Received: 10 Jul. 2026 | Revised: 20 Jul. 2026 | Accepted: 23 Jul. 2026 | Published online: 15 Aug. 2026

RESEARCH ARTICLE

Contemporary Education and Teaching Research
AI Reshaping the Classroom: Transformation of Teaching and Learning Models
2026, Vol 1, 37-49
DOI: 10.61360/BoniCETRAIRC262020430104

AI-Assisted Task-Driven Reform of a Computer Vision Course: A Hierarchical Practice Framework for Programming Competence

Shanshan Huang[1,*], Hongyue Huang[1] & Hanyuan Wang[1]
[1]*Yunnan University, P.R. China*

Abstract: The rapid development of artificial intelligence (AI) is reshaping not only educational content but also the organization of classroom practice, learning support, and assessment. Computer vision courses are typical of this transformation: they require students to understand complex models while also completing data processing, programming, training, debugging, and evaluation tasks. In response to common problems such as fragmented experiments, insufficient programming support, and unregulated use of generative AI tools, this paper proposes an AI-assisted, task-driven reform framework for a computer vision course. The framework uses virtual try-on with traditional ethnic costume imagery as a culturally situated mainline task and organizes practice into three progressive levels: code reproduction, model modification, and independent design. It also introduces traceable AI use, structured experimental reporting, and multi-source assessment. The reform provides a practical model for integrating human-AI collaboration into specialized AI courses while preserving students' independent reasoning, programming competence, and experimental accountability.

Keywords: Artificial intelligence in education, computer vision course, task-driven learning, hierarchical practice, programming competence

1. Introduction

Artificial intelligence (AI) has moved from being a specialized research topic to becoming a broad educational condition that affects what is taught, how students learn, and how teachers organize feedback. In higher education, this shift is especially visible in courses related to computer science and artificial intelligence, where students must learn both conceptual knowledge and the practical ability to build working systems. The classroom is no longer limited to teacher explanation and student imitation. It increasingly includes interactive coding environments, intelligent programming assistants, automated feedback, and learning analytics. This creates a new question for curriculum reform: how can AI technologies be integrated into course practice without weakening students' independent judgment and technical responsibility?

Computer vision is a representative case (Cun & Sha, 2023). As a core course for AI, computer science, and related majors, it covers image classification, object detection, semantic segmentation, image generation, image editing, and vision-language understanding. These topics are closely connected in real AI applications (Zhang, 2025; Arnald, 2025; Holmes & Miao, 2023), yet they are often presented as separate teaching units. As a result, students may be able to recall model names, basic principles, and evaluation metrics, but still lack a coherent understanding of how different techniques are connected within an end-to-end visual task. More importantly, they often struggle to complete the full practical workflow, including data collection, data preprocessing, model construction, model training, experimental evaluation, error analysis, and iterative improvement. With the rapid development of deep learning frameworks and large-scale visual models, the gap between theoretical understanding and programming competence has become increasingly prominent.

The emergence of generative AI and large language models makes this problem more urgent (Jiang et al., 2025). On the one hand, AI tools such as code generation, conversational debugging, and error interpretation can lower the threshold for programming practice. They can help students understand unfamiliar functions, locate bugs, compare implementation alternatives, and improve experimental efficiency. On the other hand, excessive or unregulated reliance on AI may lead to superficial task completion. Students may copy generated code without understanding its logic, fail to verify the correctness of AI-generated solutions, or avoid the necessary process of debugging and reflection. Therefore, AI should not be treated only as a shortcut for finishing programming tasks. Instead, it should be incorporated into a pedagogical framework that emphasizes scaffolding, inquiry, verification, reflection, and accountable human-AI collaboration (Liao et al., 2024; Jin et al., 2024).

Corresponding Author: Shanshan Huang
Yunnan University, P.R. China

Table 1. Comparison of Conventional and Reformed Teaching Approaches

Dimension	Conventional Mode	Reformed Mode	Academic Implication
Course organization	Knowledge-point driven; linear delivery of isolated units.	Task-driven PBL centered on virtual try-on and image editing.	From fragmented knowledge to systematic competence.
Experimental logic	Verification-oriented; single tasks with weak connections.	Iterative and systematic; end-to-end project development.	Simulates a real AI development pipeline.
Programming requirement	API calls, script execution, and black-box testing.	Engineering implementation, model modification, profiling, and optimization.	Strengthens computational thinking and model understanding.
AI collaboration	Defensive view of AI as a potential cheating risk.	Constructive use of AI as a coding scaffold with prompt records and verification.	Cultivates responsible human-AI synergy.
Assessment system	Result-oriented; based mainly on final exams or outputs.	Process-oriented; based on code quality, Git records, reflection, and failure analysis.	Introduces code review and process evidence

To address these issues, this paper proposes an AI-assisted task-driven reform of a computer vision course. The redesigned course uses virtual try-on and image editing based on traditional ethnic costume imagery as a unified mainline task. This task naturally connects multiple computer vision techniques, including classification, detection, segmentation, generation, image editing, multimodal understanding, and visual evaluation. By organizing dispersed knowledge points around a continuous application scenario, the course helps students understand the relationship among different techniques and complete an end-to-end visual computing workflow. The reform further introduces a hierarchical programming practice framework consisting of three levels: code reproduction, model modification, and independent model design. Students first reproduce baseline models to understand standard experimental procedures, then modify model components or training strategies to conduct comparative analysis, and finally design their own solutions for open-ended tasks. In parallel, students are required to complete full-process project practice, covering dataset construction, data processing, model training, testing, result analysis, and improvement. AI tools are allowed but regulated through explicit usage boundaries, learning records, code verification, and reflective reports.

The main contribution of this paper is an operational teaching reform framework for computer vision education. This framework integrates a unified task mainline, hierarchical programming training, full-process project practice, and regulated AI assistance into one course design. Rather than proposing a new computer vision algorithm, this paper provides a practical model for improving students' programming competence, systematic understanding, and responsible use of AI in specialized technical courses.

2. Main Content

2.1. Instructional challenges and reform motivation

Through teaching practice, student feedback, and a review of students' project performance, we identify three key challenges in current computer vision education: fragmented knowledge structure, insufficient programming and project practice, and unregulated AI-assisted learning. These challenges reveal a mismatch between traditional course organization and the practical requirements of AI system development, thereby motivating the need for an integrated instructional design that connects task-driven learning, hierarchical programming practice, full-process project training, and regulated AI assistance.

First, computer vision courses often face the problem of fragmented knowledge structure. Traditional instruction usually follows a chapter-based sequence, covering topics such as image preprocessing, image classification, object detection, semantic segmentation, image generation, and multimodal learning. Although this arrangement is consistent with the logical structure of textbook knowledge, it may lead students to understand these topics as separate techniques rather than as interconnected components of a complete visual intelligence system. In practical AI applications, however, different computer vision tasks are usually combined to solve complex problems. For example, a virtual try-on system may involve clothing recognition, human parsing, garment-region segmentation, pose and structure preservation, conditional image generation, and visual quality evaluation. Without a shared application context, students may find it difficult to understand why different visual modules need to cooperate and how they jointly contribute to an end-to-end system.

Second, programming and full-process project practice remain insufficient in many computer vision courses. Many students are able to explain the basic concepts of convolution, attention mechanisms, or diffusion models, but still encounter difficulties when modifying a dataset class, tracing tensor shape errors, adjusting a loss function, interpreting abnormal training curves, or improving an experimental pipeline. This gap is not merely caused by insufficient coding experience; but also closely related to the organization of practical teaching. When experiments are designed mainly as short

verification tasks, students may simply run prepared scripts and obtain expected outputs, without developing a deep understanding of the relationship between model principles, implementation details.

This challenge is further complicated by the uneven programming foundations of students. In the same computer vision classroom, some students may already be familiar with Python, PyTorch, Git, and deep learning project structures, whereas others may still have difficulty reading and modifying existing code. A uniform experimental task may therefore be too difficult for some students and too shallow for others. Moreover, programming competence in computer vision should not be reduced to model implementation alone. In actual projects, data preparation and data management are crucial to whether a model can be trained, evaluated, and improved effectively. Students need systematic experience in checking image quality, cleaning labels, designing augmentation strategies, splitting datasets, implementing Dataset and DataLoader components, aligning image-text inputs, and analyzing failure cases. However, these abilities are often obscured when courses rely heavily on ready-made datasets, fixed experimental settings, and prepared scripts.

Third, the widespread use of generative AI tools introduces a new challenge of AI-assisted learning governance. Large language models and intelligent programming assistants can provide timely support for code explanation, error diagnosis, function implementation, and experimental comparison. They can lower the threshold for programming practice and help students overcome technical obstacles during project development. However, their outputs are probabilistic and context-dependent. They may generate plausible but incorrect code, ignore specific course requirements, or produce unsupported statements in experimental reports. If students copy AI-generated results without understanding, verification, or reflection, AI may weaken rather than strengthen their programming competence.

The above diagnosis suggests that the reform of a computer vision course cannot be limited to adding more experiments or permitting students to use AI tools. A coherent solution must connect content integration, differentiated programming practice, data engineering, AI-use regulation, and evidence-based assessment. On this basis, the next section proposes an AI-assisted task-driven hierarchical practice framework in which each component is designed to respond to a specific instructional challenge.

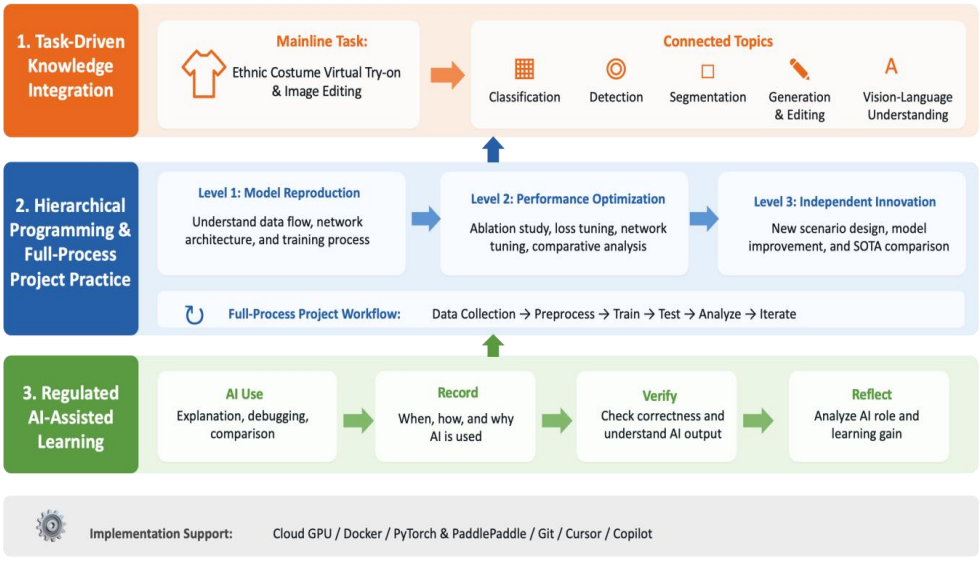


Figure 1. Overall design of the AI-assisted task-driven hierarchical practice framework

2.2. Theoretical foundations and design rationale

To address the instructional challenges identified above, the theoretical foundations of the framework are organized around four interrelated design questions: how to connect fragmented course content, how to arrange progressive practice for students with different levels of prior experience, how to regulate the use of generative AI, and how to evaluate student learning comprehensively.

First, task-centered instruction and constructivist learning provide the basis for addressing the fragmentation of computer vision knowledge (Merrill, 2007; Rosenberg-Kima et al., 2022). Task-centered instruction emphasizes organizing learning around authentic whole tasks, while constructivist learning highlights students' active construction of knowledge through situated practice and reflection. Accordingly, ethnic-costume virtual try-on is adopted as the course-wide task, and image classification, segmentation, detection, generation, and multimodal understanding are organized as successive components required to complete this task. This organization establishes explicit connections among otherwise separate knowledge modules.

Second, differentiated instruction, Bloom's taxonomy, and scaffolding theory jointly inform the practical progression designed for students with different programming foundations and practical

abilities (Tomlinson, 2014; Anderson & Krathwohl, 2001). Differentiated instruction determines how learning support is adapted to student readiness, Bloom's taxonomy defines the progression from lower- to higher-order cognitive objectives, and scaffolding theory guides the gradual withdrawal of instructor support. Based on these principles, the three-level sequence of baseline reproduction, component modification, and independent design corresponds respectively to understanding and application, analysis and evaluation, and creation and validation. Across the three levels, cognitive demand, task openness, and learner autonomy gradually increase, whereas instructor support decreases.

Third, responsible AI education provides the basis for regulating the use of generative AI by emphasizing human agency, critical judgement, and transparency (Miao & Holmes, 2023; Kasneci et al., 2023). Generative AI is positioned as a scaffold for conceptual understanding, programming, and debugging rather than as a substitute for student work. A traceable"prompt–evaluation–revision–verification" process therefore requires students to examine, modify, and experimentally validate AI-generated suggestions while distinguishing AI-generated content from their own contributions.

Finally, formative assessment and constructive alignment provide the basis for evaluating whether the intended learning outcomes are achieved (Black & Wiliam, 1998; Biggs, 1996). The assessment considers not only final code and model performance but also conceptual understanding, programming implementation, experimental analysis, problem solving, and responsible AI use. This ensures alignment among the intended learning outcomes, instructional activities, and assessment evidence. The corresponding practical activities, AI-use rules, and assessment procedures are presented in the following sections.

2.3. An AI-assisted task-driven hierarchical practice framework

2.3.1 Overall design

To address the three teaching challenges identified in Section 2.1, this study proposes an AI-assisted task-driven hierarchical practice framework for computer vision education, as shown in Figure 1. The framework consists of three closely related components: task-driven knowledge integration, hierarchical programming and full-process project practice, and regulated AI-assisted learning. First, the course introduces ethnic costume virtual try-on and image editing as a unified mainline task to connect key computer vision topics, including classification, detection, segmentation, generation, image editing, and vision-language understanding, thereby reducing the fragmentation of knowledge structure. Second, the course designs a progressive practice path from model reproduction to model modification and then to independent model design, while requiring students to complete the full project workflow of data collection, preprocessing, model training, testing, analysis, and iterative improvement, thereby strengthening programming competence and project implementation ability. Third, the course establishes explicit norms for AI-assisted learning, requiring students to record, explain, verify, and reflect on their use of generative AI tools, so that AI functions as a learning scaffold rather than a substitute for students' own reasoning and practice. Through these three components, the framework forms a direct correspondence between teaching challenges and reform measures, providing an integrated instructional design for improving students' systematic understanding, practical competence, and responsible use of AI in computer vision learning.

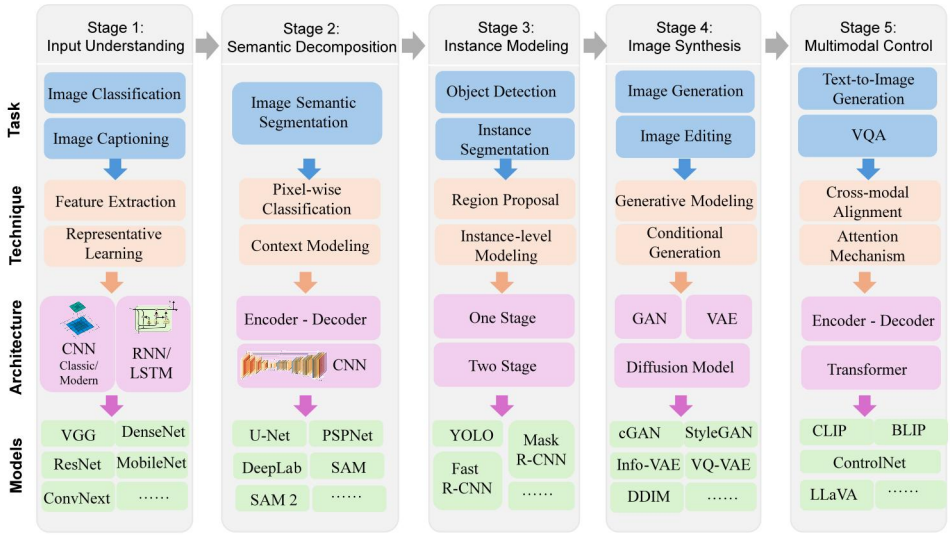


Figure 2. Comprehensive Knowledge Mapping and Experimental Organization driven by the Virtual Try-on Mainline

2.3.2 Task-driven knowledge integration

To integrate fragmented computer vision topics into a coherent learning trajectory, the course adopts virtual try-on with traditional ethnic costume imagery as the unified mainline task. The reason for choosing this task is that virtual try-on is not merely a

generative application, but a complete computer vision pipeline that involves image understanding, semantic decomposition, instance-level modeling, visual synthesis, and multimodal interaction. In a practical virtual try-on scenario, the system must first understand the input image and identify the attributes of the person and costume; it then needs to locate editable regions, distinguish different garment components and accessories, synthesize or edit the target visual content, and finally respond to user instructions expressed through text, reference images, or multimodal conditions. Therefore, this task provides a natural pedagogical carrier for connecting classification, captioning, segmentation, detection, generation, editing, and multimodal learning within one application-oriented framework.

As illustrated in Figure 2, the proposed task thread is organized into five progressive stages that guide students from visual perception to controllable generation. *The first stage focuses on input understanding*, where students use image classification and image captioning to identify the category, color, pattern, style, and semantic content of ethnic costume images, with representative models such as VGG, ResNet, DenseNet, MobileNet, ConvNeXt, and recurrent captioning models. *The second stage extends this process to semantic decomposition.* Since virtual try-on requires precise control over editable and preserved regions, students learn to segment images into the person, garments, accessories, and background through methods such as U-Net, PSPNet, DeepLab, SAM, and SAM 2, thereby establishing the spatial basis for controllable editing. *The third stage moves toward instance modeling,* where object detection and instance segmentation methods, including YOLO, Faster R-CNN, and Mask R-CNN, are introduced to distinguish individual garments, ornaments, and overlapping regions and to prepare structured inputs for later try-on or editing tasks. *The fourth stage introduces image synthesis*, in which students learn how GANs, VAEs, StyleGAN, VQ-VAE, DDIM, and diffusion models generate or edit costume images while preserving identity, pose, background, and non-target details. *The fifth stage introduces multimodal control*, where natural-language descriptions, reference images, and visual questions are connected with visual generation through cross-modal alignment, attention mechanisms, Transformers, and models such as CLIP, BLIP, ControlNet, and LLaVA. Through these stages, students understand how perception, decomposition, structure modeling, synthesis, and multimodal control jointly support an end-to-end virtual try-on and image editing system.

This unified mainline-task design strengthens the teaching reform from three perspectives. At the knowledge level, it embeds recognition, segmentation, detection, generation, and multimodal control into a continuous virtual try-on pipeline, thereby promoting systematic understanding rather than fragmented knowledge acquisition. At the learning-process level, it follows a progressive trajectory from input understanding and semantic decomposition to instance modeling, image synthesis, and multimodal control, enabling students to gradually develop from basic visual perception to controllable generative reasoning. At the educational-value level, the use of traditional ethnic costume imagery connects technical training with data authorization, privacy protection, cultural respect, and responsible AI practice. In this way, the virtual try-on project thread serves as both a technical integration mechanism and a pedagogical carrier for cultivating system thinking, engineering practice, and responsible innovation in computer vision education.

2.3.3 Hierarchical programming and full-process project practice

To support progressive competence development, we constructed a three-level practice system comprising baseline reproduction, component modification, and independent design. The progression was determined before course implementation by jointly considering the pre-course distribution of self-reported programming proficiency, recurring difficulties observed in the pre-reform cohort, and the principles of Bloom's taxonomy and progressive scaffolding. Accordingly, task openness, technical complexity, and student autonomy increase across the three levels. The sequence is therefore treated as a theory- and data-informed progression rather than an empirically optimized difficulty gradient. Across all three levels, students engage in the full project workflow, including data preparation, model implementation, training, evaluation, result analysis, and iterative improvement.

At Level 1, the practice focuses on "code execution." Students are required to reproduce a baseline model using the dataset, experimental environment, dependency configuration, baseline code, and running instructions provided by the instructor. The purpose of this level is not merely to ensure that students can successfully run the model, but to help them understand the basic operational logic of a computer vision project. Through this process, students become familiar with the project structure, data-loading pipeline, model input and output, training procedure, testing process, and evaluation results, thereby establishing a common technical foundation for subsequent model modification and independent design.

At Level 2, the practice focus shifts from "running the model" to "modifying the model." After students understand the baseline workflow, they are required to make targeted and explainable modifications to selected components of the model. These modifications may involve changing the data preprocessing strategy, adjusting data augmentation operations, replacing or adding network modules, modifying loss-function weights, tuning hyperparameters, or improving visualization methods. The key requirement at this level is that the modification should not be arbitrary; students need to explain why a certain component is modified, what

effect the modification is expected to produce, and whether the experimental results support the expectation. Through before-and-after comparison and simple ablation analysis, students learn to connect code-level changes with model-level behavior and performance-level outcomes. This level therefore strengthens their model understanding, debugging ability, and evidence-based analysis.

At Level 3, the practice further develops into "independent design." Students are encouraged to identify a specific problem within the unified mainline task, such as inaccurate garment-region localization, insufficient detail preservation, unstable editing results, or weak text-image consistency. Based on the identified problem, they need to formulate an improvement idea, design the experimental procedure, implement the core module, and validate the effectiveness of the proposed solution through controlled experiments. Compared with Level 1 and Level 2, this level places greater emphasis on task openness and student autonomy. Students are expected not only to obtain results, but also to justify their design choices through quantitative metrics, visual comparison, and failure-case analysis. In this process, they gradually develop the ability to define problems, design solutions, organize experiments, and construct evidence-based arguments. Through this hierarchical design, programming practice is transformed from simple code execution into a progressive learning process. Students gradually develop the ability to understand project workflows, modify and analyze models, design solutions, and construct evidence-based experimental arguments.

Table 2. Three-Level Practice System and Quantitative Scoring Scheme

Practice level	Core requirement	Assessment evidence	Score allocation by dimension
Level 1: Baseline Reproduction (25 points)	Run and explain the baseline project workflow	Running logs, training curves, test and visualization results, and AI-use records	SU: 8; PC: 10; PA: 4; EA: 2; RAI: 1
Level 2: Component Modification (35 points)	Modify selected components and compare performance changes	Modification notes, key code changes, comparative or ablation results, and AI-verification records	SU: 7; PC: 9; PA: 10; EA: 6; RAI: 3
Level 3: Independent Design (40 points)	Propose, implement, and validate an independent improvement	Project proposal, code repository, experimental report, failure-case analysis, and reflection on AI use	SU: 5; PC: 6; PA: 11; EA: 12; RAI: 6
Total (100 points)	-	-	SU: 20; PC: 25; PA: 25; EA: 20; RAI: 10

**Note: SU = systematic understanding; PC = programming competence; PA = full-process project ability; EA = experimental analysis ability; RAI = responsible AI use.*

2.3.4 Traceable AI-assisted scaffolding for regulated human-AI collaboration

With the rapid development of generative AI tools, students can obtain immediate support for code explanation, error diagnosis, script generation, and technical writing. Tools such as ChatGPT, GitHub Copilot, and Cursor can reduce the difficulty of computer vision practice, especially when students encounter unfamiliar APIs, dependency conflicts, tensor-shape errors, or complex training procedures. However, without clear regulation, AI use may obscure students' actual participation and weaken the reliability of process-based assessment. Therefore, this course positions AI tools as traceable learning scaffolds rather than unrestricted productivity tools.

The proposed mechanism is organized around four requirements: guidance, boundaries, documentation, and verification. AI tools may be used differently across the three practice levels. In code execution, they support error interpretation, API explanation, code understanding, and environment checking. In model modification, they assist with tensor-shape inspection, debugging strategy design, code refactoring, and comparison of implementation alternatives. In independent design, they help students review experimental plans, identify missing evidence, organize failure-case analysis, and improve technical reports. In this way, AI assistance is embedded in specific learning tasks and aligned with the course's hierarchical practice structure.

To clarify acceptable and unacceptable use, the course defines explicit AI-use boundaries. Students may use AI for explanation, debugging, comparison, and reflection, but they may not directly submit AI-generated reports, copy AI-generated core code without understanding, fabricate logs or evaluation results, or replace experimental analysis with generated text. For model modification and independent design tasks, students are required to explain the motivation, modified components, expected effects, and supporting evidence of their decisions. AI-generated suggestions can serve as references, but not as final justifications.

To make AI use visible, each experimental report includes an AI-Assisted Learning Record, documenting the tool used, the submitted question, the received suggestion, the manual revision, the adoption decision, and the verification method. Important AI-assisted code changes are also marked through README notes, code comments, or commit messages. The adopted suggestions must be checked through execution results, training logs, visual

outputs, controlled comparisons, ablation analysis, or evaluation metrics. This requirement is particularly important in computer vision practice, where syntactically correct code may still cause data-loading errors, modality mismatch, invalid metric computation, or misleading visualization.

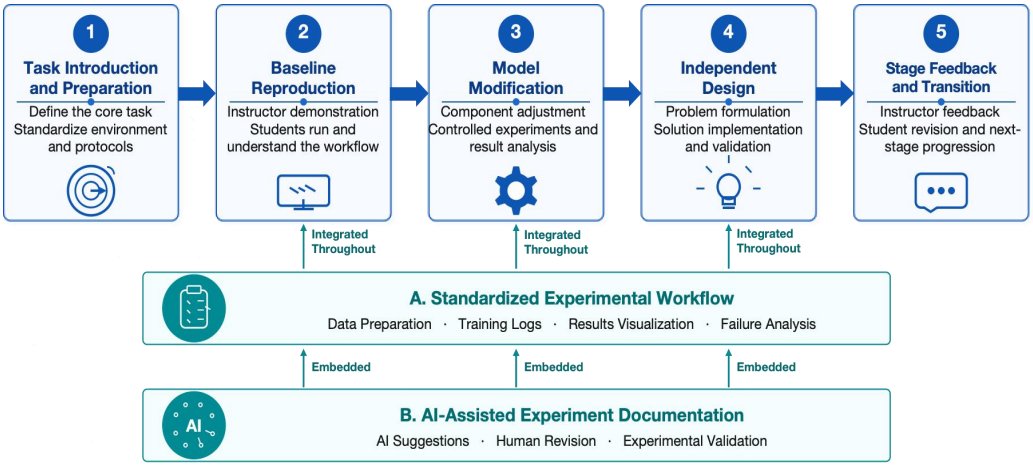


Figure 3. Classroom implementation path of the proposed reform

Through this mechanism, AI-supported learning becomes observable, assessable, and accountable. Students can benefit from AI-assisted explanation and debugging while developing the habit of questioning, revising, and validating generated suggestions. Thus, AI tools are integrated into programming practice as supervised scaffolds for responsible human-AI collaboration.

3. Classroom Implementation Path

After constructing the teaching reform framework, we further translate the proposed design into a concrete classroom implementation path. Different from Section 3, which focuses on the structure and rationale of the reform framework, this section explains how the framework is implemented in classroom teaching and programming practice. The implementation follows three steps: task introduction and basic preparation, staged practice implementation, and stage feedback with task transition. In this process, we adopt an instructional organization of "teacher demonstration–student practice–stage submission–feedback-based revision," so that the unified mainline task, hierarchical practice system, standardized experimental workflow, and regulated AI-use mechanism can be integrated into a continuous teaching process. The implementation path is shown in Figure 3.

3.1 Task introduction and basic preparation

At the beginning of implementation, we introduce ethnic costume virtual try-on and image editing as the mainline task, helping students understand that course practice is organized around an integrated computer vision application rather than isolated experiments. Based on the task pipeline, key technical modules are explained according to their functional roles: recognition identifies garment attributes; segmentation and human parsing distinguish editable and protected regions; detection and instance modeling locate garment components and accessories; generation and editing models synthesize try-on results and modify visual attributes; and multimodal control supports interaction through text prompts or reference images. This introduction helps students form an overall understanding of how different computer vision techniques cooperate in a complete application scenario.

We then provide unified experimental materials and requirements, including dataset descriptions, task instructions, baseline code, environment configuration documents, report templates, code submission rules, and AI-use guidelines. To reduce the influence of environment inconsistency, the experimental environment is standardized through cloud GPU platforms or preconfigured dependency files. At the same time, students are informed of the required process evidence, such as training logs, visualized results, failure-case analysis, and AI-assisted learning records. This stage therefore provides both technical preparation and procedural standards for subsequent practice.

3.2 Staged practice implementation

After preparation, the practice proceeds through three stages: baseline reproduction, model modification, and independent design. This sequence gradually shifts students from instructor-guided implementation to student-led exploration, with increasing task complexity and autonomy.

In the baseline reproduction stage, students first reproduce a representative model using the provided dataset, environment, code, and running instructions. This stage serves as the entry point for understanding the complete workflow of a computer vision project. Rather than focusing only on whether the code can run successfully, students are guided to trace how data are loaded, how model inputs and outputs are organized, how training and testing scripts are executed, and how evaluation results are generated.

They then submit running logs, training curves, testing results, visual examples, and a brief workflow explanation. These materials help ensure that students have established a shared technical foundation for subsequent model modification.

In the model modification stage, students move from baseline reproduction to targeted component adjustment. Based on problems observed in the initial experiments, they may modify data augmentation, network modules, loss-function, hyperparameters, visualization methods, or evaluation settings. Each modification must be supported by a clear motivation, an expected effect, and a controlled comparison. Through the analysis of training behavior, evaluation metrics, and visual results, students learn to relate implementation decisions to model behavior and performance variation.

In the independent design stage, students move beyond prescribed modification tasks and identify a specific problem within the mainline task. The problem is usually derived from earlier experimental results or failure cases, such as inaccurate garment localization, insufficient mask quality, loss of generated details, weak background consistency, unstable text-image correspondence, or limited multimodal control. Based on this diagnosis, students propose an improvement scheme and submit a short proposal covering the problem source, design idea, implementation plan, and evaluation method. After receiving feedback, they revise the proposal and complete implementation, comparison experiments, result analysis, and report writing. This stage therefore guides students from task completion toward independent problem definition, solution design, and evidence-based validation.

3.3 Stage feedback and task transition

To maintain continuity across the three practice stages, targeted feedback is provided at each transition point. After baseline reproduction, feedback focuses on environment configuration, code execution, data loading, and workflow understanding, so that students can resolve implementation-level problems before entering model modification. During model modification, feedback shifts to the clarity of modification motivation, the validity of comparison settings, the interpretation of result differences, and the analysis of failure cases. Before independent design, feedback further examines whether the problem definition is specific, whether the proposed scheme is feasible, whether the experimental scope is controllable, and whether the evaluation method can support the conclusion. In this way, feedback serves as a transition mechanism that guides students from basic execution to analytical modification and then to independent exploration.

Table 3. Evaluation Dimensions and Evidence for Course Reform

Dimension	Main focus	Evidence
Systematic understanding	Understanding connections among CV modules	Workflow explanation, project report
Programming competence	Reading, running, modifying, and debugging code	Code repository, debugging records
Full-process project ability	Completing data, model, training, testing, and iteration	Data records, logs, results
Experimental analysis ability	Comparing methods and explaining results	Metrics, visual comparison, failure cases
Responsible AI use	Recording, explaining, verifying, and reflecting on AI use	AI-assisted learning record

Specifically, AI tools are embedded into this feedback and transition process as traceable learning support. When students encounter implementation errors, comparison design problems, or difficulties in result interpretation, they may use tools such as ChatGPT, GitHub Copilot, and Cursor to interpret error messages, query APIs, check tensor dimensions, organize debugging strategies, or refine report expression. However, AI-assisted suggestions must be documented together with students' manual revisions and verification methods. This ensures that AI support contributes to problem diagnosis and solution refinement, while students remain responsible for code correctness, experimental validity, and result interpretation. Overall, through this arrangement, classroom practice is transformed from fragmented programming exercises into a progressive learning process organized around a unified mainline task.

4. Evaluation and Quality Assurance

4.1 Evaluation dimensions

The evaluation design is aligned with the objectives of the proposed reform framework, but it focuses on observable learning evidence rather than repeating the framework itself. As summarized in Table 3, the course assesses students' learning outcomes across five dimensions: systematic understanding, programming competence, full-process project ability, experimental analysis ability, and responsible AI use. These dimensions correspond to students' ability to understand the mainline task, implement and modify project code, complete the data-to-model workflow, analyze experimental results, and use AI tools responsibly.

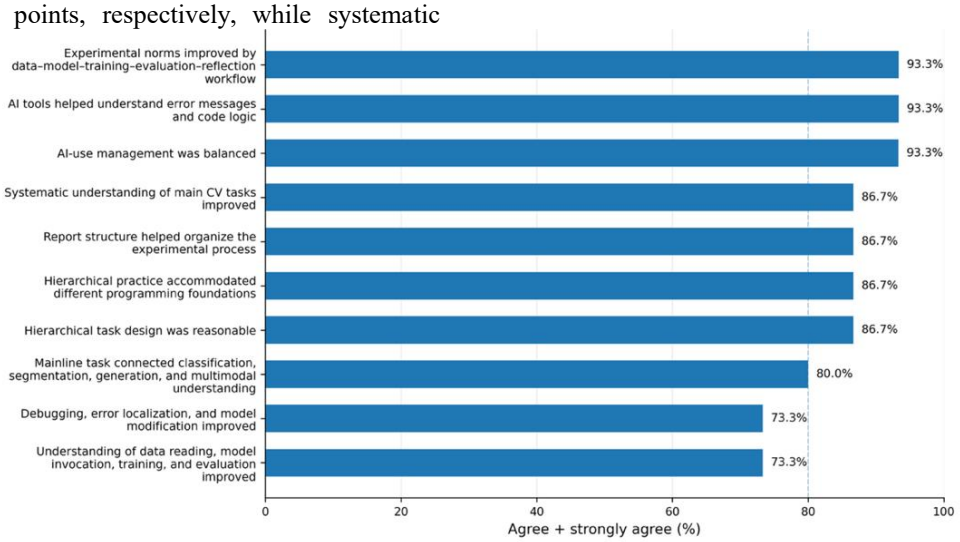
4.2 Evaluation methods

To ensure that assessment reflects both the learning process and final outcomes, the course

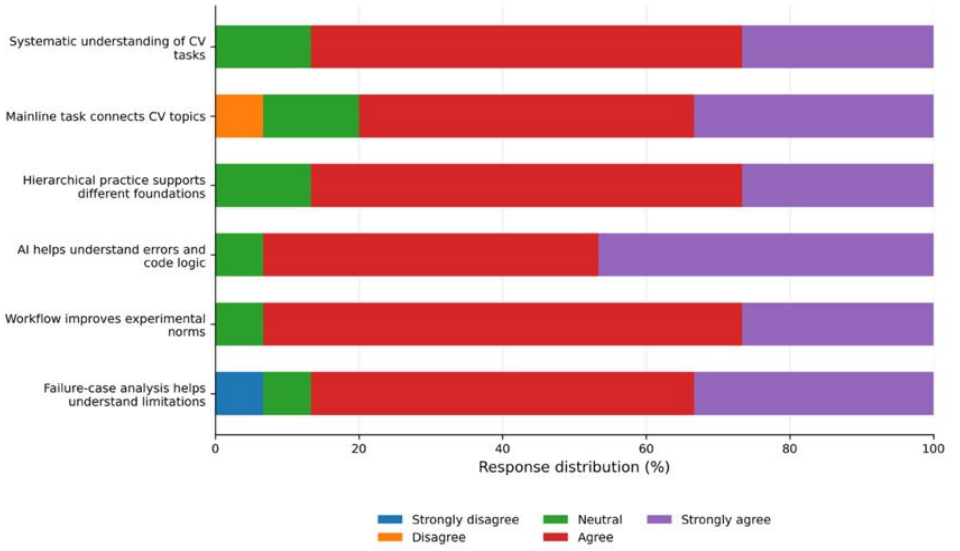
adopts an evidence-based evaluation method. Students are required to submit code repositories, training logs, evaluation metrics, visualization results, failure-case analysis, experimental reports, and AI-assisted learning records. These materials allow instructors to examine not only whether students obtain acceptable results, but also how they complete data processing, model implementation, training, testing, result interpretation, and iterative improvement.

The assessment is organized according to the three-level practice structure. Baseline reproduction is evaluated through successful execution, workflow understanding, and basic result analysis. Model modification is evaluated through modification motivation, implementation correctness, controlled comparison, and interpretation of performance changes. Independent design is evaluated through problem definition, solution design, experimental validation, and evidence-based argumentation. As specified in Table 2, the three levels account for 25, 35, and 40 points, respectively, while systematic understanding, programming competence, full-process project ability, experimental analysis, and responsible AI use account for 20, 25, 25, 20, and 10 points. Each level-dimension criterion is rated from 0 to 4 (4 = fully achieved, 3 = largely achieved, 2 = partially achieved, 1 = minimally achieved, and 0 = incomplete or unverifiable), and its score is calculated as the maximum item score multiplied by the rating and divided by four.

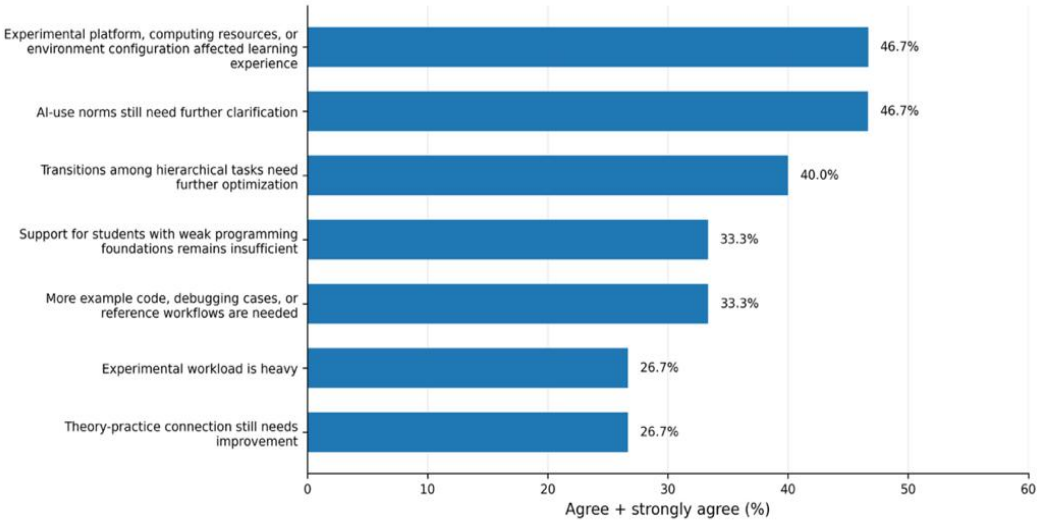
In addition, AI-assisted learning is assessed through usage records, manual revisions, verification methods, and reflective explanations. This design helps ensure that the evaluation reflects students' actual participation and practical competence rather than only final outputs. Besides, Questionnaire responses are used as supplementary evidence of students' perceptions. For dimensions measured by multiple items, internal consistency was assessed using Cronbach's alpha. Given the limited sample size, exploratory and confirmatory factor analyses were not conducted.



(a) Positive Student Responses on Course Reform Effectiveness



(b) Distribution of Responses to Key Reform-Effectiveness Items



(c) Main Improvement Needs Reported by Students

Figure 4. Student Questionnaire Feedback on Course Reform Effectiveness

5. Learning Outcomes and Reform Effectiveness

The effectiveness of the course reform was assessed primarily through course assignments, project reports, code submissions, training records, and visualized results, while questionnaire feedback served as supplementary evidence of students' perceptions. The five principal questionnaire dimensions demonstrated acceptable to high internal consistency (Cronbach's $\alpha = 0.797–0.938$), and the improvement-needs dimension also showed acceptable reliability ($\alpha = 0.780$). A separate pre-course survey indicated heterogeneous initial programming proficiency: 23.08% of students rated themselves as below average, 69.23% as average, and 7.70% as above average. This baseline distribution informed the differentiated support incorporated into the three-level practice framework.

First, students' assignments show that the unified mainline task helped improve the continuity of learning. In previous verification-oriented experiments, students tended to complete isolated tasks separately. In contrast, the virtual try-on task required students to repeatedly connect different computer vision modules, such as classification, segmentation, detection, generation, editing, and multimodal control. In their submitted reports, many students were able to explain the role of each module in the overall task pipeline and describe how data, model inputs, masks, prompts, and generated results were related. The questionnaire results further support this observation: 86.7% of students agreed or strongly agreed that the course helped them understand the main tasks of computer vision more systematically, and 80.0% agreed or strongly agreed that the mainline task could effectively connect classification, segmentation, generation, and multimodal understanding.

To provide the available comparative evidence, we retrospectively compared the project completion rate and practical project score of the reformed cohort with those of the cohort taught before the reform. The two cohorts completed the same course and were evaluated using comparable project requirements. However, because the previous cohort constitutes a non-equivalent historical comparison group, the comparison cannot fully control for differences between academic years and is therefore interpreted as preliminary rather than causal evidence.

Table 4. Comparison of Practical Outcomes Between the Pre-Reform and Reformed Cohorts

Indicator	Pre-reform cohort	Reformed cohort	Statistical test	Effect size
Completion rate	34/45 (75.6%)	45/48 (93.8%)	Fisher's exact test, p= 0.020	Cramér's V= 0.254
Project score	72.6 ± 10.1	84.1 ± 8.4	Welch's t(85.79)=5.95, p< 0.001	Cohen's d= 1.242

As shown in Table 4, the project completion rate increased from 75.6% in the pre-reform cohort to 93.8% in the reformed cohort, representing an increase of 18.2 percentage points. Fisher's exact test indicated that the difference was statistically significant ($p = 0.020$), with a small-to-moderate effect size (Cramér's $V = 0.254$). The mean practical project score also increased from 72.6 ± 10.1 to 84.1 ± 8.4. Welch's t-test showed a statistically significant difference, $t(85.79) = 5.95$, $p < 0.001$, with a large effect size (Cohen's $d = 1.242$). These results provide preliminary evidence that implementation of the reformed framework was associated with higher project completion and practical performance. However, because the comparison involved two cohorts from different teaching cycles, the observed differences cannot be attributed solely to the reform.

Second, the hierarchical practice structure promoted students' transition from code execution to code understanding and model modification. To examine whether students with different initial programming foundations could progress to the independent-design stage, an additional complete-case analysis was conducted. Based on the pre-course self-assessment, students were classified into a below-average group and an average-or-above group. The above-average categories were combined with the average category because of their relatively small proportions. The Level 3 completion rates were 81.8% and 94.6%, respectively, and the difference was not statistically significant (Fisher's exact test, $p = 0.221$, Cramér's $V = 0.194$). However, the average-or-above group achieved a significantly higher median standardized Level 3 score than the below-average group (84.0 vs. 76.0; Mann–Whitney U test, $p = 0.003$), with a relatively large effect size (rank-biserial $r = 0.590$). These results suggest that most students with weaker programming foundations were able to complete the independent-design stage, although their performance quality remained associated with their initial programming proficiency.

Although students with stronger initial programming foundations achieved higher Level 3 scores, the completion rate of the below-average group remained above 80%. This finding provides preliminary evidence that students with weaker foundations were generally able to enter and complete the independent-design stage with the support of the progressive practice framework.

In the baseline reproduction stage, students became familiar with project structure, data loading, model inputs and outputs, training scripts, testing procedures, and evaluation results. In the model modification stage, their assignments began to include more targeted changes, such as adjusting data preprocessing, modifying augmentation strategies, tuning training parameters, improving visualization, or comparing different model settings. These changes indicate that students were no longer limited to obtaining runnable results, but began to connect implementation decisions with model behavior and performance variation. The questionnaire results show that 73.33% of students agreed or strongly agreed that they could better understand the relationship among data reading, model invocation, training, and evaluation; 73.33% reported improvement in debugging, error localization, and model modification; and 66.67% believed that the experimental tasks helped them move from "running code" to "understanding and modifying code."

Third, students' full-process project awareness was strengthened through the course assignments. The submitted materials required not only final outputs, but also training logs, evaluation metrics, visualized results, failure-case analysis, and workflow explanations. This encouraged students to treat computer vision practice as a complete project process rather than as isolated model implementation. In several assignments, students analyzed how data quality, mask accuracy, training settings, and evaluation methods affected final results. The questionnaire results are consistent with this process evidence: 86.7% of students agreed or strongly agreed that the unified report structure helped them organize the experimental process more clearly, 93.3% believed that the "data–model–training–evaluation–reflection" workflow improved their awareness of experimental norms, and 86.6% agreed or strongly agreed that failure-case analysis helped them understand model limitations.

Fourth, the regulated use of AI tools helped transform AI assistance into a traceable learning process. In the course assignments, students were required to record how AI tools were used, what suggestions were obtained, what manual revisions were made, and how the final results were verified. This design made AI-supported debugging, code explanation, and report refinement visible in the learning process. The questionnaire results show that students used AI tools frequently: 80.0% reported frequent use, and 13.3% reported using AI tools in almost every experiment. More importantly, 93.3% agreed or strongly agreed that AI tools helped them understand error messages and code logic more quickly, 80.0% believed that AI helped them form debugging ideas rather than simply provide answers, and 93.3% considered the course management of AI tool use neither overly restrictive nor laissez-faire. These results suggest that regulated AI assistance can support programming practice while maintaining students' responsibility for verification and understanding.

Student perceptions also indicate that the overall reform design was generally accepted. Regarding the hierarchical task structure, 86.7% of students agreed or strongly agreed that the "model reproduction–performance optimization–independent innovation" arrangement was reasonable, 86.6% believed that hierarchical practice could accommodate students with different programming foundations, 80.0% considered Level 2 tasks appropriately challenging, and 86.7% believed that Level 3 tasks provided sufficient space for independent exploration. These responses suggest that the hierarchical design helped balance accessibility and challenge, allowing students with different foundations to participate in the same mainline task while progressing toward more open-ended practice.

At the same time, the results reveal several issues that require further improvement. Some students still perceived pressure from experimental workload, insufficient support for students with weaker programming foundations, and the influence of computing platforms or environment configuration on learning experience. For example, 26.7% of students agreed or strongly agreed that the workload was heavy, 33.3% believed that support for students with weak programming foundations was still insufficient. In addition, 46.7% agreed or strongly

agreed that AI-use norms still need further clarification. These findings indicate that future implementation should provide more step-by-step debugging cases, clearer AI-use examples, more stable computing environments, and more differentiated support for students with weaker foundations.

Taken together, the historical cohort comparison, stratified Level 3 analysis, and process-based learning evidence provide preliminary support for the feasibility and potential effectiveness of the proposed reform framework. Implementation of the framework was associated with higher project completion and practical project scores, while most students with weaker initial programming foundations were also able to complete the independent-design stage. The assignment and questionnaire evidence further suggests improvements in knowledge integration, programming practice, project awareness, and responsible AI use. However, because the study did not include a concurrent control group or parallel single-level and two-level conditions, these analyses were treated as supplementary evidence of the framework's feasibility and potential effectiveness, rather than as confirmation that the three-level progression is optimal.

6. Conclusion and Discussion

This paper proposes an AI-assisted task-driven reform framework for computer vision education. By using ethnic costume virtual try-on and image editing as a unified mainline task, the framework connects fragmented knowledge units such as classification, segmentation, detection, generation, editing, and multimodal understanding. It further constructs a hierarchical practice path from baseline reproduction to model modification and independent design, while integrating full-process project training and traceable AI-assisted learning records. Evidence from the historical cohort comparison, student assignments, project records, and questionnaire responses provides preliminary support for the feasibility and potential effectiveness of the framework. As an initial classroom implementation, this study provides encouraging, context-based evidence for the feasibility and potential effectiveness of the framework. Further research across multiple teaching cycles and larger samples will examine its broader applicability and the incremental contribution of the three practice levels, while refining differentiated learning support and the assessment of responsible AI use.

Conflict of Interest

The authors declare that they have no conflicts of interest to this work.

Acknowledgement

This research was funded by 2024 and 2025 Education and Teaching Reform Research Project of Yunnan University (No. 2025Y12).

References

Anderson, L. W., and Krathwohl, D. R., eds. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, 2001.

ARNALD, J. (2025). Research on the Reform of Education and Teaching Methods in the Era of Artificial Intelligence. *Multidisciplinary Research in Arts, Science & Commerce (23)*, 30.

Atlantis Press.Zhang, H. (2025, April). Exploring AI-Integrated Curriculum Reform in University Computer Science Program. *In 2025 11th International Symposium on System Security, Safety, and Reliability (ISSSR)* (pp. 41-50). IEEE.

Biggs, J. Enhancing teaching through constructive alignment. *Higher Education*, 1996, 32: 347–364.

Black, P., and Wiliam, D. Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice*, 1998, 5(1): 7–74.

Cun, T., & Sha, Z. (2023, June). Teaching Reform of Computer Vision Course in the Context of Artificial Intelligence. *In 2023 4th International Conference on Education, Knowledge and Information Management (ICEKIM 2023)* (pp. 782-795).

Holmes, W., & Miao, F. (2023). *Guidance for generative AI in education and research*. Unesco Publishing.

Jiang, Q., Jin, X., Miao, S., & Wang, P. (2025). Linear Algebra Teaching Reform for Artificial Intelligence Major. *Contemporary Education and Teaching Research, 06*(2), 46-56.

Jin, Y., Zhang, J., & Dai, H. (2024, August). Exploration on Teaching Content Reform of Programming Course in the Era of AI. In *International Computing and Combinatorics Conference* (pp. 30-35). Singapore: Springer Nature Singapore.

Kasneci, E., Sessler, K., Küchemann, S., et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 2023, 103: 102274.

Liao, Y., Gao, J., & Chen, F. (2024, November). Teaching Reform of New Engineering Deep Learning Courses Combining Ideology and Politics Leading, Project Driving and Intelligent Evaluation. In *2024 International Conference on Digital Technology and Intelligent Education (ICDTIE)* (pp. 10-15). IEEE.

Merrill, M. D. A task-centered instructional strategy. *Journal of Research on Technology in Education*, 2007, 40(1): 33–50.

Miao, F., and Holmes, W. *Guidance for Generative AI in Education and Research*. UNESCO, 2023.

Rosenberg-Kima, R. B., Merrill, M. D., Baylor, A. L., and Johnson, T. E. Explicit instruction in the context of whole tasks: The effectiveness of the task-centered instructional strategy in computer science education. *Educational Technology*

Research and Development, 2022, 70: 1627–1655.

Tomlinson, C. A. *The Differentiated Classroom: Responding to the Needs of All Learners*. 2nd ed. ASCD, 2014.

How to Cite: Huang, S., Huang, H. & Wang, H. (2026). AI-Assisted Task-Driven Reform of a Computer Vision Course: A Hierarchical Practice Framework for Programming Competence. *Contemporary Education and Teaching Research, 1*(Special Issue), 37-49.
https://doi.org/10.61360/BoniCETRAIRC262020430104