context window size. The packing efficiency ranges from 0 to 1, with higher values indicating less padding overhead. The pipeline achieves a packing efficiency of 0.86 in all training experiments.

## 3.2 Textbook Parsing and Knowledge Point Structuring

The proposed system targets the challenge of converting unstructured educational materials into a machine-interpretable knowledge hierarchy. This section elaborates on the pipeline for parsing PDF-formatted textbooks and structuring them into a unified Textbook–Chapter–Knowledge Point tree. Such a structured representation acts as the foundational knowledge base, ensuring all generated teaching materials remain strictly aligned with official curriculum standards.

### 3.2.1 Text Extraction and Preprocessing

The workflow starts with high-fidelity text extraction using the *pdfplumber* library. Unlike conventional text extraction tools, our algorithm retains layout coordinates and font metrics to preserve structural semantics. We design a three-stage preprocessing pipeline as follows:

1. **Layout Analysis:** We analyze font sizes, indentation levels, and positional coordinates to distinguish between headers, body text, and footnotes.
2. **Noise Reduction:** Regular expressions are applied to remove pagination artifacts, copyright notices, and decorative characters (e.g., ellipses, horizontal rules) that often plague PDF-to-text conversion.
3. **Directory Reconstruction:** We reconstruct the table of contents by analyzing the hierarchical patterns in font sizes and indentation, creating an initial chapter-segmentation map.

### 3.2.2 Semantic Segmentation and Knowledge Point Clustering

Unlike conventional document parsers based solely on layout heuristics or end-to-end learning, the proposed pipeline jointly exploits layout information and semantic representations for textbook segmentation. The cluster number  $k$  is automatically determined from the extracted textbook hierarchy, enabling semantically coherent segmentation while preserving pedagogical structure. The complete workflow is summarized in Algorithm 1.

1. **Text Vectorization.** We employ Jieba to perform Chinese word segmentation, which is essential for processing computer science terminology in Chinese textbooks. Page-level text is then assembled into a complete corpus. A TF-IDF vectorizer converts the corpus into numerical representations, whereby key terms are emphasized and stop words are removed.
2. **K-Means Clustering.** The vectorized text segments are imported into the K-Means algorithm. Pages with close semantic similarity are assigned to the same cluster, enabling automatic detection of natural chapter and section boundaries. The cluster number  $k$  is determined by the initial directory structure obtained from layout analysis, so as to maintain pedagogical rationality.

The core objective of the clustering algorithm is to minimize the Sum of Squared Errors (SSE) within clusters, optimizing the semantic grouping. Mathematically, given a set of data points  $\mathbf{x}_i$  and cluster centroids  $\boldsymbol{\mu}_j$ , the objective function  $J$  is defined as:

$$J = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2, \quad (5)$$

where  $C_j$  represents the set of points in the  $j$ -th cluster. By minimizing  $J$ , the algorithm ensures that text segments within the same cluster (representing a knowledge unit) are maximally similar.

---

**Algorithm 1:** Semantic structure analysis

---

**Input:** *pdf\_path* is the file path of the input textbook.

**Output:** A structured knowledge tree object.

```

1 corpus = [];
2 pdfplumber.open(pdf_path) → pdf;
  // Extract text and perform segmentation.
3 for page ∈ pdf.pages do
4   page.extract_text() → text;
5   if text != NULL then
6     tokens = " ".join([w for w in jieba.cut(text) if len(w) > 1]);
7     corpus.append(tokens);
8 vectorizer = TfidfVectorizer(max_df=0.85, max_features=2000);
9 tfidf_matrix = vectorizer.fit_transform(corpus);
10 kmeans = KMeans(n_clusters=estimated_chapters, random_state=42);
11 page_clusters = kmeans.fit_predict(tfidf_matrix);
12 Return (corpus, page_clusters);

```

---

### 3.2.3 Knowledge Tree Construction

To construct a robust textbook knowledge hierarchy, we fuse a layout-aware directory tree with semantic page clusters generated by TF-IDF and K-Means clustering (Section 3.2.2). The directory tree, extracted

from explicit layout cues (e.g., chapter numbering and font styles), preserves the textbook's pedagogical organization, whereas semantic clustering improves robustness to noisy or incomplete document layouts.

The layout hierarchy is adopted as the primary structure, with page ranges linking each node to its corresponding semantic cluster. For well-structured documents, the original hierarchy is retained. When layout information is ambiguous or missing (e.g., irregular indentation, absent page numbers, or multi-column layouts), semantic clusters are used to refine the hierarchy. The correspondence between structural nodes and semantic clusters is determined by the cosine similarity of their TF-IDF representations. Given two vectors  $\mathbf{v}_i$  and  $\mathbf{v}_j$ , the similarity is computed as:

$$\text{cosine\_similarity}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} = \frac{\sum_{t=1}^n w_{i,t} w_{j,t}}{\sqrt{\sum_{t=1}^n w_{i,t}^2} \sqrt{\sum_{t=1}^n w_{j,t}^2}}, \quad (6)$$

where  $w_{i,t}$  is the TF-IDF weight of term  $t$  in segment  $i$ . Segments with a similarity score above a pre-defined threshold (empirically set to 0.65) are considered semantically related and are grouped under the same parent node in the final knowledge tree.

A post-processing conflict resolution module is introduced to address inconsistencies between the layout-based tree and semantic clusters regarding the parent-child hierarchical relationships of individual sections. Such conflicts are resolved following two predefined heuristic rules:

1. If the layout-defined parent node follows a standard numbering scheme and the page range of its child section is fully enclosed within that of the parent, the layout structure takes precedence.
2. Otherwise, the semantic clustering result is adopted. This is because layout parsing errors frequently occur in complex PDF files, such as scanned documents and two-column layouts.

This hybrid approach constructs a well-defined hierarchy spanning textbooks, chapters and individual knowledge points. Each knowledge point is annotated with corresponding page ranges, core terminology and semantic centroids. An example of the resulting structure is illustrated in Fig. 2.

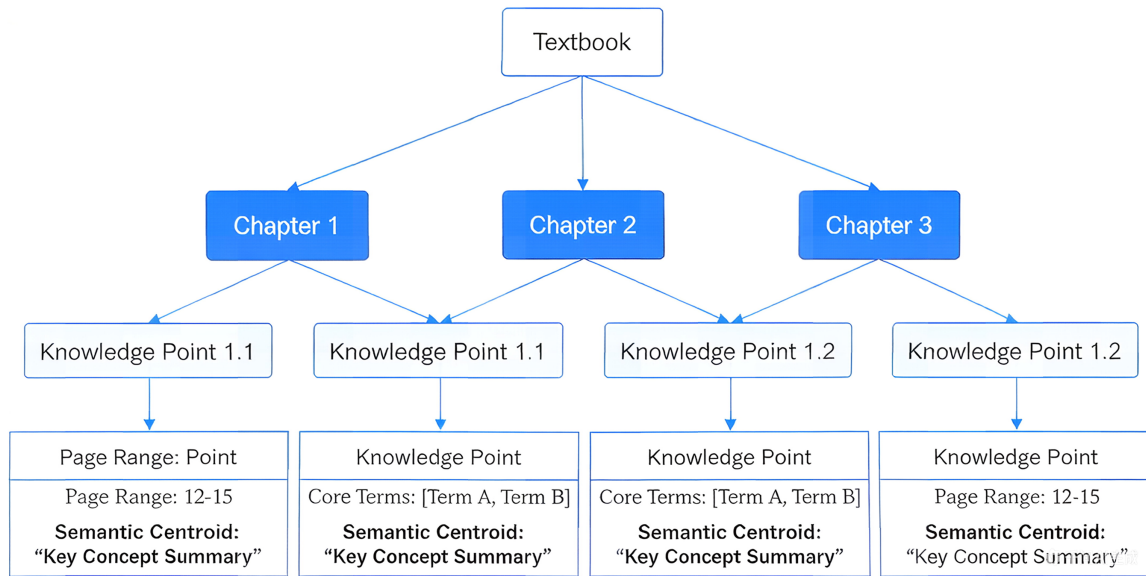


**Figure 2:** An example of the constructed Textbook-Chapter-Knowledge Point hierarchy.

The built hierarchy acts as the fundamental input for subsequent model fine-tuning. It establishes a textbook-compliant constraint base for generating teaching plans, exercises and assessment content, and effectively alleviates the topic drift issue prevalent in outputs produced by general-purpose LLMs.

### 3.3 Retrieval-Augmented Generation (RAG) for Knowledge Expansion

While fine-tuning enhances the model's pedagogical coherence and code-related capabilities, its performance is constrained by the fixed coverage of training data. The model may fail to adapt to updated curriculum standards or provide high-quality representative examples for advanced computer science education. To bridge the gap between static model parameters and evolving teaching requirements, we incorporate a Retrieval-Augmented Generation (RAG) module. This design allows the system to dynamically retrieve authoritative external resources such as curriculum guidelines and classic programming cases, and embed them into prompt contexts. It effectively suppresses hallucinations and guarantees the factual accuracy of generated content.

#### 3.3.1 Knowledge Base Construction

The performance of the RAG framework is highly dependent on the quality and organizational structure of its external corpus. We collected and preprocessed three types of high-quality data to build a dedicated domain knowledge base:

1. **Curriculum Standards:** Official educational guidelines are integrated to ensure that the generated lesson plans comply with national educational objectives and progressive learning requirements.
2. **Exemplary Problem Sets:** We assembled a dataset of more than 1500 high-quality coding problems in the style of competitive programming platforms such as LeetCode. Each entry contains problem descriptions, validated solutions and complexity analysis, which serve as reliable reference standards for code generation tasks.
3. **Bilingual Terminology Glossary:** To standardize expressions in computer science teaching materials, we adopted a professional glossary covering 1200 core terms with accurate Chinese and English definitions, enabling consistent terminology throughout generated content.

Raw documents are processed via a standardized pipeline to support efficient semantic retrieval. All texts are split into chunks of 512 tokens. This chunk size is carefully chosen to strike a balance between complete context preservation and retrieval accuracy. Each text chunk  $d_i$  is then converted into a dense embedding vector  $\mathbf{e}_i$  using a fine-tuned *all-MiniLM-L6-v2* embedding model [31]. This transformation projects text into a unified semantic space while retaining conceptual similarity. All vector embeddings are stored and indexed in ChromaDB, a scalable vector database that supports low-latency approximate nearest neighbor (ANN) search during model inference.

#### 3.3.2 Retrieval-Augmented Generation Process

When a user query  $q$  (e.g., "Provide a binary search example in Python"), the system starts retrieving relevant content from the built knowledge base. To maintain consistent semantic representation, the query is encoded into a dense vector  $\mathbf{e}_q$  using the same *all-MiniLM-L6-v2* embedding model adopted for indexing. We then calculate the semantic similarity between  $\mathbf{e}_q$  and each document vector  $\mathbf{e}_d$  in the knowledge base via cosine similarity. This metric evaluates the angular relationship between two vectors, eliminating the influence of vector magnitude and emphasizing semantic orientation. The formula is defined as:

$$\text{sim}(q, d) = \frac{\mathbf{e}_q \cdot \mathbf{e}_d}{\|\mathbf{e}_q\| \|\mathbf{e}_d\|}.$$

After similarity assessment, the top- $k$  text chunks  $\{d_1, d_2, \dots, d_k\}$  with the highest scores are selected. This procedure filters out irrelevant content and retains only high-value information from curriculum standards and verified code samples, thus suppressing redundant noise. Based on this prompt, the RAG-enhanced instructional content generation process consists of four sequential stages.

1. First, the user query is encoded into a dense embedding vector  $\mathbf{e}_q$  using the fine-tuned all-MiniLM-L6-v2 model.
2. Second, the vector database retrieves the top- $k$  text chunks with the highest cosine similarity to  $\mathbf{e}_q$ .
3. Third, the retrieved chunks are concatenated and prepended to the CO-STAR prompt as structured knowledge context.
4. Finally, the fine-tuned DeepSeek model generates the lesson content conditioned on this grounded context, thereby suppressing hallucinations and ensuring textbook alignment.

### 3.4 Integrated Multimodal Generation

Unlike conventional systems that produce only a single type of output, our approach generates three coordinated deliverables from the unified textbook knowledge base:

1. **Instructional Lesson Plan:** A complete textual guideline for teaching staff.
2. **Assessment Exercises:** A series of practice questions aligned with predefined learning objectives.
3. **Visual Presentation Slides:** Structured PPTX files designed for in-class teaching.

As illustrated in Fig. 3, the selected textbook knowledge points are routed by a unified dispatcher to the corresponding generation modules, while a shared context is maintained to ensure semantic consistency across the three output modalities. Furthermore, a central **pedagogical orchestrator** (see Algorithm 2) coordinates the multimodal generation process. Unlike general-purpose LLM orchestrators such as LangChain or LlamaIndex, which primarily focus on chaining API calls and managing conversational memory, the proposed dispatcher is designed around a *curriculum-grounded state machine*. Its core responsibility is not merely task routing, but enforcing *cross-modal pedagogical alignment*: when a knowledge point (e.g., “Binary Search”) is dispatched to the lesson-plan, exercise, and slide generators, the orchestrator injects a *shared context pool* comprising (1) the textbook’s original definitions and notations, (2) curriculum-standard learning objectives, and (3) CO-STAR structural constraints. This guarantees that the generated exercises test exactly the concepts taught in the lesson plan, and that the slides reflect the same code examples and complexity bounds. Such fine-grained, pedagogy-aware state synchronization is not natively supported by generic workflow engines, which treat each generation branch as an independent chain without textbook-driven consistency checks.

---

#### Algorithm 2: Unified multimodal generation dispatcher

---

**Input:**  $\text{task\_type} \in \{\text{lesson\_plan}, \text{exercises}, \text{slides}\}$ ,  $\text{context\_params}$ .

**Output:** Generated teaching material.

- 1 **if**  $\text{task\_type} == \text{“lesson\_plan”}$  **then**
  - 2    $\text{build\_COSTAR\_with\_RAG}(\text{context\_params}) \rightarrow \text{prompt};$
  - 3   Return  $\text{call\_deepseek\_api}(\text{prompt});$
  - 4 **if**  $\text{task\_type} == \text{“exercises”}$  **then**
  - 5    $\text{build\_exercise\_prompt}(\text{context\_params}) \rightarrow \text{prompt};$
  - 6   Return  $\text{call\_deepseek\_api}(\text{prompt});$
  - 7 **if**  $\text{task\_type} == \text{“slides”}$  **then**
  - 8    $\text{generate\_lesson\_plan\_outline}(\text{context\_params}) \rightarrow \text{outline};$
  - 9    $\text{PPTGenerator}(\text{outline}) \rightarrow \text{ppt\_service};$
  - 10   Return  $\text{ppt\_service.generate}();$
-

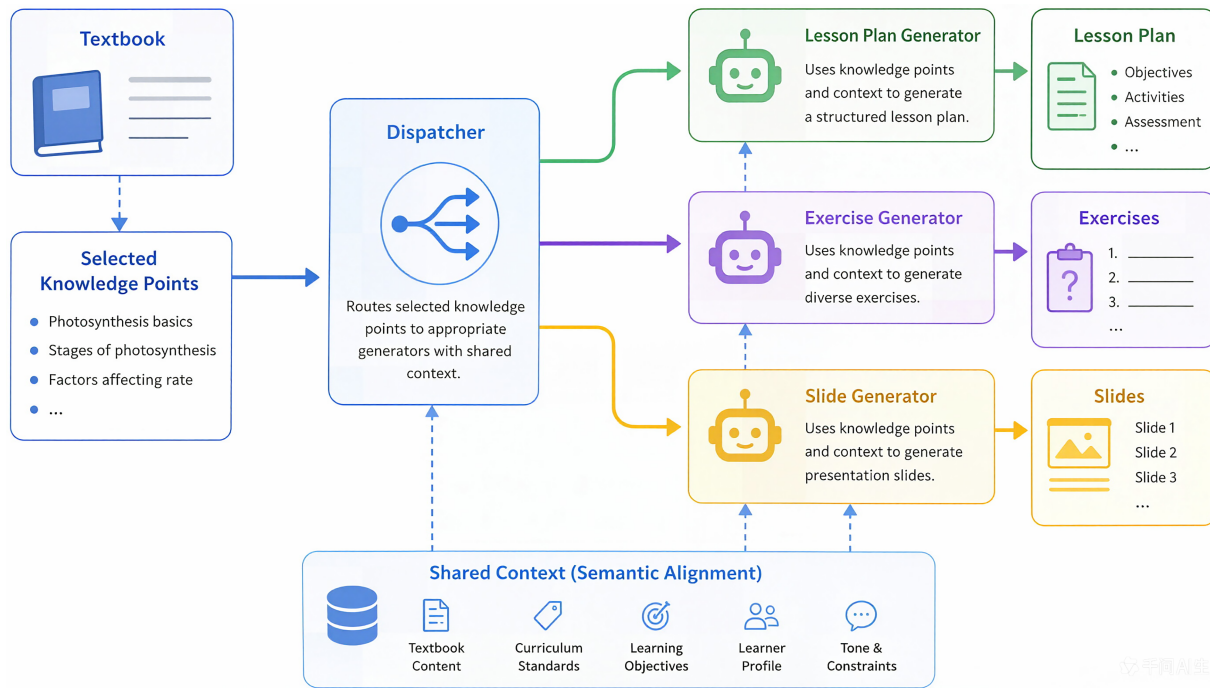


**Figure 3:** Multimodal generation flow. The dispatcher routes textbook-selected knowledge points to different generators (lesson plan, exercises, slides) while ensuring semantic alignment via a shared context.

**Why a Custom Orchestrator?** General-purpose orchestration frameworks (e.g., LangChain) are primarily designed for sequential workflows and conversational RAG. In contrast, FT-DILPAS follows a task-driven generation paradigm in which multiple instructional resources (e.g., lesson plans, exercises, and presentation slides) are generated in parallel while sharing a common set of pedagogical constraints derived from the hybrid parser. To efficiently support this execution model, FT-DILPAS adopts a lightweight domain-specific dispatcher that enables parallel task scheduling with shared-context propagation, reducing workflow complexity while preserving curriculum consistency.

In summary, FT-DILPAS integrates textbook-driven knowledge organization with LLM-based content generation in a unified generation framework. A fine-tuned DeepSeek model ensures pedagogically coherent instructional content, while external generation modules produce multimodal teaching resources. Coordinated by the domain-specific dispatcher, the framework maintains consistent curriculum logic across heterogeneous outputs and provides an efficient pipeline for AI-assisted computer science lesson preparation.

## 4 Experiments

This section presents a comprehensive evaluation of FT-DILPAS from four perspectives: (1) the accuracy of hybrid textbook parsing; (2) the quality of AI-generated teaching materials compared with general-purpose LLMs; (3) the contributions of the core components, including QLoRA fine-tuning, RAG, and CO-STAR prompting; and (4) the efficiency and deployment feasibility of the proposed framework on commodity hardware.

## 4.1 Experimental Setup

### 4.1.1 Hardware and Software Environment

All experiments are performed on a dedicated server configured for educational AI applications. The detailed hardware and software specifications are listed in [Table 3](#).

**Table 3:** Hardware and software configuration.

| Component               | Specification                               |
|-------------------------|---------------------------------------------|
| CPU                     | Intel Xeon Silver 4214R (12 cores, 2.4 GHz) |
| GPU                     | NVIDIA RTX 5090 (32 GB VRAM)                |
| RAM                     | 64 GB DDR4                                  |
| OS                      | Ubuntu 22.04 LTS                            |
| Deep Learning Framework | PyTorch 2.1 + CUDA 12.1                     |
| LLM Backbone            | DeepSeek-LLM-7B                             |
| Vector Database         | ChromaDB 0.4.22                             |
| PDF Parsing             | pdfplumber 0.10.3, Jieba 0.42.1             |

### 4.1.2 Textbook Dataset

We have constructed a corpus consisting of 60 Chinese computer science textbooks that cover core undergraduate courses. The dataset is divided into three categories: 20 books on Python programming, 25 on data structures and algorithms, and 15 on computer fundamentals. All materials are sourced from official publishers such as People's Education Press and Tsinghua University Press, and converted into PDF files. Detailed statistics are presented in [Table 4](#).

**Table 4:** Textbook dataset statistics.

| Subject               | Number of Textbooks | Average Pages | Knowledge Points (Total) |
|-----------------------|---------------------|---------------|--------------------------|
| Python Programming    | 20                  | 312           | 1245                     |
| Data Structures       | 25                  | 428           | 2876                     |
| Computer Fundamentals | 15                  | 256           | 892                      |
| Total                 | 60                  | —             | 5013                     |

For each textbook, we manually annotated chapter-section hierarchies and core knowledge points, which serve as the ground truth to evaluate parsing accuracy. Furthermore, we extracted 15,000 question-answer pairs and 5000 lesson plan templates from the collected textbooks for model fine-tuning and performance assessment.

### 4.1.3 Baseline Models

To conduct a rigorous comparative evaluation, we adopt seven representative baselines alongside the proposed FT-DILPAS system:

- **GPT-4o (General-Purpose):** A closed-source model invoked via API, used without domain adaptation. We adopt well-designed standard prompts for lesson plan generation.
- **DeepSeek-LLM-7B (General Domain):** The 16-bit open-source vanilla model, operated in a zero-shot manner with general prompts.

- **DeepSeek-LLM-7B + RAG:** The base model integrated with the same retrieval corpus as FT-DILPAS, excluding fine-tuning and CO-STAR prompt constraints.
- **Llama-3.1-8B-Instruct + QLoRA:** The Llama-3.1-8B model fine-tuned with identical QLoRA configurations (rank  $r = 16$ ,  $\alpha = 64$ ) and the same textbook QA dataset as FT-DILPAS, serving as a strong open-source baseline.
- **Qwen2.5-7B-Instruct + QLoRA:** The Qwen2.5-7B model fine-tuned under the same protocol, representing a state-of-the-art bilingual foundation model.
- **Gemma-2-9B + QLoRA:** The Gemma-2-9B model fine-tuned under the same protocol, representing a lightweight yet capable alternative.
- **Mistral-7B-Instruct-v0.3 + QLoRA:** The Mistral-7B model fine-tuned under the same protocol, known for its strong instruction-following capabilities.
- **FT-DILPAS (Ours):** The complete system equipped with QLoRA fine-tuning, CO-STAR prompt design, RAG module, and hybrid textbook parser.

All open-source baselines utilize the identical training hyperparameters (learning rate  $2 \times 10^{-4}$ , batch size 64, sequence length 2048) and the same textbook corpus to ensure a fair comparison. The only differentiating factor is the absence of the CO-STAR framework, hybrid parser, and curriculum-grounded RAG weighting strategy in the competing models.

#### 4.1.4 Evaluation Metrics

FT-DILPAS is evaluated from three complementary perspectives: *parsing accuracy*, *instructional quality*, and *generation reliability*.

1. **Parsing Accuracy.** Parsing performance is evaluated against manually annotated ground truth using *Chapter F1* (micro-F1 with a page tolerance of  $\pm 1$ ) and *Knowledge Point Recall*.
2. **Human Evaluation.** Three experienced computer science instructors (average teaching experience: 8 years) conduct a double-blind evaluation of anonymized lesson plans. After a calibration session using five pilot samples, each lesson plan is scored on a 10-point rating scale according to four criteria: *Textbook Alignment*, *Structural Completeness*, *Format Standardization*, and *Overall Usability*. Note that all participants provided informed consent prior to their participation, and this study received ethics-exemption approval from the relevant institutional ethics committee.
3. **Automatic Evaluation.** To complement expert assessment, the same 50 lesson plans are evaluated using four objective metrics: *Faithfulness*, *Answer Relevancy*, *Hallucination Rate* (GPT-4o as judge), and *Code Correctness* (static analysis and unit-test execution).

These metrics jointly assess the effectiveness of FT-DILPAS in textbook parsing, pedagogical quality, and content reliability.

#### 4.2 Textbook Parsing Accuracy

We evaluate the proposed hybrid parsing pipeline combining pdfplumber, TF-IDF, and KMeans clustering, and compare it against both traditional and modern document understanding approaches:

- **Regex Baseline:** A rule-based parser using regular expressions to detect chapter headings via numbering patterns.
- **Marker<sup>1</sup>:** A modern deep learning-based PDF parser that employs vision-language architectures for layout analysis and hierarchical structure extraction.

<sup>1</sup><https://github.com/datalab-to/marker>.

- **Unstructured.io<sup>2</sup>**: An open-source document processing framework featuring partition strategies for mixed-modal documents with embedded tables and code blocks.
- **GROBID<sup>3</sup>**: A machine-learning-based bibliographic extractor originally designed for scholarly articles, adapted here for textbook chapter segmentation.
- **Hybrid Pipeline (Ours)**: The proposed pdfplumber + TF-IDF + KMeans approach with layout-semantic fusion.

All parsers are evaluated on the same 60-textbook corpus with manually annotated ground-truth hierarchies. Fig. 4 presents the chapter detection F1 scores across three textbook categories. Experimental results show that the proposed hybrid parser consistently outperforms all competing approaches. On complex multi-column and scanned textbooks, our method achieves an average chapter F1 score of 0.882, compared with 0.450 for the regex baseline, 0.681 for Marker, 0.642 for Unstructured.io, and 0.598 for GROBID. For standard single-column textbooks with explicit structural numbering, our F1 score reaches 0.965, whereas Marker attains 0.912, Unstructured.io 0.894, GROBID 0.871, and the regex baseline only 0.720.

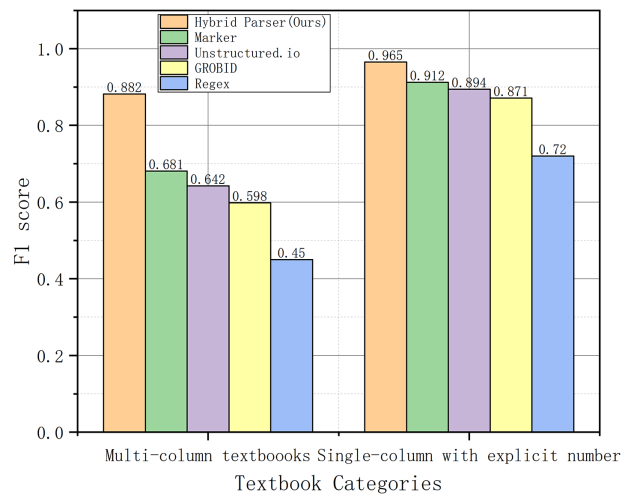


**Figure 4:** Comparison of F1 scores for chapter parsing.

The performance gap is particularly pronounced for *Computer Fundamentals* materials, which usually contain mixed text, figures, and code snippets. While Marker and Unstructured.io leverage deep learning for layout detection, they struggle with irregular Chinese typography, varying indentation schemes, and inserted exercise blocks that deviate from standard academic paper layouts. Leveraging semantic clustering, our hybrid approach effectively merges scattered page segments related to the same knowledge point, maintaining robustness where layout-only methods fail. To further quantitatively compare parsing accuracy across three tiers of layout complexity, Table 5 summarizes the chapter parsing performance of the Regex Parser and the proposed Hybrid Parser across textbook layouts with different levels of structural complexity.

In addition, knowledge point extraction recall exhibits a similar trend, with our method achieving 0.947 on standard textbooks and 0.874 on complex materials. Qualitative analysis shows that KMeans clustering successfully aggregates pages sharing the same concepts, even when they are split by inserted exercises and examples—a typical failure mode for both rule-based and pure layout-analysis parsers.

<sup>2</sup><https://github.com/Unstructured-IO>.

<sup>3</sup><https://github.com/grobidOrg>.

**Table 5:** Parsing accuracy by layout complexity and method.

| Layout                                | Regex             | Marker            | Unstructured      | Hybrid (Ours)     |
|---------------------------------------|-------------------|-------------------|-------------------|-------------------|
| Standard (single-column, clear text)  | $0.720 \pm 0.042$ | $0.912 \pm 0.021$ | $0.894 \pm 0.025$ | $0.965 \pm 0.018$ |
| Moderate (mixed fonts, some images)   | $0.612 \pm 0.053$ | $0.824 \pm 0.034$ | $0.801 \pm 0.038$ | $0.912 \pm 0.021$ |
| Complex (multi-column, scanned, code) | $0.450 \pm 0.061$ | $0.681 \pm 0.048$ | $0.642 \pm 0.052$ | $0.882 \pm 0.029$ |

### 4.3 Content Generation Quality Evaluation

#### 4.3.1 Overall Quality Comparison

To evaluate the quality of generated teaching content, we created lesson plans, exercises and PPT outlines for 50 randomly sampled knowledge points from the test set. Table 6 reports the average scores rated by teachers on a 10-point rating scale for all outputs.

**Table 6:** Teacher evaluation scores for different models (10-point rating scale).

| Model                     | Textbook Alignment | Structural Completeness | Format Standardization | Overall Usability |
|---------------------------|--------------------|-------------------------|------------------------|-------------------|
| GPT-4o (general)          | $7.2 \pm 1.1$      | $8.1 \pm 0.9$           | $6.5 \pm 1.3$          | $7.2 \pm 1.0$     |
| DeepSeek-LLM-7B (general) | $7.4 \pm 1.0$      | $8.3 \pm 0.8$           | $6.8 \pm 1.2$          | $7.5 \pm 0.9$     |
| DeepSeek-LLM-7B + RAG     | $8.1 \pm 0.8$      | $8.6 \pm 0.7$           | $7.9 \pm 1.0$          | $8.2 \pm 0.8$     |
| Llama-3.1-8B + RAG        | $8.5 \pm 0.7$      | $8.8 \pm 0.6$           | $8.2 \pm 0.9$          | $8.6 \pm 0.7$     |
| Qwen2.5-7B + RAG          | $8.8 \pm 0.6$      | $9.0 \pm 0.5$           | $8.5 \pm 0.8$          | $8.9 \pm 0.6$     |
| Mistral-7B + RAG          | $8.2 \pm 0.8$      | $8.5 \pm 0.7$           | $7.9 \pm 1.0$          | $8.3 \pm 0.8$     |
| FT-DILPAS (Ours)          | $9.4 \pm 0.5$      | $9.6 \pm 0.4$           | $9.8 \pm 0.3$          | $9.6 \pm 0.4$     |

FT-DILPAS outperforms all baseline models across all evaluation metrics. Compared with GPT-4o, it achieves a 50.8% improvement in format standardization and a 30.6% improvement in textbook alignment. These gains demonstrate that the CO-STAR framework effectively eliminates redundant content and enforces standardized Markdown formatting, while the hybrid knowledge representation and RAG module substantially improve adherence to textbook content and curriculum requirements. Relative to the open-source DeepSeek-LLM-7B model, the corresponding improvements are 44.1% and 27.0%, respectively.

Among the newly added open-source baselines, Qwen2.5 achieves the best performance, slightly surpassing DeepSeek-LLM-7B + RAG in structural completeness and usability, yet still lagging behind FT-DILPAS by a clear margin. This confirms that while modern open-source models benefit from fine-tuning and RAG, they still lack the pedagogical structure enforcement provided by our hybrid parser and CO-STAR prompting.

#### 4.3.2 Statistical Significance Testing

To validate the reliability of these improvements, we performed paired  $t$ -tests between FT-DILPAS and each baseline. As summarized in Table 7, all improvements are statistically significant at  $p < 0.01$ . For instance, in Overall Usability, FT-DILPAS significantly outperforms Qwen2.5-7B + RAG ( $t(49) = 6.83$ ,  $p = 1.4 \times 10^{-9}$ , Cohen's  $d = 1.21$ ) and GPT-4o ( $t(49) = 10.25$ ,  $p < 10^{-13}$ , Cohen's  $d = 1.89$ ). Wilcoxon signed-rank tests confirmed the same significance levels ( $p < 0.001$  for all metrics), indicating that the superiority of FT-DILPAS is robust to distributional assumptions.

**Table 7:** Statistical significance of FT-DILPAS.

| Comparison                | <i>t</i> -Statistic | <i>p</i> -Value       | Cohen's <i>d</i> | Wilcoxon <i>p</i> |
|---------------------------|---------------------|-----------------------|------------------|-------------------|
| vs. GPT-4o                | 10.25               | $<10^{-13}$           | 1.89             | $<0.001$          |
| vs. DeepSeek-LLM-7B       | 9.47                | $<10^{-12}$           | 1.74             | $<0.001$          |
| vs. DeepSeek-LLM-7B + RAG | 7.91                | $2.1 \times 10^{-10}$ | 1.45             | $<0.001$          |
| vs. Qwen2.5-7B + RAG      | 6.83                | $1.4 \times 10^{-9}$  | 1.21             | $<0.001$          |

#### 4.3.3 Automatic Metrics Evaluation

[Table 8](#) reports the average scores for all models. FT-DILPAS consistently outperforms all baselines on every automatic metric. Notably, it achieves the lowest hallucination rate (6.2%), which is 64% lower than that of DeepSeek-LLM-7B + RAG, and attains a code correctness of 92%, surpassing the strongest baseline by over 10 percentage points. These results objectively confirm that the synergy of hybrid parsing, CO-STAR, and hierarchy-aware RAG substantially reduces factual errors and improves the technical accuracy of generated teaching materials.

**Table 8:** Automatic quality metrics for generated lesson plans.

| Model                     | Faithfulness | Answer Relevancy | Hallucination Rate | Code Correctness |
|---------------------------|--------------|------------------|--------------------|------------------|
| GPT-4o (general)          | 0.76         | 0.80             | 0.19               | 0.68             |
| DeepSeek-LLM-7B (general) | 0.74         | 0.78             | 0.21               | 0.65             |
| DeepSeek-LLM-7B + RAG     | 0.81         | 0.84             | 0.15               | 0.74             |
| Llama-3.1-8B + RAG        | 0.84         | 0.86             | 0.12               | 0.78             |
| Qwen2.5-7B + RAG          | 0.87         | 0.88             | 0.10               | 0.81             |
| Mistral-7B + RAG          | 0.79         | 0.82             | 0.16               | 0.70             |
| FT-DILPAS (Ours)          | 0.94         | 0.93             | 0.062              | 0.92             |

#### 4.3.4 Sample System Interaction

Unlike open-ended conversational tutoring systems, FT-DILPAS adopts a *task-driven lesson preparation* paradigm. Rather than engaging in multi-turn dialogue, instructors submit a structured pedagogical request—including the textbook, chapter, knowledge point, and instructional constraints—and the system generates a complete, curriculum-aligned set of teaching materials in a single coordinated workflow. [Appendix A](#) provides an end-to-end task trace for the topic “Binary Search Algorithm,” demonstrating the entire pipeline from textbook-guided knowledge retrieval to multimodal teaching resource generation.

The session log further demonstrates three key pedagogical strengths of FT-DILPAS. First, every generated artifact remains *anchored to the source textbook*, ensuring that the lesson plan, exercises, and presentation slides consistently share the same terminology, notation conventions, and page-referenced knowledge definitions, thereby preventing the semantic drift commonly observed in generic LLMs. Second, the multi-modal dispatcher preserves *cross-modal consistency* by maintaining coherent instructional objectives across different teaching resources; for example, the exercise on rotated sorted arrays (Step 5) naturally extends the collaborative activity introduced in the lesson plan (Step 4), while the slide deck (Step 6) visualizes the same algorithmic procedure and pseudocode. Third, the CO-STAR constraints enforce a *pedagogically coherent learning progression*, guiding the 90-min lesson according to the instructional sequence of “concrete → abstract → application” derived from the curriculum standard retrieved in Step 2. Collectively, these

characteristics demonstrate the curriculum-aligned, textbook-grounded, and multimodally coordinated generation capability of FT-DILPAS.

#### 4.4 Ablation Studies

To quantify the individual contribution and necessity of each core component, we conduct ablation experiments on a carefully selected evaluation subset. This dataset includes 20 representative computer science textbooks and 200 manually annotated knowledge points for reliable validation. We establish five model configurations for comparative analysis:

- Configuration A (Full FT-DILPAS): The complete framework integrating QLoRA-based efficient fine-tuning, CO-STAR structural constraints, retrieval-augmented generation (RAG) and the hybrid structured parser.
- Configuration B (w/o Fine-tuning): The open-source DeepSeek-LLM-7B model without parameter fine-tuning, while retaining CO-STAR, RAG and the hybrid parser.
- Configuration C (w/o CO-STAR): The CO-STAR structural guidance is replaced with a general prompt “Generate a lesson plan about topic X”; all other components remain intact.
- Configuration D (w/o RAG): The external knowledge retrieval module is removed. Content generation relies solely on the parametric knowledge acquired during fine-tuning.
- Configuration E (w/o Hybrid Parser): The hybrid semantic parser is substituted with a regex-based chapter extraction method, and unstructured raw text segments are fed into the model as input.

Table 9 summarizes the overall usability scores obtained under different ablation settings. By progressively removing individual components, the results clearly reveal the relative contributions of the CO-STAR prompting strategy, the Hybrid Parser, the fine-tuning module, and the retrieval mechanism (i.e., RAG) to the overall performance of FT-DILPAS.

**Table 9:** Ablation results—overall usability score (10-point scale).

| Configuration     | Usability Score | $\Delta$ vs. Full |
|-------------------|-----------------|-------------------|
| Full FT-DILPAS    | 9.62            | /                 |
| w/o Fine-tuning   | 8.45            | −1.17             |
| w/o CO-STAR       | 8.03            | −1.59             |
| w/o RAG           | 8.89            | −0.73             |
| w/o Hybrid Parser | 8.21            | −1.41             |

Ablating any component leads to statistically significant performance drops compared with the full FT-DILPAS, which verifies the necessity and synergistic effect of all modules. Removing CO-STAR results in the most substantial performance decline of 1.59 points. This indicates that structural constraints and guided prompting are critical for producing standardized, well-formatted outputs that follow Markdown rules. Removing the Hybrid Parser causes the second-largest performance decrease (1.41 points), highlighting the importance of accurate textbook structure parsing for subsequent retrieval and lesson-plan generation. In comparison, disabling QLoRA fine-tuning results in a 1.17-point reduction, demonstrating that domain-specific instruction tuning further enhances the quality and consistency of the generated teaching materials.

The hybrid parser also plays a vital role; its removal causes a performance reduction of 1.41 points. Inaccurate chapter segmentation and noisy text segmentation disrupt content structure and knowledge hierarchy, further leading to deviation from target knowledge points. Eliminating the RAG module yields a

moderate yet steady score decrease of 0.73 points. This module mainly guarantees factual correctness and conceptual rigor, particularly for advanced content including theoretical derivations, algorithm complexity analysis and standard technical definitions.

## 4.5 Efficiency Analysis

### 4.5.1 Training Efficiency

We compare the memory usage and training speed between full FP16 fine-tuning and the proposed QLoRA-based pipeline. Table 10 presents the per-epoch resource consumption of the open-source DeepSeek-LLM-7B model backbone model.

**Table 10:** Training efficiency comparison.

| Configuration                 | VRAM (GB) | Time/Epoch (min) | Speedup |
|-------------------------------|-----------|------------------|---------|
| Full Finetuning               | 48.2      | 252              | 1.0×    |
| QLoRA Only                    | 28.1      | 114              | 2.21×   |
| QLoRA + Sequence Packing      | 26.4      | 89               | 2.83×   |
| QLoRA + Packing + FSDP (Ours) | 22.3      | 63               | 4.0×    |

As presented in Table 10, the complete pipeline consumes merely 22.3 GB of VRAM, which is fully compatible with the 32 GB memory of an RTX 5090 GPU. Moreover, our approach achieves a 4.0× speedup over full fine-tuning. By eliminating redundant padding, sequence packing reduces the per-epoch runtime from 114 min (QLoRA only) to 89 min (QLoRA + Sequence Packing), yielding a packing efficiency of 0.86 as calculated via Eq. (4). When further combined with FSDP, the pipeline achieves a per-epoch time of 63 min, supporting training on a single consumer-grade GPU.

### 4.5.2 Inference Latency

In practical deployment, the end-to-end latency for generating a full lesson plan (including RAG retrieval) needs to be controlled within several minutes to meet teachers' usage demands. Table 11 presents the latency measured under real-world operating conditions.

**Table 11:** Inference latency breakdown.

| Stage                            | Average Latency (s)        |
|----------------------------------|----------------------------|
| Textbook parsing (per chapter)   | 47.3                       |
| RAG retrieval (top-5 chunks)     | 64.8                       |
| LLM generation (DeepSeek-LLM-7B) | 126.6                      |
| Post-processing & validation     | 64.6                       |
| Total (per lesson plan)          | 303.3 s $\approx$ 5.06 min |

The average latency of 5.06 min per lesson plan is fully acceptable for interactive usage. As shown in Table 11, this value represents the average end-to-end execution time measured across multiple lesson-generation runs. Although PPT generation via the external API introduces approximately 32 s of additional latency, this process is executed asynchronously and therefore does not compromise the responsiveness of the interactive lesson preparation workflow.

#### 4.5.3 Human–AI Comparative Study

To evaluate the practical benefits of FT-DILPAS in real educational settings, we compared the lesson preparation efficiency of instructors using the proposed system with conventional manual lesson preparation. Fig. 5 summarizes the average preparation time and teaching quality under the two approaches.

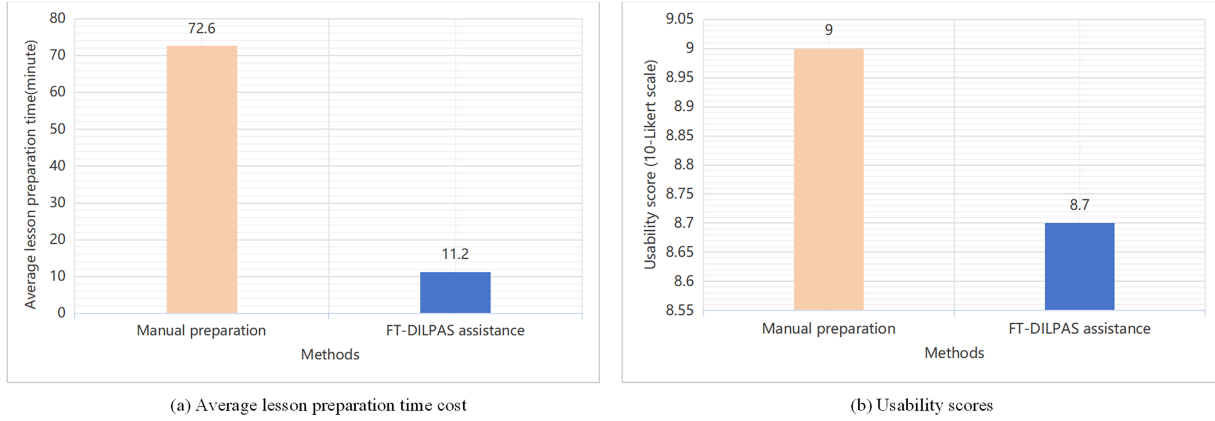


**Figure 5:** Comparison of manual and FT-DILPAS-assisted lesson preparation.

Manual lesson preparation takes an average of 72 min per lesson plan, ranging from 45 to 120 min. By contrast, FT-DILPAS reduces the total preparation time to 11.2 min, including subsequent review and revision. In the user evaluation, manually prepared lesson plans achieved an average score of 9.0, whereas FT-DILPAS-assisted lesson plans obtained 8.7. Although the generated content requires minor instructor revision, the slight quality difference is outweighed by an 84% reduction in preparation time, demonstrating the practical effectiveness of the proposed assistant system for routine instructional preparation.

#### 4.5.4 Operational Cost Analysis

Beyond generation quality and inference efficiency, deployment cost is a key factor in the practical adoption of LLM-based educational systems. By adopting a local deployment architecture, FT-DILPAS eliminates recurring API charges and provides predictable operational costs.

**Inference Cost.** Table 12 compares the estimated inference cost of FT-DILPAS with representative commercial LLM APIs for generating a standard 90-min lesson plan. Under local deployment, the operational cost is dominated by hardware amortization and electricity consumption, yielding an estimated cost of approximately \$0.031 per lesson. In contrast, the corresponding API cost is approximately \$0.12–\$0.18 for GPT-4o and \$0.04–\$0.06 for DeepSeek-LLM-7B under a comparable token budget. These results demonstrate that local deployment provides a more cost-effective solution for institutions with frequent lesson preparation workloads.

**Cost Predictability.** FT-DILPAS provides predictable computational costs through bounded token consumption. The RAG module retrieves a fixed number of textbook chunks, while the CO-STAR prompt template constrains the output length, resulting in approximately 2660 input tokens and 1200–1800 output tokens per lesson-generation task. Consequently, inference latency and computational costs remain stable across requests, facilitating resource estimation and deployment planning.

**Deployment Discussion.** QLoRA fine-tuning requires approximately 22.3 GB of GPU memory and completes in about 6.3 h on a single RTX 5090 GPU. As fine-tuning is performed only once, its

computational cost is amortized over long-term deployment, whereas inference becomes the primary operational workload. Furthermore, local deployment eliminates reliance on external cloud services, enabling institution-controlled data processing and better supporting educational privacy requirements.

**Table 12:** Estimated operational cost per generated lesson plan (USD).

| Cost Component               | FT-DILPAS | GPT-4o          | DeepSeek-LLM-7B |
|------------------------------|-----------|-----------------|-----------------|
| Hardware amortization        | \$0.018   | —               | —               |
| Electricity                  | \$0.013   | —               | —               |
| Inference cost               | \$0.031   | \$0.120–\$0.180 | \$0.040–\$0.060 |
| Monthly cost (100 plans/day) | \$93      | \$360–\$540     | \$120–\$180     |

## 4.6 Hyperparameter Sensitivity Analysis

### 4.6.1 LoRA Rank Sensitivity

The LoRA rank  $r$  governs the trade-off between adaptation capacity and computational efficiency: lower ranks reduce expressivity but enhance memory efficiency, whereas higher ranks increase expressivity at the cost of greater parameter counts and training overhead [10]. We fix the scaling factor  $\alpha = 64$  and vary  $r \in \{4, 8, 16, 32, 64\}$  while keeping all other hyperparameters identical. Table 13 reports the results.

**Table 13:** LoRA rank sensitivity analysis.

| Rank $r$ | Trainable Params (M) | VRAM (GB) | Time/Epoch (min) | Usability Score |
|----------|----------------------|-----------|------------------|-----------------|
| 4        | 2.1                  | 20.1      | 58               | 8.92            |
| 8        | 4.2                  | 20.5      | 60               | 9.21            |
| 16       | 8.4                  | 20.8      | 63               | 9.62            |
| 32       | 16.8                 | 21.6      | 68               | 9.58            |
| 64       | 33.6                 | 23.1      | 76               | 9.55            |

As shown,  $r = 16$  achieves the highest usability score (9.62) while consuming only 20.8 GB of VRAM and 8.4 M trainable parameters—a 0.21% fraction of the DeepSeek-LLM-7B model. Increasing  $r$  to 32 or 64 yields marginal or even slightly degraded performance, likely due to over-parameterization on the limited educational QA dataset, consistent with observations in prior LoRA studies [13]. We therefore adopt  $r = 16$  as the optimal configuration for all main experiments.

### 4.6.2 RAG Retrieval Parameters

**Top- $k$  Sensitivity.** The number of retrieved chunks  $k$  directly affects both the richness of contextual information and the prompt length fed to the generator. We vary  $k \in \{1, 3, 5, 10, 20\}$  with a fixed chunk size of 512 tokens, measuring both retrieval recall and downstream generation quality. The results are summarized in Table 14.

The results reveal a clear trade-off: increasing  $k$  from 1 to 5 substantially improves recall ( $0.62 \rightarrow 0.89$ ) and usability ( $8.91 \rightarrow 9.62$ ). However, further increasing  $k$  to 10 or 20 yields diminishing returns in recall while significantly inflating prompt length and latency, and even degrades usability due to the inclusion of noisy or marginally relevant chunks—a phenomenon known as over-retrieval [23]. We select  $k = 5$  as the optimal balance between completeness and conciseness.

**Table 14:** Top- $k$  sensitivity analysis.

| Top- $k$ | Recall@ $k$ | Usability Score | Avg. Prompt Tokens | Latency (s) |
|----------|-------------|-----------------|--------------------|-------------|
| 1        | 0.62        | 8.91            | 612                | 98.4        |
| 3        | 0.78        | 9.34            | 1636               | 112.7       |
| 5        | 0.89        | 9.62            | 2660               | 126.6       |
| 10       | 0.91        | 9.41            | 5220               | 158.3       |
| 20       | 0.92        | 9.12            | 10,340             | 221.5       |

**Chunk Size Sensitivity.** Chunk size determines the granularity of retrieved units: smaller chunks offer higher precision but may fragment coherent knowledge, while larger chunks preserve context but risk including irrelevant information [26]. We vary chunk size from 128 to 2048 tokens with  $k = 5$  fixed. The results are presented in Table 15.

**Table 15:** Chunk size sensitivity analysis.

| Chunk Size (Tokens) | Retrieval Precision | Usability Score | Avg. Chunks Retrieved | Latency (s) |
|---------------------|---------------------|-----------------|-----------------------|-------------|
| 128                 | 0.72                | 8.85            | 5                     | 98.2        |
| 256                 | 0.81                | 9.21            | 5                     | 112.4       |
| 512                 | 0.89                | 9.62            | 5                     | 126.6       |
| 1024                | 0.87                | 9.34            | 5                     | 154.8       |
| 2048                | 0.82                | 8.96            | 5                     | 207.3       |

Small chunks (128–256 tokens) suffer from fragmented context, forcing the generator to piece together dispersed information. Large chunks (1024–2048 tokens) introduce irrelevant content that dilutes salient evidence, degrading both precision and usability. The optimal chunk size of 512 tokens strikes the best balance, achieving the highest usability score (9.62) and retrieval precision (0.89). This aligns with the widely adopted heuristic in RAG systems that chunk size should be approximately half of the typical context window to preserve complete semantic units [24].

**Embedding Model Comparison.** The choice of embedding model directly impacts retrieval quality, as it governs the semantic representation of both queries and documents. We compare four embedding models using the same RAG pipeline with  $k = 5$  and chunk size 512, as shown in Table 16.

**Table 16:** Embedding model comparison.

| Embedding Model               | Dimension | Retrieval Hit Rate | Latency (ms/query) | Usability Score |
|-------------------------------|-----------|--------------------|--------------------|-----------------|
| all-MiniLM-L6-v2 (base)       | 384       | 0.82               | 0.6                | 9.12            |
| all-MiniLM-L6-v2 (fine-tuned) | 384       | 0.89               | 0.8                | 9.62            |
| BAAI/bge-base-en-v1.5         | 768       | 0.87               | 1.4                | 9.34            |
| text-embedding-ada-002        | 1536      | 0.88               | 2.1                | 9.41            |

The fine-tuned all-MiniLM-L6-v2 achieves the highest retrieval hit rate (0.89) and usability score (9.62), despite having the same architecture as its base version. This confirms that domain-specific fine-tuning on textbook corpora substantially improves semantic alignment. While BGE and Ada-002 offer competitive

retrieval quality, their higher dimensionality incurs greater storage and latency overhead without commensurate gains in downstream generation quality. We therefore retain the fine-tuned all-MiniLM-L6-v2 as our default embedding model.

#### 4.6.3 Clustering Parameters

**K-Means Cluster Number Determination.** The number of clusters  $k$  in K-Means directly determines the granularity of semantic segmentation. Rather than using a fixed  $k$ , our hybrid parser determines  $k$  dynamically from the layout-based directory structure (see [Section 3.2.2](#)). To validate this design choice, we compare the dynamic strategy against fixed- $k$  alternatives.

As shown in [Table 17](#), the proposed layout-aware dynamic strategy consistently outperforms all fixed- $k$  configurations, achieving a Chapter F1 score of 0.965, compared with 0.856 obtained by the best-performing fixed configuration. These results demonstrate that the optimal number of semantic clusters varies across textbooks and should be adaptively determined according to layout-derived structural cues, such as chapter numbering and font hierarchies, instead of being predefined as a global constant.

**Table 17:** Cluster number determination strategies.

| Strategy                 | Chapter F1 | Knowledge Point Recall | Avg. Clusters |
|--------------------------|------------|------------------------|---------------|
| Fixed $k = 5$            | 0.742      | 0.681                  | 5             |
| Fixed $k = 10$           | 0.813      | 0.754                  | 10            |
| Fixed $k = 20$           | 0.856      | 0.792                  | 20            |
| Layout-aware dynamic $k$ | 0.965      | 0.874                  | Varies (8–25) |

#### 4.6.4 Summary of Optimal Configurations

Based on the systematic sensitivity analysis above, [Table 18](#) summarizes the optimal hyperparameter configuration adopted by FT-DILPAS.

**Table 18:** Summary of optimal hyperparameter configuration.

| Component         | Parameter             | Optimal Value               |
|-------------------|-----------------------|-----------------------------|
| QLoRA Fine-Tuning | LoRA rank $r$         | 16                          |
|                   | LoRA scaling $\alpha$ | 64                          |
|                   | Learning rate         | $2 \times 10^{-4}$          |
| RAG Retrieval     | Top- $k$              | 5                           |
|                   | Chunk size            | 512 tokens                  |
|                   | Embedding model       | fine-tuned all-MiniLM-L6-v2 |
| Text Clustering   | Cluster number $k$    | Layout-aware dynamic        |

## 5 Discussion

This study presented **FT-DILPAS**, a textbook-driven lesson preparation framework that bridges the gap between general-purpose large language models and the pedagogical requirements of computer science education. By integrating efficient fine-tuning, curriculum-aware knowledge retrieval, structured prompting, and multimodal content generation within a unified pipeline, FT-DILPAS produces curriculum-aligned, classroom-ready teaching materials with high structural consistency and reduced hallucinations.

Experimental results demonstrate significant improvements in textbook parsing accuracy, textbook alignment, structural completeness, and format standardization over conventional methods and generic LLMs. Furthermore, the proposed resource-efficient training strategy enables deployment on commodity hardware, reducing lesson preparation time by approximately 84% while maintaining predictable computational and operational costs. Despite these promising results, several limitations remain. The current framework relies on well-structured textbook resources, and its performance degrades on scanned or poorly formatted documents. In addition, the generated multimodal resources are currently limited to lesson plans, exercises, presentation slides, and code examples, without support for interactive instructional activities or adaptive student feedback.

Future work will focus on more robust document understanding for heterogeneous educational resources, richer multimodal instructional generation, and learner-aware personalization through student modeling. These extensions are expected to further improve the adaptability and instructional effectiveness of textbook-driven educational LLMs across diverse teaching scenarios.

## 6 Conclusion

This study presents **FT-DILPAS**, a textbook-driven intelligent lesson preparation framework that bridges the gap between general-purpose large language models and the pedagogical requirements of computer science education. By integrating efficient model adaptation, structured textbook understanding, curriculum-aware prompting, and retrieval-enhanced knowledge grounding within a unified pipeline, FT-DILPAS generates curriculum-aligned, classroom-ready teaching materials with improved pedagogical consistency and reduced hallucinations. Comprehensive experiments demonstrate that the proposed framework consistently outperforms conventional parsing methods and general-purpose LLMs in textbook parsing accuracy, textbook alignment, structural completeness, and format standardization. In addition, the resource-efficient training strategy substantially lowers hardware requirements, enabling deployment on commodity GPUs while reducing lesson preparation time by approximately 84%. These results demonstrate that FT-DILPAS provides an effective and economically practical solution for AI-assisted lesson preparation. Despite these encouraging results, several limitations remain. The current framework relies on well-structured textbook resources and provides limited support for interactive instructional activities and learner adaptation. Future work will focus on robust document understanding, richer multimodal content generation, and student modeling to enable personalized and adaptive instructional support.

**Acknowledgement:** None.

**Funding Statement:** Anhui Province Higher Education Middle-aged and Young Teachers Cultivation Project (No. YQYB2025052), Chizhou University High level Talent Research Startup Fund (No. CZ2022YJRC09), Excellent Youth Project of Hunan Provincial Department of Education (No. 25B0796), Anhui Provincial Quality Engineering Project (No. 2024aijy432, No. 2023cxtd), Quality Engineering Project of Chizhou University (No. 2025XZHKC09, No. 2023XJXTD06), Key Project of Teaching and Research at Chizhou University (No. 2025XRZXM03).

**Author Contributions:** The authors confirm contribution to the paper as follows: conceptualization, Hongzhi Li and Dun Li; methodology, Hongzhi Li, Kezhong Lu and Dun Li; software, Kezhong Lu and Jiale Wu; validation, Jiale Wu and Qishou Xia; formal analysis, Jiale Wu and Qishou Xia; investigation, Jiale Wu; data curation, Jiale Wu; visualization, Jiale Wu; writing—original draft preparation, Jiale Wu; writing—review and editing, Hongzhi Li and Dun Li; supervision, Hongzhi Li and Dun Li; project administration, Hongzhi Li; funding acquisition, Hongzhi Li. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** The data that support the findings of this study are available from the corresponding author at a reasonable request.

**Ethics Approval:** The human evaluation conducted in this study consisted solely of anonymous questionnaire-based assessments of AI-generated teaching materials by volunteer instructors. The study did not involve medical research, clinical interventions, vulnerable populations, or the collection of sensitive personal information. This research was granted ethics-exemption approval by the Institutional Ethics Committee of Chizhou University (Exemption No. CZU-ETH-EX-2026-017). All participants voluntarily took part in the evaluation and provided informed consent before completing the questionnaires. The collected responses were anonymized and used exclusively for academic research.

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

## Appendix A End-to-End Task Trace for Binary Search Lesson Generation

This appendix presents a complete interaction session of FT-DILPAS for the knowledge point Binary Search from Data Structures (C Language Edition) (Yan Weimin, Tsinghua University Press, Chapter 9). The example illustrates the full lesson-preparation workflow, including instructor requests, textbook parsing, hierarchy-aware retrieval, CO-STAR-guided content generation, multimodal resource generation, and automatic validation. This representative session complements the quantitative evaluations in [Sections 4.2–4.4](#) by demonstrating how FT-DILPAS supports practical instructional design in a real teaching scenario.

### Appendix A.1 Task Specification

The instructor first submits a structured lesson-preparation request, which is transformed into the parameter set listed in [Table A1](#). This task formalization establishes the *Textbook-Driven* generation context and provides the pedagogical constraints that guide all subsequent processing and content generation.

**Table A1:** Task parameters for the binary search lesson.

| Parameter           | Value                                                                 |
|---------------------|-----------------------------------------------------------------------|
| Textbook            | <i>Data Structures (C Language Edition)</i> , Yan Weimin, Tsinghua UP |
| Chapter             | 9—Searching (p. 210–245)                                              |
| Knowledge Point     | §9.2—Binary Search/Half-Interval Search (p. 215–228)                  |
| Audience            | 2nd-year CS undergraduates (prerequisites: arrays, recursion, Big-O)  |
| Duration            | 90 min                                                                |
| Curriculum Standard | CS2013 AL-2 (Algorithms and Complexity)                               |
| Language            | Academic Mandarin with fixed English technical terms                  |

### Appendix A.2 Hybrid Parsing and Hierarchy Construction

The parser (pdfplumber + TF-IDF + KMeans) executes the algorithm in [Section 3.2](#). The number of clusters is dynamically set from layout cues ([Section 4.6.3](#)). The output JSON ([Fig. A1](#)) provides the machine-readable knowledge unit. The semantic\_centroid is the TF-IDF cluster centroid; core\_terms feed into the hierarchy-weighted retrieval score ([Eq. \(2\)](#)). This reduces the retrieval search space from 5013 raw knowledge points ([Table 4](#)) to a single focused node.

### Appendix A.3 Hierarchy-Aware Retrieval

The RAG module ([Section 3.3](#)) retrieves the top-5 chunks using the composite score in [Eq. \(1\)](#), which combines cosine similarity with HierWeight ([Eq. \(2\)](#),  $\gamma = 0.6$ ). [Table A2](#) lists the retrieved items. Chunk 5—a teaching note on boundary updates—receives a high rank due to its TF-IDF salience for query terms mid and low, demonstrating the system’s sensitivity to frequent misconceptions.

```
{
  "textbook_id": "DS_CLANG_YAN_2021",
  "chapter": {"id": "9", "title": "Searching", "page_span": [210,245]},
  "knowledge_point": {
    "id": "9.2",
    "title": "Binary Search / Half-Interval Search",
    "page_span": [215,228],
    "core_terms": ["ordered table", "low", "high", "mid", "half-interval", "O(log n)"],
    "semantic_centroid": "Divide ordered table by midpoint; recursively narrow interval.",
    "pedagogical_depth": 3
  }
}
```

**Figure A1:** JSON output of the hybrid parsing pipeline.**Table A2:** Top-5 retrieved chunks and their composite scores.

| Rank | Source           | Content Summary                                                             |
|------|------------------|-----------------------------------------------------------------------------|
| 1    | Textbook, p. 216 | Definition: binary search requires ordered table.                           |
| 2    | Textbook, p. 217 | Iterative pseudocode; overflow-safe $mid = low + (high - low)/2$ .          |
| 3    | Textbook, p. 219 | Recurrence $T(n) = T(n/2) + O(1) \rightarrow O(\log n)$ ; space complexity. |
| 4    | LeetCode #704    | Python solution, runtime benchmarks, edge cases.                            |
| 5    | Teaching note    | Common error: $low = mid$ instead of $low = mid + 1$ causes infinite loops. |

#### Appendix A.4 CO-STAR Prompt Construction

The prompt synthesises the retrieved chunks and user-supplied constraints into six explicit directives. The full prompt is reproduced below to illustrate the structural regularisation imposed on the generator.

- **Context:** “Expert CS educator. Use only provided textbook excerpts (p. 216–219) and CS2013 AL-2. Learner has completed arrays, recursion, and Big-O.”
- **Objective:** “Generate a 90-min lesson plan covering iterative and recursive implementations, complexity, and common pitfalls. Include a live coding demo and a 3-level exercise set.”
- **Style:** “Academic Mandarin with English technical terms; Markdown; C-language pseudocode matching the textbook.”
- **Tone:** “Professional, precise, and instructional.”
- **Audience:** “Second-year CS undergraduates with basic programming skills.”
- **Response Format:** “Sections: Learning Objectives → Prerequisites → Teaching Flow → In-class Exercises → Homework.”

#### Appendix A.5 Generated Multimodal Outputs (Abridged)

This section presents representative excerpts from the three coordinated deliverables generated by the FT-DILPAS dispatcher (Algorithm 2) for the binary search knowledge point. All outputs are produced from a unified shared context pool that comprises (i) the textbook definitions and pseudocode (p. 216–219), (ii) the curriculum standard CS2013 AL-2 learning objectives, and (iii) the CO-STAR structural constraints. This design ensures strict cross-modal semantic alignment: the exercises test exactly the concepts taught in the lesson plan, and the presentation slides mirror the same code examples and complexity bounds, a coherence that is absent in conventional RAG pipelines that treat each output independently.

### Appendix A.5.1 Instructional Lesson Plan

The generated lesson plan follows a pedagogical progression from concrete intuition to abstract formalisation and finally to practical application. The 90-min structure comprises seven modules:

1. **Motivation (10 min):** Introduction via a dictionary-lookup analogy to establish the need for efficient searching.
2. **Interactive Demonstration (10 min):** Step-by-step animation on the sorted array [5, 13, 19, 21, 37, 56, 64, 75, 80, 88, 92] from the textbook (p. 216), illustrating the halving mechanism.
3. **Iterative Pseudocode (15 min):** Formal presentation of the iterative algorithm with the overflow-safe midpoint calculation  $\text{mid} = \text{low} + (\text{high} - \text{low}) / 2$ , directly quoted from the textbook (p. 217).
4. **Recursive Implementation (10 min):** Derivation of the recursive version, highlighting the base case and the recurrence relation.
5. **Complexity Analysis (10 min):** Derivation of  $T(n) = T(n/2) + O(1)$  yielding  $O(\log n)$  time and  $O(\log n)$  space for the recursive variant (p. 219).
6. **Live Coding Session (15 min):** Instructor-led implementation in C, covering edge cases (empty, single-element, duplicate-containing arrays).
7. **Summary and Error Correction (10 min):** Consolidation of key takeaways and a focused discussion on common boundary-update mistakes (e.g.,  $\text{low} = \text{mid}$  instead of  $\text{low} = \text{mid} + 1$ ).

Each module is annotated with corresponding textbook page references and estimated duration, facilitating direct classroom adoption.

### Appendix A.5.2 Assessment Exercises

The dispatcher routes the same shared context to the exercise generator, producing a three-tier problem set that progressively increases in cognitive demand:

- **Basic (comprehension):** Trace the iterative binary search for target 37 on the demo array, reporting  $(\text{low}, \text{mid}, \text{high})$  at each iteration. This task reinforces the procedural understanding of the algorithm.
- **Intermediate (application):** Implement a function `first_occurrence` that returns the index of the first occurrence of a target in a sorted array with duplicates, maintaining  $O(\log n)$  complexity. This extends the standard algorithm to handle non-unique elements.
- **Advanced (synthesis):** Adapt the binary search to a rotated sorted array (LeetCode #33) and justify the modifications to the boundary conditions. This variant requires a deeper comprehension of the invariant properties and tests the learner's ability to generalise the core idea.

All exercises are accompanied by expected outputs and complexity requirements, consistent with the textbook's notation and the complexity analysis delivered in the lesson plan.

### Appendix A.5.3 Visual Presentation Slides

The slide generator produces a 12-slide PowerPoint deck directly derived from the lesson plan outline. [Table A3](#) summarizes the slide structure. Each slide is populated with content that strictly references the shared context pool; for instance, Slide 4 explicitly cites the source array from page 216, Slide 5 reproduces the overflow-safe pseudocode from page 217, and Slide 8 derives the recurrence with the same notation used in the lesson. The `PPTGenerator` service (Algorithm 2) converts the structured outline into formatted slides, ensuring that all visual materials remain pedagogically consistent with the textual outputs.

**Table A3:** Slide deck outline (condensed).

| Slide | Title                   | Key Content                                                               |
|-------|-------------------------|---------------------------------------------------------------------------|
| 1     | Binary Search Algorithm | Title slide; source citation.                                             |
| 2     | Learning Objectives     | Precondition; iterative/recursive; complexity; error correction.          |
| 3     | Motivation              | Dictionary analogy; efficiency vs. linear scan.                           |
| 4     | Precondition            | Sorted array required; example array (p. 216).                            |
| 5     | Iterative Pseudocode    | $mid = low + (high - low)/2$ ; overflow note (p. 217).                    |
| 6     | Recursive Pseudocode    | Recursive version; $O(\log n)$ space.                                     |
| 7     | Step-by-Step Trace      | Target 37; 4 frames with low/mid/high updates.                            |
| 8     | Complexity              | $T(n) = T(n/2) + O(1) \rightarrow O(\log n)$ .                            |
| 9     | Common Pitfalls         | Off-by-one; overflow; unsorted input; infinite loops.                     |
| 10    | Live Demo Outline       | Test cases: normal, not found, empty, single, duplicates.                 |
| 11    | In-class Exercises      | Basic (trace), intermediate (first-occurrence), advanced (rotated array). |
| 12    | Summary & Homework      | Key takeaways; LeetCode #704, #35, #34; source p. 219.                    |

## References

- Zhang Y, Sun K. The construction of talent cultivation mode of the integration between industry and education in higher vocational education from the perspective of “three educations” reform. In: Proceedings of the 2021 International Conference on Education, Humanity and Language, Art (EHLA 2021); 2021 Jun 19–21; Sanya, China.
- Koberstein-Schwarz M, Meisert A. Facilitating preservice biology teacher development through material-based lesson planning. *Educ Sci.* 2023;14(1):42. doi:10.3390/educsci14010042.
- Garay Abad L, Hattie J. The impact of teaching materials on instructional design and teacher development. *Front Educ.* 2025;10:1577721. doi:10.3389/educ.2025.1577721.
- Murtaza M, Cheng CT, Albahlal BM, Muslam MMA, Raza MS. The impact of LLM chatbots on learning outcomes in advanced driver assistance systems education. *Sci Rep.* 2025;15(1):7260. doi:10.1038/s41598-025-91330-3.
- Abu-Rasheed H, Jumbo C, Al Amin R, Weber C, Wiese V, Obermaisser R, et al. LLM-assisted knowledge graph completion for curriculum and domain modelling in personalized higher education recommendations. In: Proceedings of the 2025 IEEE Global Engineering Education Conference (Educon); 2025 Apr 22–25; London, UK. p. 1–5.
- Pitts G, Hridi AP, Narayanan ABL. A survey of LLM-based applications in programming education: balancing automation and human oversight. In: Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+ NLP); 2025 Nov 9; Suzhou, China. p. 255–62.
- Tan K, Yao J, Pang T, Fan C, Song Y. ELF: educational LLM framework of improving and evaluating AI-generated content for classroom teaching. *ACM J Data Inf Qual.* 2025;17:1–23.
- Kharrufa A, Alghamdi S, Aziz A, Bull C. LLMs integration in software engineering team projects: roles, impact, and a pedagogical design space for AI tools in computing education. *ACM Trans Comput Educ.* 2026;26:1–27.
- Kiyak YS, İş-Kara T, Emekli E. Applications and outcomes of large-language-model-generated feedback in undergraduate medical education: a scoping review. *Med Sci Educ.* 2026;36(1):1–19. doi:10.1007/s40670-025-02621-3.
- Chen F, Huang Y, Wang J, Yu C, Meng X. IFQA-LLM: intelligent intention-driven financial question-answering with large language models: F. Chen et al. *J Supercomput.* 2025;81:1245.
- Lai CH, Lin CY. Analysis of learning behaviors and outcomes for students with different knowledge levels: a case study of intelligent tutoring system for coding and learning (ITS-CAL). *Appl Sci.* 2025;15(4):1922. doi:10.3390/app15041922.
- Ateş H. Integrating augmented reality into intelligent tutoring systems to enhance science education outcomes. *Educ Inf Technol.* 2025;30(4):4435–70. doi:10.1007/s10639-024-12970-y.

13. Wang T, Zhan Y, Lian J, Hu Z, Yuan NJ, Zhang Q, et al. LLM-powered multi-agent framework for goal-oriented learning in intelligent tutoring system. In: *Proceedings of the Companion Proceedings of the ACM on Web Conference 2025*; 2025 Apr 28–May 2; Sydney, NSW, Australia. p. 510–9.
14. Almetnawy H, Orabi A, Alneyadi AR, Ahmed T, Lakas A. An adaptive intelligent tutoring system powered by generative AI. In: *Proceedings of the 2025 IEEE Global Engineering Education Conference (Educon)*; 2025 Apr 22–25; London, UK. p. 1–10.
15. Ben Chaabene NEH, Hammami H. Neuro-symbolic synergy in education: a survey of LLM-knowledge graph integration for explainable reasoning and emotion-aware student support. *Smart Learn Environ*. 2026;13:6.
16. Filippi S, Motyl B. Large language models (LLMs) in engineering education: a systematic review and suggestions for practical adoption. *Information*. 2024;15:345.
17. Nguyen SS, Kittur J. Review of current and potential uses of large language models in engineering. *Front Educ*. 2025;10:1649650. doi:10.3389/feduc.2025.1649650.
18. Yi X, Feng W, He Y, Wang F. How to harness LLMs in project-based learning: empirical evidence for individual autonomy and moderate constraints in engineering education. *Systems*. 2025;13:1112.
19. Fu Q, Zhao Y, Jia Z, Zheng Y. Large language models (LLMs) in programming learning: the current research state and agenda. *IEEE Trans Learn Technol*. 2025;18:942–61.
20. Jury B, Lorusso A, Leinonen J, Denny P, Luxton-Reilly A. Evaluating LLM-generated worked examples in an introductory programming course. In: *Proceedings of the 26th Australasian Computing Education Conference*; 2024 Jan 29–Feb 2; Sydney, Australia. p. 77–86.
21. Wang S, Xu T, Li H, Zhang C, Liang J, Tang J, et al. Large language models for education: a survey and outlook. *IEEE Signal Process Mag*. 2026;42:51–63.
22. Chen Y, Chen H, Su S. Fine-tuning large language models in education. In: *Proceedings of the 2023 13th International Conference on Information Technology in Medicine and Education (ITME)*; 2023 Nov 24–26; Wuyishan, China. p. 718–23.
23. Wang L, Chen S, Jiang L, Pan S, Cai R, Yang S, et al. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artif Intell Rev*. 2025;58(8):227. doi:10.1007/s10462-025-11236-4.
24. Liu S, Keung J, Yang Z, Liu F, Zhou Q, Liao Y. Delving into parameter-efficient fine-tuning in code change learning: an empirical study. In: *Proceedings of the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*; 2024 Mar 12–15; Rovaniemi, Finland. p. 465–76.
25. Monteiro MEP, Talau M, Lopes HS. LLM QLoRA fine-tuning of Llama, DeepSeek, and Qwen: a skyrim case study. *IEEE Access*. 2026;14:16548–66. doi:10.1109/access.2026.3659054.
26. Brzęk B, Dzikowski G. Efficient medical question answering through QLoRA fine-tuning and knowledge distillation. *Procedia Comput Sci*. 2025;270:1468–77. doi:10.1016/j.procs.2025.09.268.
27. Shen A, Lai Z, Li D, Hu X. Optimizing fine-tuning in quantized language models: an in-depth analysis of key variables. *Comput Mater Contin*. 2025;82:307.
28. Hao Y, Lai Z, Wang W, Li S, Liu W, Ge K, et al. HSDP: accelerating large-scale model training via efficient sharded data parallelism. In: *Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*; 2024 Oct 30–Nov 2; Kaifeng, China. p. 116–25.
29. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLoRA: efficient finetuning of quantized LLMs. *Adv Neural Inf Process Syst*. 2023;36:10088–115.
30. Yin J, Dong J, Wang Y, De Sa C. ModuLoRA: finetuning 2-bit LLMs on consumer GPUs by integrating with modular quantizers. *Trans Mach Learn Res*. 2024;2024:1596.
31. Yin C, Zhang Z. A study of sentence similarity based on the all-minilm-l6-v2 model with “same semantics, different structure” after fine tuning. In: *Proceedings of the International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*; 2024 Aug 9–11; Singapore. p. 677–84.