A Project-Based Learning Method for Intelligent Security Face Recognition Systems Based on Cloud–Edge–Device Collaborative Architecture

Rui Cao

School of Computer Science and Engineering Xi'an Technological University Xi'an, China

1275280817@qq.com

Abstract. This paper takes the "Intelligent Security System V1.0" project, in which the author participated as a core R&D member, as the research object, and systematically reviews the complete research-based learning process from project initiation, literature review, technology selection, system architecture design, module development, integration testing, to final deployment and documentation. Rather than focusing on the theoretical derivation or code-level implementation details of a particular face recognition algorithm, this paper emphasizes how, in the context of a complex system engineering project, an intelligent security system integrating edge computing and face recognition technology was accomplished through independent inquiry, team collaboration, resource integration, and continuous iterative optimization. Through this project, the author gained a deep understanding of the core value of research-based learning—the transition from passive reception to active knowledge construction—and mastered scientific methodologies in engineering practice, including problem decomposition, experimental design, performance tuning, version management, and documentation. This experience has laid a solid foundation for further academic study and professional development.

Keywords: Research-based learning intelligent security; edge computing; face recognition; system design; cloud–edge–device collaboration

1. Introduction

In traditional engineering education, knowledge transmission is often dominated by "receptive learning": instructors lecture on theoretical frameworks, and students demonstrate mastery through attending classes, solving exercises, and taking examinations. While this model offers advantages in the efficiency of knowledge delivery, its fundamental limitations are equally evident—when students face an open-ended, real-world engineering project with no standard solutions, or even where the problem itself requires self-definition, they are often at a loss. The development of an intelligent security face recognition system is precisely such a complex engineering problem: there are no ready made lab manuals, no fixed development platforms, no explicit technical pathways, and certainly no standard answers to consult. Learners must autonomously decide which algorithms to use, how to design the system architecture, and how to evaluate performance metrics. Such "ill structured problems" are challenges that traditional education has long avoided but engineering practice frequently encounters.

Project Based Learning (PBL) is an effective pedagogical model for addressing this challenge. Its core characteristics include: authentic drivers (learning triggered by real world problems or needs), independent inquiry (learners self plan pathways, search for resources, and design solutions), product orientation (learning outcomes take the form of a runnable, verifiable artifact, such as a software system or hardware device), and iterative reflection (continuous testing, failing, adjusting, and improving in a closed loop). Research has shown that PBL significantly enhances learners' intrinsic motivation, critical thinking, collaboration skills, and ability to solve complex problems. In recent years, many universities worldwide have promoted PBL in computer science, electronic information, and related disciplines, achieving notable teaching reform outcomes.

Based on this understanding, this paper takes the project "Design and Implementation of an Intelligent Security Face Recognition System Integrating Edge Computing" as a case study to

https://doi.org/10.58244/joerai.263888

systematically examine the complete research based learning process—from project initiation, literature review, technology selection, system architecture design, module development, integration testing, to final deployment and documentation. Rather than focusing on the theoretical derivation or code level implementation details of a specific face recognition algorithm, this paper emphasizes how, in the context of complex systems engineering, an intelligent security system integrating edge computing and face recognition technology was accomplished through independent inquiry, team collaboration, resource integration, and continuous iterative optimization. On this basis, a transferable project based learning methodology is distilled, and the underlying learning psychology and educational principles are revealed, with the aim of providing a methodological reference for similar research based learning endeavors.

Before formally elaborating the methodology, it is necessary to define the three tier goal structure of project based learning. This framework serves as a basis for designing and evaluating subsequent learning activities: the knowledge tier (mastering specific technologies directly relevant to the project, such as face recognition principles, edge computing architectures, lightweight neural networks, database design, and front /back end development); the methodology tier(acquiring transferable engineering methods, including literature search strategies, technology selection and evaluation, experimental design, problem decomposition, version management, and technical documentation); and the competency tier(developing enduring learning dispositions and thinking habits, such as patience in debugging, critical thinking, team communication skills, and awareness of engineering ethics). The subsequent chapters of this paper will closely follow this three tier framework in their discussion.

2. Core Process of the Project-Based Learning Method: A Five-Phase Breakdown

Based on the practical experience of this project, the project-based learning method can be divided into five core phases: Problem Definition and Goal Setting, Literature Review and Technology Mapping, Key Technology Development, Coding, Testing, and Iteration, and Knowledge Output and Artifact Consolidation. Each phase entails specific learning tasks, potential challenges, and actionable methodological strategies, which learners may flexibly adapt according to their own project types. This section presents the overall framework, while subsequent chapters (III to V) will elaborate on the details.

2.1 Phase I: Problem Definition and Goal Setting

The core objective of this phase is to transition from observing real world scenarios to progressively narrowing down to an executable project scope. Learners should begin by identifying pain points in everyday life that could be addressed through technical improvements. For example: low throughput at school dormitory access control during peak hours, frequent tailgating by strangers, and frequent recognition failures or false positives of face recognition devices under complex lighting conditions. These practical problems motivate learners to conceive the initial idea of integrating edge computing with face recognition. Subsequently, learners should systematically investigate the limitations of existing solutions: the conventional "camera + cloud based recognition" approach relies on a stable network connection, fails when offline, and uploading facial images to the cloud raises privacy concerns. On this basis, potential technical breakthroughs are introduced through academic lectures, technology surveys, or cutting edge literature—edge computing being one such breakthrough. By offloading part of the computational tasks to edge devices located close to data sources, edge computing reduces latency, decreases bandwidth consumption, and protects data privacy, precisely addressing the shortcomings of traditional cloud solutions. Finally, one to two in depth discussions with an academic supervisor are conducted to define the system boundaries (which functions are implemented in the V1.0 release and which are deferred to future iterations), the core feature set (face enrollment, real time detection and recognition, offline operation, alert push, cloud based log management), and performance metrics (recognition latency ≤100 ms, recognition rate ≥95%, etc.). Goal setting follows the SMART criteria: Specific, Measurable, Achievable, Relevant, and Time bound. The learning deliverable of this phase include a project initiation report, a feature list, performance metrics, and meeting minutes documenting the discussions with the supervisor.

2.2 Phase II: Literature Review and Technology Mapping

After finalizing the research topic, learners are often confronted with a vast volume of Chinese and English technical literature, which can readily lead to information overload and loss of direction. The funnel-based retrieval method proves highly effective at this stage. This method comprises three progressive levels: Level 1 – broad retrieval, using keyword combinations such as "edge computing + face recognition," "edge computing face recognition" (in Chinese), and "intelligent security system" across databases including CNKI, IEEE Xplore, and the ACM Digital Library, yielding approximately 800 relevant papers; Level 2 – rapid filtering, restricting the search to the last five years, citation count $\geqslant$10 or publication in CCF-recommended conferences/journals, and full-text availability, thereby reducing the set to about 80 papers; Level 3 – close reading and categorization, reading the abstracts and conclusions followed by close reading of 20–30 core papers, and organizing notes into four categories: "edge architecture design," "model compression and light weighting," "face recognition datasets and evaluation," and "real-world system case studies."

The construction of a technology road map is a critical step in translating literature findings into an engineering blueprint. Learners adopt a top-down decomposition method, dividing the system into four layers: data acquisition layer (USB camera / network camera), edge computing layer (ARM-based development board + lightweight model + SQLite), cloud management layer (Spring Boot + MySQL + Vue), and end–edge–cloud communication layer (MQTT + Restful API, AES-256 encryption).

2.3 Phase III: Key Technology Development

This phase represents the period of greatest pressure and most rapid growth in project-based learning. Learners must solve non-trivial technical challenges under real-world constraints. Typical technology development scenarios in this project included model Light weighting, edge–cloud data consistency, and engineering-oriented database design. Chapter IV will elaborate on these development processes and the underlying learning methodologies, thereby revealing the general strategy of "problem decomposition → theoretical study → experimental validation."

2.4 Phase IV: Coding, Testing, and Iteration

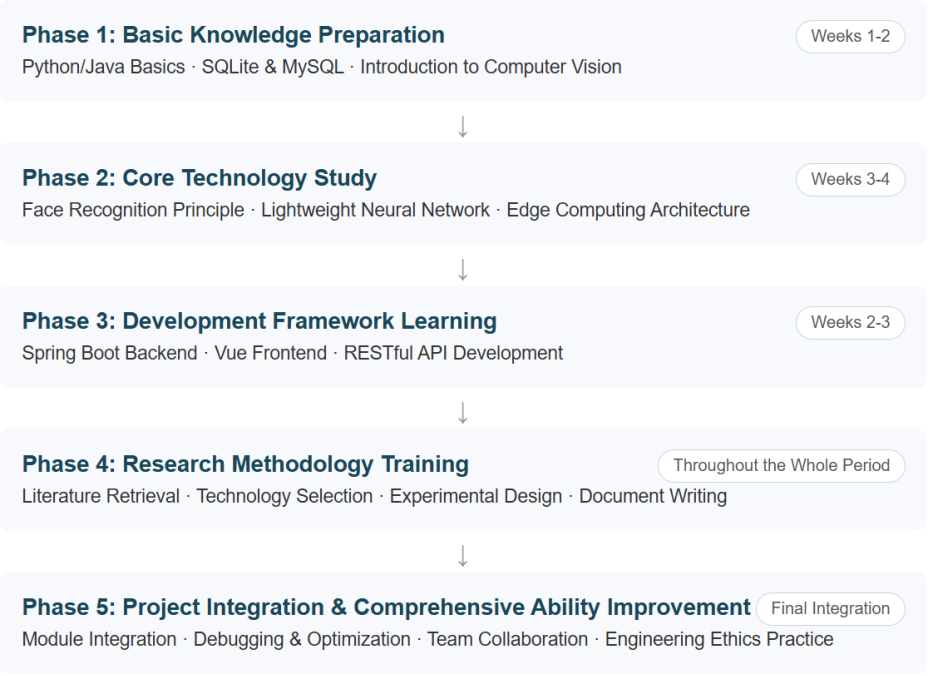

Figure 1. Five-phase learning pathway planning for the project

This phase emphasizes systematic development environment setup, version control norms, systematic debugging methods, and performance optimization methodologies. Learners set up the environment following the order of "edge side → cloud back-end → cloud front-end → communication middleware," performing verification tests immediately after completing each

environment. For version control, a Git branching strategy is adopted: the main branch stores stable releases, the dev branch is used for daily development, and feature/xxx branches are created for developing independent functionalities. Commit messages follow the Conventional Commits format. During debugging, learners shift from "random trial-and-error" to hypothesis-driven debugging, synthesizing a six-step procedure. In performance optimization, the principle of "measure first, then optimize" is adhered to, using PyTorch Pro-filer and c profile to locate bottlenecks, and employing batch-processing inference and pipeline parallelism strategies respectively.

3. Preliminary Exploration: Literature Review and Technology Mapping

Literature review is not merely a means of information acquisition but also an important learning method—it trains learners to rapidly locate high quality information in the vast ocean of knowledge and to construct their own knowledge frameworks. The "funnel based retrieval method," which the learner encountered during the "Information Literacy Training" course in the second semester of their sophomore year, played a critical role in this project and was implemented in three stages:

3.1 Broad retrieval

Using keyword combinations such as "edge computing face recognition," "edge computing + face recognition," "intelligent security system," and "cloud–edge–device collaboration" in both Chinese and English across CNKI, IEEE Explore, and Web of Science, approximately 800 relevant papers were retrieved, broadly covering related research directions.

3.2 Rapid filtering

Three filtering criteria were applied: publication within the last five years, citation count ≥10 or publication in CCF recommended conferences/journals, and full text availability. The set of papers was thereby reduced from 800 to about 80.

3.3 Close reading and categorization

The abstracts and conclusions of the 80 papers were read, and 20 core papers were closely read. Notes were organized into four categories: "edge architecture design," "model compression and light weighting," "face recognition datasets and evaluation," and "real world system case studies."

This iterative filtering process not only improved efficiency but, more importantly, trained the learner's information evaluation skills—learning to distinguish between "trendy but low quality" papers and "solid and relevant" work. In addition, the learner used Zotero for literature management, annotating each paper with its core contribution, limitations, and relevance to the project. Such structured note taking greatly facilitated later retrospective access during writing and technical decision making.

The process of constructing the technology road map likewise demonstrated the value of the learning method: proceeding from top level functional decomposition downward, annotating inter module dependencies, selecting specific technology stacks, and assessing risks. For example, at the edge computing layer, the learner initially considered using a Raspberry Pi 4B. However, literature review revealed its limited NPU compute capability, whereas a domestic RK3588 development board offered stronger NPU performance and lower power consumption, leading to a timely hardware selection adjustment. Such evidence based decision making is precisely an advantage of project based learning.

To further strengthen literature review skills, the learner also practiced the "snowball method"—backward tracking of important original papers from the reference list of a high quality review article—and "forward citation tracking" (searching Google Scholar for subsequent works that have cited core papers). Together, these methods constitute a complete methodological system for literature review.

Figure 2. Funnel-based literature retrieval workflow

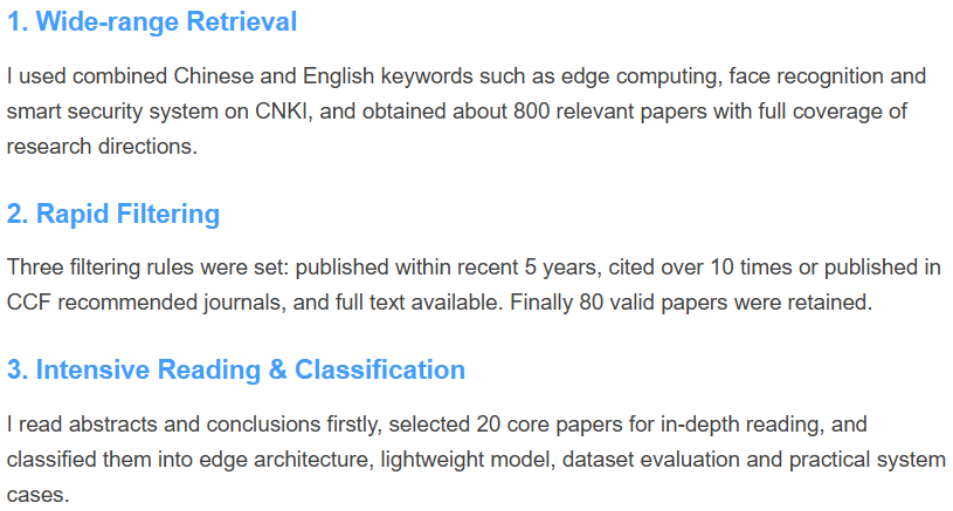

Figure 3. Illustration of the funnel-based literature retrieval method

During the construction of the technology road map, the learner employed several auxiliary tools: XMind for mind mapping, Draw.io for architecture diagramming, and periodic reviews with the supervising instructor. The technology road map is not merely a visualization of the technical solution but also a living document, iterated continuously as the project progresses. For example, the initial road map considered only a face recognition model at the edge side; however, subsequent development revealed the need to incorporate a liveness detection module to prevent photo attacks. Consequently, liveness detection was added as a sub-module to the road map. Such dynamic adjustments reflect the balance between planning and adaptation inherent in project-based learning. The learner also summarized ten essential elements that a technology road map should include: system boundary, data flow, control flow, hardware selection, software stack, communication protocol, security mechanisms, performance metrics, risk checklist, and alternative solutions. Each element is validated through both literature review and experimentation, ensuring that the road map is not an untethered abstraction.

Figure 4. Technology road map

4. Breakthrough: Learning Methods in Key Technology Development

4.1 Model Light weighting: A Combined Optimization Strategy from MobileFaceNet to ECA CNN ViT

In the early stage of system development, the learner adopted the open-source pre-trained MobileFaceNet model, which achieved an accuracy of over 99% on the LFW dataset with a model size of approximately 15 MB, making it suitable for mobile deployment. However, after deploying the model to an ARM Cortex-A53 CPU development board without GPU acceleration, the inference

time for a single 640×480 face image was 120 ms, far exceeding the real-time target of 50 ms. The core issue lies in the trade-off among speed, accuracy, and model size.

The learner first studied model pruning techniques. Using torch.nn.utils.prune in PyTorch, 30% structured pruning was applied to the core convolutional layers. The model size was reduced to 11 MB, inference speed improved to 85 ms, but accuracy dropped from 98% to 92%. To recover accuracy, knowledge distillation was introduced: the original MobileFaceNet served as the teacher network and the pruned model as the student network. Distillation training was performed on the CASIA-WebFace dataset for 10 epochs, raising accuracy back to 97.5%. Finally, INT8 quantization was performed using 500 face images as the calibration dataset. After quantization, the model size was compressed to 3.8 MB, inference latency decreased to 42 ms, and accuracy remained above 97%, fully satisfying the real-time requirements.

Throughout the optimization process, the learner adhered to the principle of "changing only one variable at a time." Before and after each optimization step, changes in three key metrics were recorded, resulting in a detailed optimization log (see Table 1). This experience enabled the learner to deeply understand that a single optimization method is often insufficient; only by combining multiple techniques—pruning, distillation, quantization—and fine-tuning at each stage can engineering targets be achieved. More importantly, the learner acquired a standard workflow for deep learning model deployment, a capability transferable to any end-side AI project.

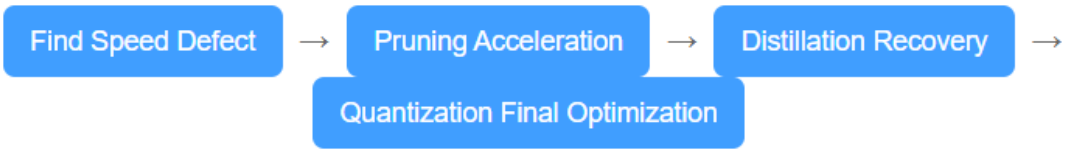

Figure 5. Learning workflow for model Light weighting optimization

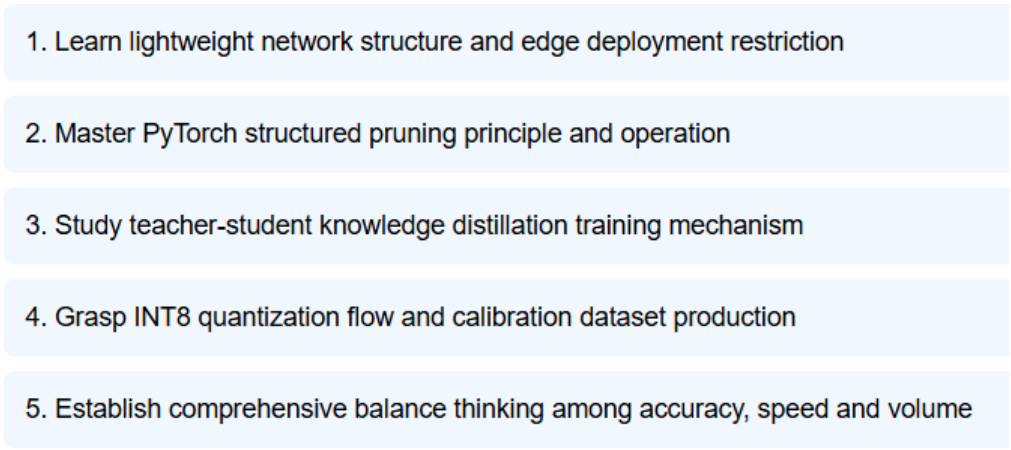

Figure 6. Key learning points for combined optimization of lightweight models

4.2 Edge–Cloud Collaboration: A Learning Path for Data Consistency and Offline Fault Tolerance

The initial design assumption of the system was that the edge device would always remain online; however, network fluctuations and disconnections are inevitable in real-world environments. In a simulated offline test, after disconnecting the edge device's network cable for one hour and then reconnecting, the cloud recognition logs exhibited a large number of duplicate records, timestamps were disordered, and the updated personnel database from the cloud could not be synchronized to the edge side, resulting in an administrator being denied access. The learner recognized this as a typical distributed-system data inconsistency problem.

To address this, the learner proactively consulted the chapters on eventual consistency, conflict resolution, and replica synchronization in Designing Data-Intensive Applications and Distributed Systems: Concepts and Design. After theoretical study and reflection, a "local-first, asynchronous sync" strategy was designed: each recognition operation is first written to a local SQLite database and

simultaneously placed into an in-memory queue. An independent background thread attempts every 5 seconds to push the records in the queue to the cloud via a Restful API. Upon successful write to the cloud MySQL database, an acknowledgment message is returned, and the edge device removes the corresponding record from the queue. If the push fails, the record is retained and retried in the next cycle; upon consecutive failures, exponential back off is applied. For database synchronization, the cloud is defined as the authoritative source. The edge side actively pulls the version number of the cloud database every 30 minutes or at startup; if the version number has been updated, an incremental synchronization is performed. This strategy resolves data loss and duplication during offline periods and ensures eventual consistency.

The learner's key insight was that consistency design in distributed environments often determines system success or failure more than any single algorithm. The correct approach is to first systematically study the theory (CAP theorem, BASE principles), draw a state machine diagram, and only then write code. In addition, the learner designed a conflict resolution rule: when the same personnel record is modified concurrently on the edge and the cloud, the record with the later timestamp is adopted (Last-Write-Wins), and a conflict log is retained for manual intervention. This design was validated through multiple rounds of simulation testing, ensuring system robustness under extreme network conditions.

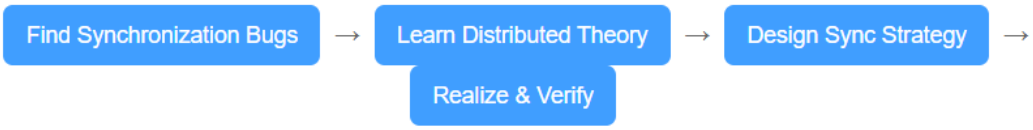

Figure 7. Problem-solving process for edge–cloud data consistency

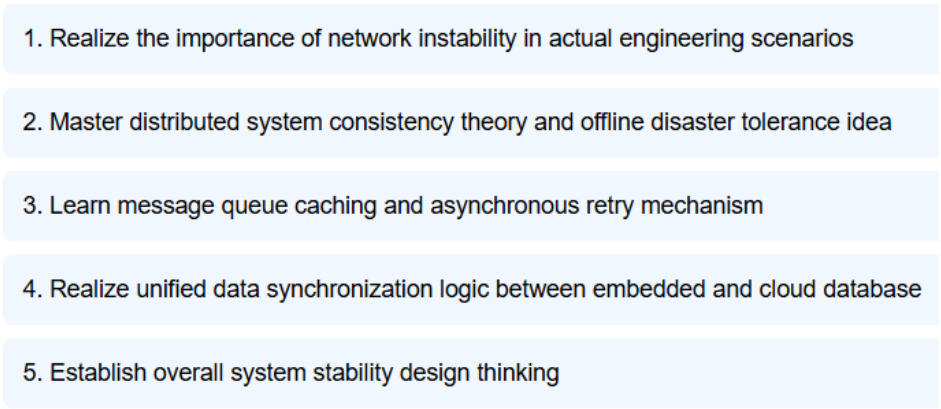

Figure 8. Key learning points for edge–cloud data issues

4.3 Database Design: A Standardized Methodology from Theory to Practice

Although the learner had achieved good grades in the sophomore-year Database Principles course, three typical errors still emerged when designing the actual table structures: using "name" as the primary key, leading to name collisions; storing face image base64-encoded strings directly in the log table, causing slow queries; and inconsistencies between the edge-side SQLite table structures and the cloud-side MySQL table structures, resulting in synchronization issues. This is a true reflection of the gap between textbook knowledge and real-world engineering, underscoring the importance of database design.

Driven by practical problems, the learner addressed three real pain points: primary key conflicts, slow queries, and cross-end inconsistency. For the primary key issue, the final solution was to use UUIDs on the edge side and auto-increment integers on the cloud side, with a mapping table established between the two. For the large-field issue, the log table stored only the file path of the image, while the images themselves were saved to MinIO object storage or the local file system. This reduced the size of a single log entry from several hundred kilobytes to just tens of bytes. For the structural inconsistency issue, the learner redesigned a unified data model, defining the attributes, types, constraints, and foreign keys of all entities in a standalone document. A data access layer (DAL) was

then implemented, providing a unified CRUD interface that internally generated the appropriate SQL dialect based on the database type, thereby achieving engineering-level unification of the table structures.

After resolving these issues, the learner proactively revisited database theory: re-studying normalization, indexing, and transaction isolation levels to solidify the underlying principles. Using EXPLAIN, the query plans before and after optimization were analyzed, truly internalizing textbook knowledge into engineering competence. This process confirms the learning principle of "output-driven input"—only when encountering performance bottlenecks or data anomalies under real constraints does database theory cease to be a passive memorization point for examinations and become an actively sought-after solution. The learner also additionally studied SQLite's WAL mode and MySQL's InnoDB isolation levels, selecting the READ COMMITTED isolation level for the project to strike a balance between consistency and performance.

Through this closed loop of "identifying database problems → repairing and optimizing one by one → unifying table structures → revisiting theory for deeper understanding," the learner gradually developed an engineering mindset: designing holistically from the perspectives of data life cycle, scalability, and consistency, rather than being confined to writing individual CREATE TABLE statements.

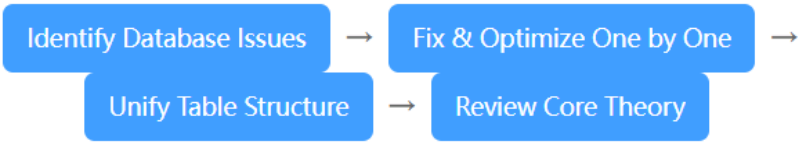

Figure 9. Engineering workflow for database design

1. Theory into Practice: Recognize the gap between textbook knowledge and real-world engineering, understand the importance of database design

2. Problem-Driven Learning: Learn from three real issues: primary key conflicts, slow queries, and schema inconsistency

3. Standard Design: Learn UUID, path storage, unified models, and DAL

4. Theoretical Improvement: Relearn normalization, indexes, and transaction isolation levels to strengthen fundamentals

5. Engineering Thinking: Build a holistic mindset for data lifecycle, scalability, and consistency design

Figure 10. Key learning points for database design

5. Implementation: A Methodology for Coding, Testing, and Iteration

5.1 Development Environment Setup and Version Management Standards

The technology stack of this project involved embedded Python/PyTorch, Java Spring Boot, a Vue frontend, and MQTT communication, resulting in a relatively high complexity of environment setup. The learner established a clear build order: • edge-side base environment (cross-compilation toolchain, PyTorch Mobile, OpenCV), • cloud back-end environment (JDK, Maven, MySQL, Spring Boot), • cloud front-end environment (Node.js, Vue CLI), and • communication middleware (Mosquitto MQTT broker, Postman API testing). Upon completing each environment, a "Hello World"-level verification was immediately performed to prevent error accumulation.

Version management adopted a simplified Git Flow: the main branch held deployable stable versions, with only the lead developer permitted to merge; the dev branch was used for daily development and was required to always be compilable; each feature (e.g., face_detection, sync_protocol) was

developed in its own branch and merged into dev upon completion. Commit messages followed the format type: subject, where type included feat (new feature), fix (bug fix), docs (documentation), and refactor (code refactoring). Although these standards increased the initial workload, they significantly reduced communication overhead during later debugging and collaboration. The learner also developed the habit of pushing code at the end of each day and performing a branch cleanup weekly, ensuring code traceability.

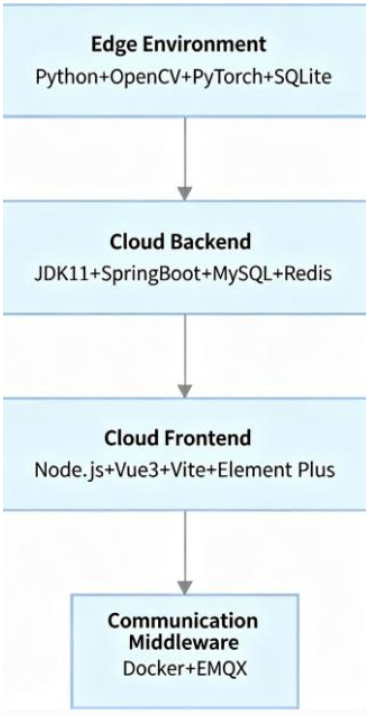

Figure 11. Learning path for development environment setup

5.2 Systematic Debugging Methods: From "Random Trial-and-Error" to "Hypothesis-Driven"

During the integration testing phase, the learner's project experienced a "darkest hour." Real-world testing revealed a critical occlusion defect: while the system achieved a recognition rate above 96% in the laboratory environment, the rate plummeted to 60% when faces were obscured by masks or scarves. Three consecutive days of attempts to replace preprocessing algorithms, adjust detection thresholds, and retrain the model brought no improvement, marking a conventional testing bottleneck.

This process prompted the learner to engage in practice-driven cognitive learning: distinguishing the gap between ideal laboratory conditions and real-world complex scenarios, and emphasizing robust system design. On the fourth day, the learner changed strategy—no longer making blind attempts, but first precisely reproducing the failure scenarios (masked faces, low-angle lighting) and then delving into the underlying principles to identify the root cause. Drawing on domain knowledge, a hypothesis was formulated: the model was distributing attention to occludable regions (the lower face) rather than focusing on regions less likely to be occluded (eyes and eyebrows). This exemplified principle-based deep learning: understanding the distribution patterns of facial features and distinguishing stable feature regions from occluded, ineffective ones.

To test this hypothesis, the learner visualized the attention heat maps of the model's intermediate layers and found that occluded regions were indeed assigned high weights. This led to technology-advancement learning: systematically studying the principles of spatial attention mechanisms and mastering the embedding of lightweight attention modules. The solution was to insert a lightweight spatial attention module into the intermediate layers of the feature extraction network, allowing the network to automatically learn a spatial weight map, thereby completing an attention-mechanism optimization.

Simultaneously, the learner engaged in data set-construction learning: acquiring techniques for real-world sample collection, occlusion-based data augmentation, and fine-tuning with small samples. An additional 300 face images with masks, scarves, and bangs occlusion were collected for data augmentation, and the final dataset was expanded for fine-tuning. After fine-tuning, the recognition rate under occlusion scenarios recovered from 60% to 93%.

Through this complete closed loop, the learner achieved a mindset-upgrade learning: moving beyond fixed parameter tuning to develop a mindset of solving real-world problems at the network architecture level. A six-step hypothesis-driven debugging method was summarized: precise reproduction → hypothesis formulation → minimal experiment design → solution proposal → targeted data collection → validation and documentation. This method was later successfully transferred to other debugging scenarios (e.g., network latency jitter, memory leaks), becoming a core component of the learner's engineering competence. From a psychological perspective, frustration during debugging often leads to cognitive shortcuts; learners must deliberately train their meta cognitive monitoring ability to "pause-reflect-hypothesize-experiment."

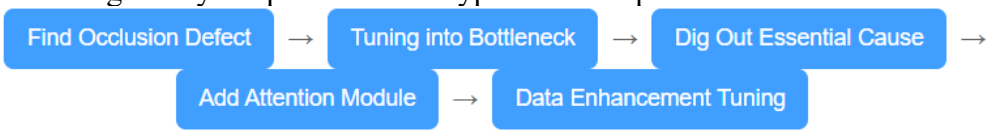

Figure 12. Learning path for development environment setup

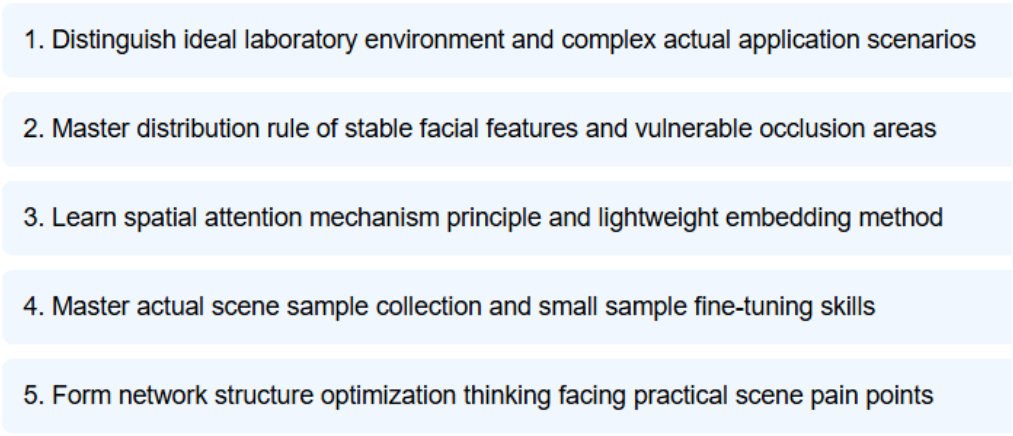

Figure 13. Debugging and optimization workflow for occlusion scenario recognition

5.3 Performance Optimization Methodology: The Profiling–Optimization–Validation Loop

When the system prototype was completed and subjected to a stress test simulating 100 concurrent recognition requests (25 frames per second per stream), a bottleneck was identified: recognition latency surged from 42 ms to 210 ms, and the system approached a stall. The learner first conducted performance profiling using PyTorch Profiler and c profile, acquiring skills in performance testing and bottleneck localization. The analysis revealed that the bottleneck was not in model inference per se but in two aspects: first, sequential inference on each face without batching; second, serial execution of image acquisition, preprocessing, inference, and post processing, leaving the CPU idle while waiting for I/O.

To address the first bottleneck, the learner studied batched inference: mastering batch-based vectorized computation by cropping all detected face regions from 10 frames, packing them into a single batch, and feeding the batch into the model. This improved throughput by a factor of four and reduced latency to 95 ms. For the second bottleneck, the learner studied pipelined architectures: understanding multi-threading, queues, and asynchronous parallel design concepts. A pipeline parallelism architecture was adopted, dividing the entire processing flow into four independent threads—image acquisition, preprocessing, inference, and postprocessing—connected via thread-safe queues, allowing different stages to process different frames concurrently. After

optimization, the average recognition latency under 100 concurrent streams was 88 ms, meeting the design specification of ⩽100 ms.

Throughout the optimization process, the learner adhered to data-driven optimization: conducting comparative tests before and after each optimization step to validate effectiveness with empirical data. Engineering discipline was also emphasized: complex optimizations were accompanied by comments and documentation to ensure code maintainability. Specific principles included: measure first, then optimize (no blind tweaking), change only one variable at a time (for clear attribution), and prioritize readability (optimize moderately, comment thoroughly). The performance optimization experience led the learner to recognize that engineering deployment requires systematic consideration of the trade-off between throughput and latency, and that architectural-level adjustments are often more effective than parameter tuning. Additionally, a simple adaptive load-balancing mechanism was introduced: when the queue length exceeded a threshold, some non-critical frames were dynamically skipped to prevent the core recognition functionality from crashing. These engineering techniques derived from a deeper understanding of operating systems and concurrency.

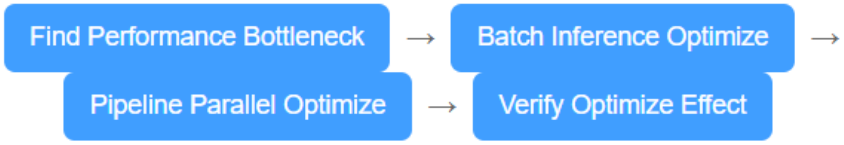

Figure 14. Key points of technical learning and mindset improvement for occlusion robustness issues

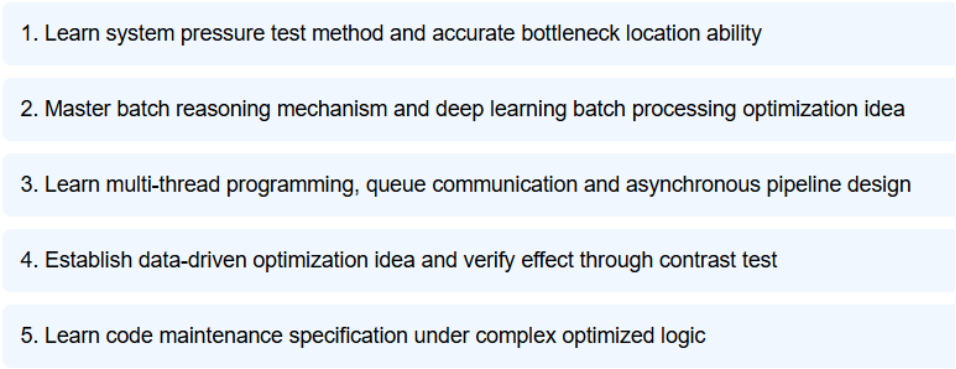

Figure 15. Optimization workflow for high-concurrency performance bottlenecks

6. Core Principles of the Project-Based Learning Method

Building on the above practical experience, this chapter distills five core principles from the perspective of the learning sciences, explaining why project-based learning effectively promotes deep learning and capability transfer. Each principle is analyzed in depth with reference to specific project examples and cognitive theories.

6.1 Principle I: Active Construction vs. Passive Reception

Constructivist learning theory posits that knowledge is not passively received but actively constructed by learners through interaction with the environment. In this project, the learner did not first complete the study of edge computing, face recognition, and database design before taking action; rather, while addressing the specific problem of "how to achieve inference under 50 ms on an ARM CPU," the learner temporarily acquired knowledge of pruning, distillation, and quantization as needed. This "just-in-time learning" model is more consistent with the brain's cognitive principles: urgency enhances attention, and immediate application strengthens memory encoding. The educational implication is that project problems should be appropriately challenging and should allow for "learning by doing." Further research indicates that actively constructed knowledge is more easily transferred to new contexts because learners encode both the knowledge itself and the conditions of its use simultaneously.

6.2 Principle II: Situated Cognition and Decision-Making Under Real-World Constraints

Situated learning theory posits that knowledge cannot be separated from the contexts in which it is used. An algorithm's accuracy metric is just a number in a paper, but in the project, it means whether a classmate wearing a mask will be denied entry to the dormitory. Real-world contexts endow technical metrics with ethical weight and decision-making significance. Every technical decision made by the learner under real constraints—such as accepting a 3% loss in accuracy in exchange for a fourfold increase in speed—represents truly internalized knowledge. Therefore, project-based learning should, whenever possible, select topics with real users or real deployment environments. In this project, the learner also considered physical constraints such as device power consumption and heat dissipation—factors that pure theoretical research would never encounter but that are precisely critical for engineering deployment.

6.3 Principle III: Feedback-Driven Iteration

High-efficiency feedback is a critical factor for accelerated skill growth. In this project, feedback sources included compiler errors (second-scale feedback), unit tests (minute-scale), performance data (hour-scale), and supervisor reviews (day-scale). Each instance of feedback served as a "calibration," forming a cycle of hypothesis → action → observation → reflection. The learner adjusted hypotheses, modified code, and retested based on feedback—a high-density feedback loop that traditional classrooms can hardly provide. Educational design should help learners establish a fast "code-run-test" closed loop. Furthermore, the learner proactively created automated test scripts that ran regression tests automatically after each code commit, shortening the feedback cycle to the minute scale and significantly improving debugging efficiency.

6.4 Principle IV: Problem Decomposition to Reduce Cognitive Load

Human working memory has a limited capacity (approximately 4±1 chunks of information). When confronted with a grand task such as "design an intelligent security system," a novice's working memory becomes instantly saturated, preventing effective thinking. Problem decomposition breaks a complex task into several subtasks, each simple enough to be handled within working memory capacity. The decomposition tools used in this project included technology road maps, Git branching strategies, and daily task lists. Instructors should demonstrate how to decompose problems and require learners to specify subtasks clearly in their weekly reports. Cognitive load theory also suggests that novices need external scaffolding (e.g., modular decomposition templates) for subtask design, with the scaffolding gradually removed as competence improves.

6.5 Principle V: Metacognitive Development and Strategy Transfer

Metacognition refers to "cognition about cognition"—that is, learners' awareness of how they learn and how they solve problems. The highest goal of project-based learning is not merely to complete the current project but to cultivate the ability to "learn how to learn." In this project, the learner gradually realized that "when debugging makes no progress for two days, one should stop and redefine the problem," learned to systematize personal methods and strategies (e.g., the funnel-based retrieval method, hypothesis-driven debugging), and transferred these strategies to new domains. The educational implication is to schedule a methodology debriefing after the project, requiring learners to distill three to five transferable strategies and encouraging them to keep learning logs. Metacognitive training can be reinforced through explicit reflective writing and team sharing. In this project, the learner wrote a weekly learning report, which included a dedicated "methodological reflection" section, playing a key role in strategy transfer.

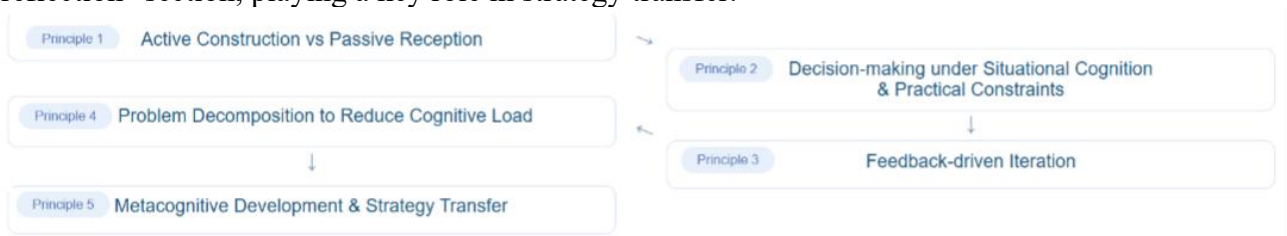

Figure 16. Core principles of the project-based learning method

7. Typical Learning Difficulties and Coping Strategies

Project-based learning does not always proceed smoothly; learners frequently encounter various difficulties in practice. Based on observations from this project and a review of the PBL literature, seven typical types of difficulties and their coping strategies are summarized below for the reference of educators and learners.

Difficulty 1: Information Overload and Loss of Direction. In the early stage of literature review, the sheer volume of materials can easily lead to anxiety. Coping strategy: Strictly implement the funnel-based retrieval method, progressively narrowing from broad to focused retrieval; set daily reading goals (e.g., close reading of two core papers) and take structured notes.

Difficulty 2: Hesitation in Technology Selection. Faced with multiple open-source solutions or frameworks, learners worry that a wrong choice may lead to rework later. Coping strategy: Conduct a 2–3 day rapid prototype validation (spike) for each candidate solution, compare core metrics, and then make an evidence-based decision. Also accept that "there is no perfect technology selection, only what fits the current constraints."

Difficulty 3: Debugging Bottlenecks and Emotional Breakdown. Being unable to solve a bug for several consecutive days may lead to thoughts of giving up. Coping strategy: Adopt the six-step "hypothesis-driven debugging" method to avoid random trial-and-error; set a time box (e.g., pause and seek help after 2 hours of no progress); keep a debugging log—sometimes writing down the problem process itself can trigger new insights.

Difficulty 4: Team Collaboration Conflicts (applicable to multi-person projects). Poor communication and uneven task distribution. Coping strategy: Hold daily stand-up meetings, use GitHub Projects or Trello for task tracking; establish clear code standards and code review processes; conduct regular team retrospectives.

Difficulty 5: Procrastination in Documentation. Reluctance to write documentation after code is written. Coping strategy: Break documentation into each iteration cycle, requiring that design documents be updated concurrently with feature completion; use documentation generation tools (e.g., Sphinx, Javadoc) to reduce burden; emphasize the value of documentation for one's future self and the team.

Difficulty 6: Over-optimization or Under-optimization of Performance. Either premature optimization complicates the code, or neglect of performance makes the product unusable. Coping strategy: First set quantifiable performance metrics (e.g., latency $\leqslant$100 ms); only when metrics are not met should targeted optimization be performed; after optimization, always run regression tests to ensure no functionality has been broken.

8. Knowledge Output and Artifact Consolidation

After the project had largely stabilized, the learner, under the supervisor’s recommendation, applied for a computer software copyright. The application process required submitting a user manual, a design specification, and the source code. Writing the user manual forced the learner to adopt the perspective of a user with zero background knowledge; non-project classmates were invited to trial the system and provide feedback, leading to multiple revisions. The preparation of the design specification systematically organized the architecture, module division, interface definitions, data flow, exception handling, and security design. During this writing process, the learner identified several technical decisions that had not been fully understood at the time (e.g., why asynchronous synchronization was chosen over strong consistency) and subsequently refined the design. The source code reorganization phase involved revisiting all code, unifying naming conventions, removing dead code, splitting overly long functions, and adding comments for complex logic. Although the software copyright application took approximately two weeks, it significantly strengthened the learner’s engineering documentation skills—a capability that is critically important in industrial R&D. Additionally, the learner organized the project outcomes into technical blog posts and a GitHub repository, engaged in basic interactions within the open-source community, and thereby laid the foundation for future code contributions.

9. Evaluation Methods and Continuous Improvement in Project-Based Learning

To ensure the quality of project-based learning, the learner simultaneously assumed the dual roles of "evaluated subject" and "self-evaluator." Throughout the project, a multi-dimensional evaluation approach was adopted: formative assessment (weekly progress reports, code reviews, design reviews), summative assessment (final demonstration, technical report, software copyright certificate), and learning process portfolios (debugging logs, study notes, iteration records). Among these, the learning process portfolio was particularly important, as it documented not only technical decisions but also failures and reflections, providing direct evidence of metacognitive development. In addition, the learner and the supervising instructor jointly developed quantitative scoring criteria covering functional completeness, performance metrics, code quality, documentation completeness, and depth of learning reflection. After the project concluded, the learner conducted a comprehensive retrospective, identifying strengths (e.g., systematic debugging methods) and areas for improvement (e.g., insufficiently detailed initial hardware selection). These evaluation results provided a basis for continuous improvement in subsequent projects.

10. Conclusion and Future Perspectives

Through the "Intelligent Security System V1.0" project, the learner not only mastered specific technologies such as cloud–edge–device collaborative architecture, model Light weighting, database design, and front-end/back-end separation but, more importantly, acquired a transferable project-based learning methodology. This methodology starts with a real-world problem, establishes a knowledge map through literature review, drives deep learning through iterative debugging, consolidates knowledge outcomes through documentation, and ultimately forms a systematic thinking framework for addressing complex engineering problems. Looking ahead, the learner plans to pursue further work in federated learning, multimodal fusion, engineering robustness, and open-source community contributions. The value of project-based learning will permeate the entire professional career, for technologies may become obsolete, but the ability to solve problems, the methods of self-directed learning, and the courage to face uncertainty will never become outdated. It is hoped that the five-phase process, the five learning principles, and the typical difficulty coping strategies articulated in this paper will provide useful references for learners and educators in other engineering domains. Future research will further quantify the impact of project-based learning on learning outcomes—for example, through pre-/post-test comparisons and learning log analyses—to provide a more robust evidence base for engineering education reform.

Acknowledgements

This article is supported by "Xi'an Technology University's National-level College Students' Innovation and Entrepreneurship Training Program in 2025 (Project Number: 202510702031)".

References

[1] JIAN Yifan, SUN Huaiying, CHEN Liqiong. A multi-object face recognition method based on convolutional neural networks[J]. Journal of Applied Technology, 2026, 26(01): 55-62.

[2] YUE Ye, WEN Ruiping, WANG Chuanlong. Face recognition algorithm with feature-aware convolutional neural networks[J]. Chinese Journal of Engineering Mathematics, 2024, 41(03): 410-420.

[3] SU Xueping, SUN Dandan, LI Yunhong, et al. Masked face recognition algorithm combining multi-view features[J]. Journal of Northwest University (Natural Science Edition), 2025, 55(02): 286-296.

[4] LU Lidan, XIA Haiying, TAN Yumei, et al. Attention-guided joint learning of local features for facial expression recognition[J]. Journal of Image and Graphics, 2024, 29(08): 2377-2387.

[5] XU Qishen, MI Jinying, QIAN Lei, et al. A low-resource, high-efficiency hybrid-mode face recognition design based on FPGA[J/OL]. Telecommunications Science, 2025.

[6] LAVE, J., & WENGER, E. (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge University Press.

[7] HATTIE, J. (2008). Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement. Routledge.

[8] SWELLER, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257-285.

[9] FLAVELL, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. American Psychologist, 34(10), 906-911.

[10] KOLB, D. A. (1984). Experiential Learning: Experience as the Source of Learning and Development. Prentice-Hall.

[11] THOMAS, J. W. (2000). A review of research on project-based learning. Autodesk Foundation.

[12] BLUMENFELD, P. C., et al. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. Educational Psychologist, 26(3-4), 369-398.

[13] ZHANG Hua. The essence and implementation pathways of project-based learning[J]. Global Education Outlook, 2019, 48(10): 22-33.

[14] ZHONG Qiquan. Curriculum development based on core competencies: Challenges and issues[J]. Global Education Outlook, 2016, 45(01): 3-25.