Undergraduate Graduation Design Practice in Computer Science and Technology

Yuxuan Dong[1, a *]

[1]Xi'an Technological University, Xi'an, China

[a]2174207343@qq.com

Abstract. This paper takes the undergraduate graduation design of the Computer Science and Technology major as the research object, and fully records the entire process of the project "Design and Implementation of a Rapid Goods Recognition System for Cabinets Based on Deep Learning" from initiation to implementation. It focuses on analyzing the key actions and core gains in the stages of topic selection and decision-making, technical preparation, engineering practice, and reflection and summary. In the topic selection stage, it breaks through the limitations of traditional development directions and finally selects the uncontacted deep learning field among small program development, conventional system design, and deep learning applications. In the technical preparation stage, it builds a theoretical framework for target detection and system development based on pre-graduate study during the winter vacation and literature review. In the engineering practice stage, aiming at the problem of insufficient computing power of personal equipment, it systematically explores server rental and selection schemes, and overcomes a series of technical difficulties in environment setup, code reproduction, and function integration. Finally, a cabinet goods recognition system with image detection, video recognition, and result export functions is completed. The research confirms that this graduation design not only realizes the cross-field integration of professional knowledge but also cultivates the abilities of independent learning, problem diagnosis, and engineering implementation. It provides a reusable graduation design practice paradigm for computer major students and helps them efficiently complete the core tasks in the final stage of their studies.

Keywords: Undergraduate Graduation; Deep Learning; Cabinet Goods Recognition; Server Configuration; YOLOv7 Model

1. Introduction

As the core practical link of the four-year academic studies of the Computer Science and Technology major, undergraduate graduation design is not only a comprehensive test of professional knowledge such as programming languages, algorithm principles, and system architecture but also an important link connecting classroom theory and industrial applications [1]. Different from disciplines that focus on physical manufacturing, such as mechanical engineering, the graduation design of the computer major has significant technical iteration characteristics ─ from traditional management system development to the application of artificial intelligence, the difference in topic selection directions directly determines the depth of practice and the dimension of ability growth.

At the beginning of the graduation design, I faced the common "technical route selection dilemma" of computer major students: on the one hand, I hoped to transform the already mastered programming languages such as Python and Java into practical applications; on the other hand, I was worried that the topic selection would fall into the inefficient misunderstanding of "repeated development". Initially, three alternative directions were formulated: the first is lightweight WeChat mini-program development, whose technology stack (Vue, WeChat Developer Tools) has been practiced in the course design, with low development difficulty but limited innovation value; the second is the traditional B/S architecture management system, which adopts the SpringBoot MySQL architecture, with a mature development process but difficult to reflect technical breakthroughs; the third is the deep learning-driven cabinet goods recognition application, which involves uncontacted fields such as computer vision and neural networks but is in line with the actual demand of "unmanned inventory" in the intelligent retail industry. This entanglement is essentially a game between the "safe choice in

https://doi.org/10.58244/joerai.263698

the comfort zone" and the "growth choice in the challenge zone" ― the former can ensure the timely completion of tasks, while the latter may bring unexpected ability improvement [2].

Finally, the deep learning direction is determined. The core reason lies in the practical value and technical integration of the cabinet goods recognition scenario: this scenario not only requires model training capabilities (the core of deep learning) but also requires system development capabilities (interface and function implementation), and also involves engineering issues such as data processing and hardware adaptation, which can comprehensively exercise the comprehensive quality of computer majors. Just as the law of "technology iteration precedes application implementation" in the computer field, actively embracing the unmastered technical field can make the graduation design an opportunity to break through the boundary of one's own abilities, rather than a simple application of knowledge [3].

2 Topic Selection Decision-Making and Preliminary Preparation

2.1 Topic Selection Balance

The Decision-Making Process of Breaking Through the Technical Comfort Zone. The core contradiction in the topic selection of the computer major's graduation design lies in the balance between "technical familiarity" and "practical innovation".

The three alternative directions I initially had each had obvious advantages and disadvantages:

1) Mini-program development: The advantage is that it is based on the WeChat ecosystem, with low difficulty in user interaction design, and relevant experience has been accumulated in the "campus information query" course design; the disadvantage is that the functions are mostly concentrated on data display and form submission, which is difficult to reflect the technical depth of the computer major, and the market has a saturation of similar applications, lacking practical application value.

2) Traditional management system: The advantage is that the development process is standardized, the link from demand analysis to deployment and launch is clear, and the function implementation can be completed quickly; the disadvantage is that the technology stack is relatively traditional, which cannot touch the current hot directions in the computer field (such as artificial intelligence and big data), and the graduation design results have limited support for future career development.

3) Deep learning-based cabinet recognition: The advantage is that it is in line with the pain points of the intelligent retail industry (low efficiency and high error rate of manual inventory), can integrate multi-dimensional technologies such as target detection models, data annotation, and system development, and has public datasets (such as VOC2007) and mature models (YOLO series) to reduce the entry threshold; the disadvantage is that it is necessary to learn knowledge such as neural network principles and model training from scratch, and the hardware computing power requirement may exceed the carrying capacity of personal equipment.

During the topic selection process, three in-depth communications with the supervisor played a key role in promotion. During the first communication, I put forward the concern of "lack of deep learning foundation", and the supervisor pointed out: "The core competitiveness of the computer major lies in the ability to quickly learn new technologies, and the graduation design should be a carrier to exercise this ability, rather than a simple application of knowledge"; the second communication focused on the feasibility of the scenario, and the supervisor suggested giving priority to the cabinet recognition direction with "easy access to data and open-source foundation for models" to avoid falling into the complex trap of "algorithm innovation"; the third communication determined the specific technical route, and clarified that YOLOv7 is used as the basic model, the core recognition function is realized first, and then the visual interface is expanded to ensure the practical logic of "first implementation, then optimization" [4].

Finally, the topic is determined as "Design and Implementation of a Rapid Goods Recognition System for Cabinets Based on Deep Learning". This choice not only avoids the inefficient dilemma of "repeated development" but also endows the graduation design with innovation through "cross-field technology integration". More importantly, it forces me to jump out of the limitation of "only being

https://doi.org/10.58244/joerai.263698

able to write basic code" and face the dual challenges of theory and engineering in the deep learning field.

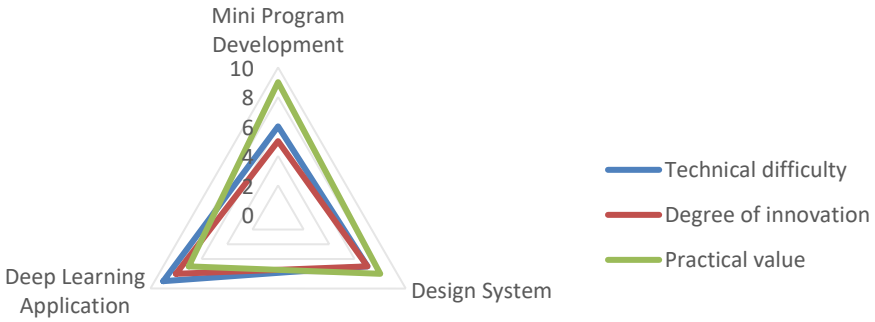

Figure 1. Trade-off Diagram for Topic Selection Decision-making

2.2 Pre-Graduation Study During Winter Vacation: Building a Systematic Technical Foundation

To reduce the technical resistance after the start of the graduation design, I carried out pre-graduate study for 40 days during the winter vacation, adopting a three-dimensional learning model of "video practice + literature review + tool rehearsal" to ensure that I could quickly enter the practical stage after the start of the new semester:

1) Theoretical introduction: Through the Bilibili series course "Deep Learning and Computer Vision Practice", I systematically learned the basic structure of neural networks, the principle of convolution operations, and the core process of target detection (such as the "end-to-end detection" mechanism of the YOLO model). I focused on understanding the complete link of "feature extraction → bounding box prediction → category judgment", compared the advantages and disadvantages of Faster R-CNN, SSD, and YOLO series models, and clarified the balanced advantages of YOLOv7 in "real-time performance" and "accuracy" — this is crucial for the "real-time inventory" demand of cabinet goods recognition [5].

2) Literature research: Using the school library database (CNKI, IEEE Xplore), I searched for literatures related to "cabinet goods recognition", "small target detection optimization", and "YOLO model improvement", and sorted out three core technical pain points in this field: complex background (shelf texture) interfering with recognition, missing detection of small-sized goods (such as chewing gum and toothpicks), and low accuracy in occluded scenarios (goods stacking), which provided directions for subsequent model optimization; at the same time, I referred to 10 undergraduate graduation design literatures to learn the writing logic of "theoretical derivation → experimental verification → system implementation" to avoid logical confusion in the later thesis writing.

3) Tool preview: I installed basic tools such as Python 3.8, PyCharm, and Anaconda on the personal computer in advance, practiced operations such as image reading, cropping, and format conversion of OpenCV through simple cases, was familiar with the environment creation and package management commands of Anaconda (such as conda create and conda install), and tried to use the LabelImg tool to complete a small amount of image annotation to avoid wasting time on basic tool operations in the later stage.

The greatest gain of the pre-graduate study is the establishment of a "technical knowledge graph": it is clear that the cabinet goods recognition system needs to be decomposed into three modules: "data collection and preprocessing → model training and optimization → system development and visualization", and each module is further subdivided into specific technical points (for example, data preprocessing needs to include annotation, enhancement, and division; model training needs to

involve parameter tuning and computing power adaptation). This structured cognition avoids the inefficient problem of "blind trial and error" in the subsequent practice [5].

3 Technical Breakthrough

3.1 Computing Power Bottleneck: Hardware Limitations of Personal Equipment

In the stage of preparing for model training, the typical obstacle of deep learning projects in the computer major was first encountered — insufficient computing power of personal equipment. The laptop I used was equipped with an NVIDIA MX350 graphics card with only 2GB of video memory, while the minimum video memory requirement for YOLOv7 model training is 4GB (lightweight YOLOv7-tiny), and the complete model requires more than 8GB of video memory. Initially, an attempt was made to reduce the model complexity: the lightweight YOLOv7-tiny was used, and the batch_size (number of batch training samples) was reduced to 4. As a result, it was found that the recognition accuracy of the model for small-sized goods dropped by 32%, which could not meet the core demand of "small packaged goods recognition" in the cabinet scenario at all [7].

This problem is not an individual case but a common dilemma for computer major students to carry out deep learning projects: personal equipment is limited by cost and portability and cannot meet the high computing power requirements of model training. If it cannot be solved, the entire graduation design will fall into the deadlock of "theoretical design cannot be implemented and verified" — the core value of deep learning projects lies in the "actual data verification of model effects", rather than just staying in code writing and theoretical derivation [8].

3.2 Server Rental: A Complete Exploration from Selection to Practice

Aiming at the problem of insufficient computing power, I systematically investigated the mainstream server rental platforms and configuration schemes, and formed a scientific selection strategy by comparing the three dimensions of "configuration flexibility", "price cost", and "ease of use":

1) Selection: General cloud servers such as Alibaba Cloud and Tencent Cloud were prioritized for exclusion because such platforms require manual configuration of deep learning environments (such as installing CUDA and PyTorch), which are not user-friendly for beginners; finally, deep learning-specific platforms such as AutoDL and Jilian AI Cloud were selected. These platforms are pre-installed with mainstream frameworks and dependency libraries and support "hourly billing", which is suitable for the limited budget of students (the average daily cost is controlled at 15-20 yuan).

2) Configuration selection: The core focuses on four parameters: GPU model, video memory size, CPU core number, and memory capacity. Initially, a Tesla T4 graphics card with 2GB video memory was tested, and the "CUDA out of memory" (video memory overflow) error occurred frequently during training; after switching to an RTX 3060 graphics card with 8GB video memory, it could support batch training with batch_size=16, and 100 rounds of training only took 8 hours (4 times faster than the T4 graphics card); at the same time, it was equipped with a 4-core CPU and 16GB memory to ensure that data loading (such as reading large datasets) and model reasoning do not get stuck, avoiding the resource waste of "GPU idle waiting for CPU".

3) Remote operation: Code writing and training monitoring were completed through the Jupyter Notebook provided by the platform, and SSH tools (such as PuTTY) were used to realize file transmission between the local and the server — for example, the locally annotated dataset was uploaded to the server through the SFTP protocol to avoid data loss caused by network fluctuations; during the training process, the model weight file (checkpoint) was saved every 10 rounds to prevent the loss of training results due to server expiration or connection interruption.

In the practice of server rental, two major problems were encountered: first, the "RTX 3090 with 16GB video memory" configuration was selected initially, with a daily cost of 50 yuan, which far exceeded the budget. Later, through the method of "first testing the feasibility with low configuration", it was determined that the RTX 3060 with 8GB video memory was the most cost-effective solution; second, the training was terminated due to remote connection interruption, which was later solved by adding "automatic resumption logic" in the code (reading the latest weight file to continue training). This experience made me deeply realize that computer engineering practice is not only "code

https://doi.org/10.58244/joerai.263698

implementation" but also includes hidden capabilities such as hardware resource planning and cost control — these capabilities are often ignored in classroom learning but are crucial in the actual project implementation [9].

4 System Development

4.1 Environment Setup: The Basic Engineering of Deep Learning Projects

Environment setup is the first technical threshold of deep learning projects. The core challenge is to ensure the compatibility of Python version, framework version, and dependency library version — any version mismatch may cause the model to fail to call the GPU or have abnormal functions. Initially, the installation command was directly copied by referring to the open-source project blog. As a result, due to the incompatibility between the PyTorch version and the CUDA version, the problem of "torch.cuda.is_available() returning False" occurred. Later, through the method of "step-by-step verification and layer-by-layer troubleshooting", a stable environment was set up in 3 days:

1) Version matching confirmation: The server was pre-installed with CUDA 11.6. According to the "CUDA version - framework version" correspondence table on the PyTorch official website, the versions of "torch==1.13.1+cu116" and "torchvision==0.14.1+cu116" were selected to avoid GPU call failure caused by incompatibility between the framework and the hardware interface.

2) Batch management of dependency libraries: A requirements.txt file was created to uniformly manage core dependency libraries, including opencv-python (image processing, version 4.6.0.66), pandas (data statistics, version 1.5.3), pyqt5 (interface development, version 1.5.7), and ultralytics (YOLOv7 implementation, version 8.0.12). The command "pip install -r requirements.txt" was used for batch installation to reduce the risk of version conflicts caused by manual installation.

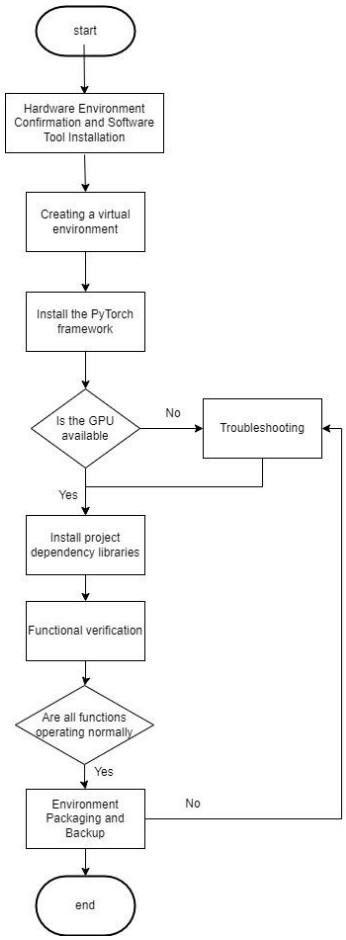

Figure 2. Environment Setup Flowchart

3) Functional verification test: Three sections of test code were written to verify the stability of the environment: the first was GPU availability test (printing torch.cuda.device_count() to confirm device recognition); the second was image processing test (reading and displaying a cabinet goods image with OpenCV); the third was model loading test (loading the pre-trained YOLOv7 weight file to infer the test image). Formal development was carried out only after ensuring that there were no abnormalities in each link.

During the environment setup process, problems such as "OpenCV failing to read images with Chinese paths" and "PyQt5 interface displaying garbled characters" were also solved: the former replaced cv2.imread() with the method of "np.fromfile() reading binary data + cv2.imdecode() decoding"; the latter solved the problem by setting the Chinese font (such as "Microsoft YaHei") of "QtGui.QFont". These seemingly trivial problems are actually important practices of the "compatibility thinking" in computer system development ― the stability of engineering projects often depends on the ability to handle detailed problems [10].

4.2 Code Reproduction: The Technical Transition from Imitation to Understanding

Code reproduction is the core link of deep learning projects. The purpose is to lay a foundation for subsequent model optimization and function integration on the basis of understanding the logic of open-source projects. I selected the open-source project "YOLOv7 Commodity Detection" with more than 5k stars on GitHub as the reproduction object, and completed the reproduction and problem troubleshooting in 2 weeks according to the process of "data preparation → model training → inference verification":

1) Data preparation: A hybrid dataset of "public dataset + self-built dataset" was constructed: the public dataset selected commodity category images in VOC2007 (a total of 5000 images), and the self-built dataset was obtained by offline shooting of supermarket cabinets (including 8 categories of goods such as food and daily necessities, a total of 3000 images); the LabelImg tool was used for annotation in PASCAL VOC format, and the annotation content included the bounding box coordinates and category names of the goods (such as "beverages", "snacks", "daily necessities"); the dataset was divided into a training set (7200 images) and a verification set (800 images) in a ratio of 9:1, and data enhancement technologies such as rotation (0°, 90°, 180°), horizontal flipping, and adding Gaussian noise were used to improve the generalization ability of the model to different scenarios.

2) Model training: Training parameters were configured based on the server's RTX 3060 graphics card: total rounds (epochs) of 100, batch size (batch_size) of 16, initial learning rate of 0.01, and the optimizer selected SGD (with momentum of 0.9); during the training process, TensorBoard was used to monitor key indicators in real time: training losses (obj loss, cls loss, box loss) and verification indicators (precision, recall, mAP@0.5) to ensure that the loss curve decreased steadily and the evaluation indicators continued to improve.

3) Inference verification: 800 images in the verification set were used to test the model effect, focusing on analyzing two types of problems: the missing detection rate of small-sized goods (such as chewing gum and toothpicks) and the false detection rate of occluded goods (such as stacked snack bags); a visualization tool (such as Matplotlib) was used to draw the detection results, and the commodity category, confidence, and bounding box were annotated to intuitively judge the performance shortcomings of the model.

The solution to two typical problems in the code reproduction process gave me a deeper understanding of deep learning engineering practice: first, "data annotation format errors" caused the model to fail to read training data. By comparing the standard VOC format XML files, problems such as "coordinate values exceeding the image resolution range" and "inconsistent case of category names" were corrected; second, "weight file damage after training interruption" was solved by adding the "checkpoint automatic saving" function in the training code (saving weights every 10 rounds and saving the optimizer state at the same time), ensuring that training could be resumed from the nearest node after interruption and avoiding the waste of computing power in the early stage [11]. It is worth noting that code reproduction is not "copying code", but understanding the model operation logic through the cycle of "modifying parameters → observing results → analyzing reasons" ― for example,

observing the fluctuation of the loss curve after adjusting the learning rate to master the influence law of "learning rate decay" on model convergence.

4.3 System Development: The Implementation Practice from Model to Application

After completing the model training, the system development stage was entered. The core goal was to transform "model inference capability" into "user-operable application functions" and finally realize the visualization and ease of use of "cabinet goods recognition". The system adopted a C/S architecture and built a desktop application based on PyQt5. The core functions included local image detection, real-time video detection, and recognition result export. The development process was divided into three modules:

1) Interface module: A GUI layout of "three areas and two columns" was designed — the left side was the "function selection area" (buttons for image upload and video access), the middle was the "result display area" (real-time display of detected images/videos, with commodity bounding boxes and category labels superimposed), and the right side was the "data list area" (displaying the name, confidence, and quantity statistics of recognized goods); the Qt Designer tool was used to complete the interface visualization design, and then PyUIC was used to convert the .ui file into Python code to ensure the interface was beautiful and the operation was intuitive.

2) Detection module: The YOLOv7 model was encapsulated into an independent "detection interface". The input was the image path or video stream frame, and the output was the commodity category, confidence, and bounding box coordinates; the detection logic was optimized for the cabinet scenario: an "image preprocessing" step (such as brightness equalization and denoising) was added to solve the problem of low recognition accuracy in strong/weak light environments; multi-threading technology was used to separate "image collection" and "model inference" to avoid interface lag caused by inference time consumption — for example, during video detection, the main thread was responsible for reading camera frame data, and the sub-thread was responsible for model inference. The results were transmitted through the signal-slot mechanism to ensure that the frame rate was stable above 20 FPS.

3) Export module: The recognition results could be exported in two formats: one was an Excel table (including commodity name, confidence, and detection time), which was convenient for users to count cabinet inventory; the other was annotated images/videos (superimposing detection boxes and category labels on the original images/videos), which was convenient for intuitively verifying the recognition effect; the export function was implemented through the pandas library (Excel generation) and OpenCV library (video writing), and at the same time, an "export progress bar" and "abnormal prompt" (such as non-existent file path) were added to improve the user experience.

The most challenging part of the system development stage was "multi-technical stack integration" — it was necessary to take into account model performance, interface response speed, and function stability at the same time. For example, initially, model inference and interface drawing were placed in the same thread, resulting in a video detection frame rate of only 5 FPS, which could not meet the real-time requirement; later, an independent "inference thread" and "interface refresh thread" were created, and thread isolation was realized using Qt's QThread class. At the same time, the signal-slot mechanism was used to safely transmit data, and the frame rate was increased to 22 FPS, which fully met the real-time inventory requirement of the cabinet scenario [12]. In addition, aiming at the problem of "false detection caused by dense stacking of goods", the model post-processing logic (such as adjusting the threshold of non-maximum suppression NMS) was optimized, and the false detection rate was reduced from 18% to 8%, further improving the practicality of the system.

5 Summary and Reflection on Graduation Design

5.1 Ability Improvement: Comprehensive Growth Beyond Technology Itself

Looking back on the entire graduation design process, the gains went far beyond "completing a operable system". More importantly, it cultivated the core literacy and engineering thinking of the computer major, which was specifically reflected in three dimensions:

https://doi.org/10.58244/joerai.263698

1) Independent learning ability: From "zero foundation" to mastering cross-field technologies such as YOLOv7 model training, server configuration, and PyQt5 development, an efficient learning path of "official documentation first → technical blog supplement → community question verification" was formed. For example, when learning remote server connection, first refer to the AutoDL official help documentation, then refer to the practical cases of CSDN blog, and finally solve the "SSH connection timeout" problem on Stack Overflow. This independent learning mode is far better than passive acceptance of knowledge.
2) Problem-solving ability: Facing a series of problems such as "insufficient computing power", "environment conflict", and "code bug", a standardized process of "phenomenon description → cause location → scheme verification → summary and precipitation" was gradually established. For example, when solving the problem of "model training loss not decreasing", first observe the trend of the loss curve through TensorBoard, then check three possible reasons: dataset annotation format, learning rate parameters, and model weight initialization, and finally locate the problem as "annotation coordinate error causing the model to fail to learn effective features". This structured problem diagnosis ability is the core of computer engineering practice.
3) Engineering thinking ability: The focus shifted from "only paying attention to code correctness" to "taking into account functions, performance, cost, and user experience". For example, when selecting a server, the balance between "configuration level" and "budget limit" was considered; when developing the system, "interface ease of use" and "detection real-time performance" were considered; when optimizing the model, the balance between "accuracy" and "inference speed" was considered — this kind of thinking of "finding the optimal solution under constraints" is the key transformation of the computer major from "student" to "engineer" [13].

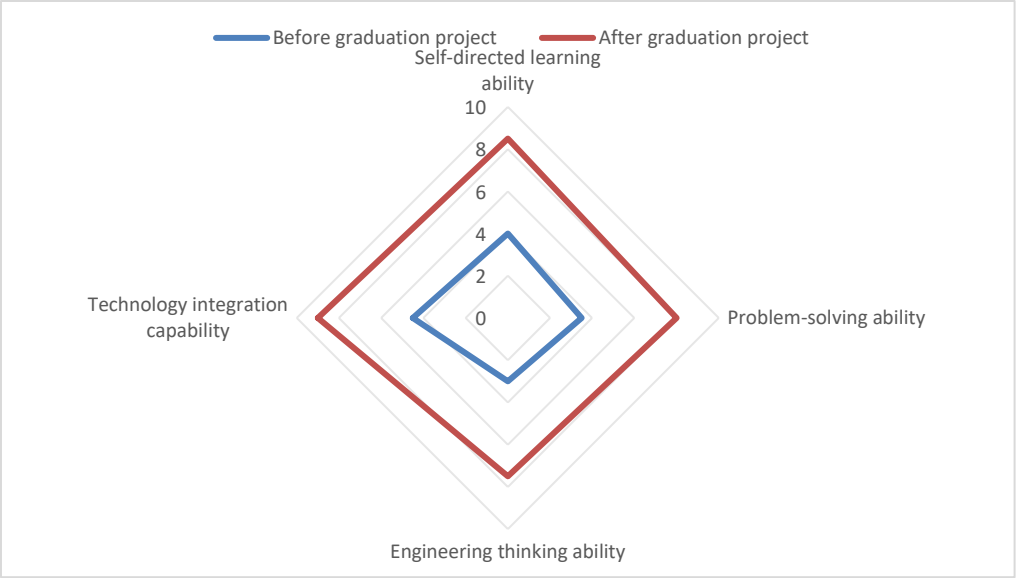

Figure 3. Competency Enhancement Radar Chart

5.2 Problems and Deficiencies

Although the system has realized the core functions, there are still three obvious deficiencies that need to be improved in subsequent optimizations:
First, the recognition accuracy of small-sized goods needs to be improved. The missing detection rate of goods with a size of less than 32×32 pixels (such as toothpicks and small packaged candies) is about 15%. The core reason is that the shallow feature extraction ability of the model is insufficient, and the underlying structure of the Feature Pyramid Network (FPN) for small targets has not been optimized;
Second, the system compatibility is limited. At present, it only supports Windows 10/11 systems and is not adapted to Linux or embedded devices (such as Raspberry Pi), which cannot meet the actual demand of "edge deployment" in the cabinet scenario, and the interface adaptation problem of screens with different resolutions is not considered;

https://doi.org/10.58244/joerai.263698

Third, there is a lack of user feedback iteration. During the system development process, no actual cabinet operators were invited to test, which may lead to problems such as "the operation process not conforming to actual usage habits" (such as single export format and no support for batch image detection), affecting the practical application value. These deficiencies are essentially caused by "insufficient engineering practice experience" and "insufficient scenario awareness" ― for example, the actual operation process of cabinet operators was not fully investigated, and functions were designed only based on their own understanding, leading to deviations between the system and actual needs.

5.3 Suggestions for Computer Major Graduation Design

Based on my own practice, five suggestions are put forward for the graduation design of junior students majoring in computer science and technology to help them complete the project efficiently:

1) "Dare to jump and try" in the topic selection stage: Priority should be given to directions with "technical challenges but open-source foundations" to avoid no improvement in ability due to "too familiar technology"; if a cross-field direction such as deep learning is selected, it is necessary to confirm that there are public datasets and mature models to reduce the entry threshold.

2) "Pre-start" in the preliminary preparation: Use winter vacation or spare time to learn core technologies in advance (such as model principles and development frameworks) instead of starting after the topic is determined; it is recommended to make a "technical knowledge graph" to clarify the learning objectives and outputs of each stage and avoid blind learning.

3) "Advance planning" for hardware problems: If the project involves deep learning, test the computing power of personal equipment as soon as possible. If it is insufficient, investigate server rental schemes in advance, compare the prices and ease of use of different platforms, and avoid delaying the project progress due to computing power problems.

4) "Iterative advancement" in the development process: Split the graduation design into small stages of "environment setup → core functions → optimization and iteration", and set clear delivery standards for each stage (such as successful model inference after environment setup), to avoid irreparable problems due to "last-minute rush"; at the same time, adhere to "developing while recording", and organize key problems and solutions into documents to facilitate later thesis writing.

5) "Proactive communication and collaboration" in the communication stage: Communicate with the supervisor regularly on the progress, especially when encountering technical bottlenecks, it is necessary to consult with "preliminary ideas and specific problems" instead of passively waiting for solutions; at the same time, a "technical exchange group" can be established with classmates to share environment configuration and code debugging experience and improve the efficiency of problem-solving [14].

6 Conclusion

The graduation design of "Design and Implementation of a Rapid Goods Recognition System for Cabinets Based on Deep Learning" is essentially a "cross-field integration practice" of professional knowledge in the Computer Science and Technology major ― from technical confusion during topic selection, to exploration and attempt of server configuration, to detailed breakthroughs in environment setup, and to function integration in system implementation. Each link is a reconstruction and improvement of "computer professional ability". This process made me deeply realize that the core competitiveness of the computer major is not only "the ability to write code" but also "the ability to learn new technologies", "the ability to solve practical problems", and "the ability to implement innovative applications".

For computer major students, the value of graduation design does not lie in "creating a perfect system" but in "breaking through their own limitations in the process" [6].From "being afraid of contacting deep learning" to "independently completing model training and system development", this leap not only brings growth at the technical level but also maturity in mentality ― understanding that in the computer technology industry with rapid iteration, "continuous learning" and "daring to challenge" are the long-term competitiveness. Just like the implementation process of the cabinet goods

https://doi.org/10.58244/joerai.263698

recognition system from "theoretical model" to "practical application", the undergraduate graduation design also completes the role transition from "classroom student" to "quasi-engineering and technical personnel", laying a solid practical foundation for subsequent career development.

References

[1] Brown A R ,Adams C J ,Ferner C , et al.Teaching parallel design patterns to undergraduates in computer science[C]//St. Olaf College, Northfield, MN, USA;;Calvin College, Grand Rapids, MI, USA;;University of North Carolina Wilmington, Wilmington, NC, USA;;Macalester College, Saint Paul, MN, USA;;University of North Carolina at Charlotte, Charlotte, NC, USA, 2014:

[2] Prajish P ,Sridhar I .VeriSIM: A model-based learning pedagogy for fostering software design evaluation skills in computer science undergraduates[J].Research and Practice in Technology Enhanced Learning,2022,17(1).

[3] Shi Y ,Huang Q ,Lyu J , et al.Progress of MRI-based radiomics and deep learning for predicting the prognosis of locally advanced rectal cancer (Review).[J].Oncology letters,2025,30(5):536.

[4] Bhadra R ,Chowdhury S ,Roy A , et al.QaPRExt: a unified deep learning framework for quality-aware PPG-derived respiration signal extraction for personalized healthcare[J].Measurement Science and Technology,2025,36(10):106114-106114.

[5] Ziaee A ,Suter G .Multi-unit space function and space access element classification in apartment buildings using machine learning and graph deep learning[J].Journal of Building Engineering,2025,112113472-113472.

[6] Li N ,Wang Z ,Zhao R , et al.YOLO-PDC: algorithm for aluminum surface defect detection based on multiscale enhanced model of YOLOv7[J].Journal of Real-Time Image Processing,2025,22 (2):86-86.

[7] Xiaohui Y ,Hanhong T ,Xiaoyan L , et al.Design and Implementation of Goods Storage Cabinet Based on K210 Face Recognition[C],2023:

[8] Sihyung L .Reducing Complexity of Server Configuration through Public Cloud Storage[J].Electronics,2021,10(11):1277-1277.

[9] Pigaiani N ,Musile G ,Scott S K , et al.Post-mortem formation of ethanol: Is 1-propanol a reliable marker? A proof-of-concept study using an in vitro putrefactive environment setup.[J].Journal of forensic sciences,2024,69(3):974-985.

[10] Yang L ,Feng T ,Ping Z , et al.A novel in situ sample environment setup for combined small angle x-ray scattering (SAXS), wide-angle x-ray scattering (WAXS), and Fourier transform infrared spectrometer (FTIR) simultaneous measurement.[J].The Review of scientific instruments,2023,94(3):033103-033103.

[11] Shi R ,Li H ,Ma T , et al.YOLOv7-GPSS: a YOLOv7-based wire rope surface defect detection algorithm[J].Signal, Image and Video Processing,2025,19(11):949-949.

[12] Wang F ,Song C .YOLO-ARM: An enhanced YOLOv7 framework with adaptive attention receptive module for high-precision robotic vision object detection[J].Alexandria Engineering Journal,2025,1291326-1339.

[13] Liu C Y ,Lee T W ,Yin C C , et al.Dual-Model YOLOv7-Based Image Recognition System to Simplify Data Entry for the Elderly in Mobile Health.[J].Studies in health technology and informatics,2025,3291860-1861.

[14] Zhao X ,Guo F ,Wang Y , et al.CEM-YOLO: Defect Detection in Large Hydropower Station Water Conveyance Pipelines Based on YOLOv7[J].Journal of Physics: Conference Series,2025, 2988(1):012012-012012.