The implementation and application verification of an intelligent recognition system oriented toward edge computing

Ruonan Wang, Mingrui Lai, Yan Zhang*

Guangzhou Institute of Science and Technology, Guangzhou 510540, China
**Corresponding author: 1434076937@qq.com*

Abstract: *With the deep integration of the Internet of Things and artificial intelligence technologies, intelligent recognition systems face severe challenges such as resource constraints and dynamic environmental changes in edge computing scenarios. This paper conducts research on the implementation of intelligent recognition systems for edge computing and proposes a distributed computing architecture featuring end-edge-cloud collaboration. This architecture achieves load balancing and efficient adaptation through task decoupling and heterogeneous resource abstraction. To address the challenges of model deployment, this study employs techniques such as structured pruning, knowledge distillation, and quantization-aware training to achieve model lightweighting, and further realizes online model deployment through dynamic computation graph optimization. Additionally, this paper designs a resource demand prediction model based on the characteristics of recognition tasks, a dynamic priority scheduling mechanism under latency constraints, and an adaptive update mechanism for data drift, thereby ensuring stable system operation under dynamically changing resources. The aforementioned research provides a comprehensive technical solution for the implementation of edge intelligent recognition systems, effectively balancing the trade-offs among recognition accuracy, resource consumption, and real-time responsiveness.*

Keywords: *edge computing; intelligent recognition; model lightweighting; resource scheduling; end-edge-cloud collaboration*

Introduction

Edge computing shifts computational power closer to data sources, thereby providing low-latency and high-bandwidth support for real-time intelligent recognition. However, edge devices have inherent limitations in terms of computing capacity, storage space, and energy supply, whereas intelligent recognition tasks impose high demands on computational resources and recognition accuracy. The contradiction between these two factors has become a key bottleneck restricting the development of edge intelligence. The traditional cloud-centric processing model struggles to meet real-time requirements, and the transmission of large amounts of data introduces privacy leakage risks. Therefore, research on intelligent recognition systems for edge computing holds significant theoretical value and practical importance. Although current research has made progress in areas such as model compression and edge scheduling, it still lacks a systematic collaborative optimization solution. This paper conducts research from three aspects: architecture collaboration, algorithm lightweighting, and resource scheduling. It proposes an end-edge-cloud collaborative architecture to achieve hierarchical data processing, reduces model complexity through structured pruning and quantization-aware training, and designs a dynamic priority scheduling mechanism along with an adaptive update mechanism to cope with environmental changes, aiming to provide comprehensive technical support for the implementation of edge intelligent recognition systems.

1. Intelligent Recognition Computing Architecture and Collaborative Mechanism for Edge Computing

1.1 End-Edge-Cloud Collaborative Distributed Intelligent Recognition Computing Architecture

An intelligent recognition system for edge computing must first address the issues of computing power stratification and data coordination. The end-edge-cloud collaborative architecture divides

recognition tasks into three levels: terminal collection, edge preprocessing, and cloud training, thereby forming a closed-loop mechanism for near-end data processing and iterative model updates. Terminal devices are responsible for the acquisition and preliminary filtering of raw images or video streams, edge nodes undertake inference tasks with high real-time requirements, and the cloud focuses on model training and update distribution for large-scale datasets. This layered architecture can effectively reduce the data transmission pressure on the core network, shorten the latency of recognition responses, and simultaneously ensure the privacy and security of local data.

At the specific implementation level, this architecture achieves seamless communication and collaborative work across different layers by defining unified interface protocols and data exchange formats. As the core hub of the architecture, edge nodes not only need to perform real-time recognition inference but are also responsible for filtering and compressing the data uploaded by terminals, transmitting only representative samples or abnormal data that is difficult to recognize back to the cloud. After receiving the data reported from the edge, the cloud leverages its powerful computing resources to perform incremental training or fine-tuning on the base model and then delivers the updated model parameters to the edge nodes. Through this architecture, the intelligent recognition system can maintain high performance and high availability in dynamically changing network environments while possessing the capability of elastic scaling to accommodate deployment requirements of different scales[1].

1.2 Load Balancing Strategy for Recognition Inference Based on Task Decoupling

Intelligent recognition tasks typically consist of multiple computational stages, such as image preprocessing, feature extraction, target classification, and post-processing, with each stage exhibiting significantly different demands for computing resources. Based on the concept of task decoupling, this strategy divides the complete recognition pipeline into several independent computing units and schedules each unit to different processing cores or computing nodes according to its characteristics. For example, this approach schedules computation-intensive convolution operations to GPUs or NPUs, while assigning post-processing tasks, which involve stronger control logic, to CPUs. This fine-grained task decoupling can fully exploit the heterogeneous computing potential of edge devices and prevent any single resource from becoming a system bottleneck.

On the basis of decoupling, the load balancing strategy needs to dynamically monitor the resource utilization rates and task queue lengths of each computing node, and it adjusts the allocation path of recognition tasks in real time. By introducing a distributed task scheduler, the system can reasonably distribute the decoupled subtasks to the optimal execution units according to network bandwidth, node loads, and task priorities. For recognition scenarios with high real-time requirements, the system can adopt a combination of data parallelism and model parallelism, allocating different data slices or model segments of the same recognition task to multiple edge nodes for parallel processing. This load balancing mechanism based on task decoupling not only improves the throughput of the recognition system but also enhances the system's robustness and stability when facing sudden request surges.

1.3 Abstraction of Heterogeneous Computing Resources and Adaptation Mechanism for Inference Tasks

Edge computing environments typically deploy multiple types of computing hardware, including ARM processors, GPUs, FPGAs, and various dedicated AI accelerator chips. These heterogeneous resources exhibit significant differences in instruction set architecture, memory models, and computing capabilities. To shield the complexity of the underlying hardware, this approach constructs a unified resource abstraction layer, which virtualizes physical computing resources into standardized computing units and provides a consistent calling interface to upper-layer applications. The resource abstraction layer is responsible for managing the loading of hardware drivers, the switching of computing contexts, and the allocation of memory space. This layer adopts a hardware abstraction layer design pattern and implements corresponding adapter modules for each hardware type. The abstraction layer exposes unified computing primitives to the outside, including operations such as tensor allocation, data transfer, and kernel execution, allowing upper-layer applications to leverage heterogeneous computing power without concerning themselves with underlying hardware details. Additionally, the abstraction layer incorporates a resource pooling management mechanism, which performs unified cataloging and status monitoring of computing resources, supports dynamic resource allocation and release, and thereby improves resource utilization efficiency[2].

Based on the resource abstraction layer, the inference task adaptation mechanism automatically selects the most suitable hardware backend to perform inference computation according to the network structure, operator types, and precision requirements of the recognition model. The adaptation process consists of two key steps: the first is operator-level hardware matching, which maps operations such as convolution, pooling, and activation in the model to the acceleration libraries or instruction sets of the corresponding hardware; the second is the compilation and optimization of the computational graph, which generates an efficient execution engine for the specific hardware. For hardware resources that support dynamic partitioning, the adaptation mechanism can also adjust the computing power ratio allocated to each task in real time, ensuring that high-priority recognition tasks receive sufficient resource guarantees. Through this abstraction and adaptation mechanism, the intelligent recognition system can fully leverage the heterogeneous computing power of edge nodes, thereby achieving an optimal balance between recognition performance and power consumption.

2. Recognition Algorithm Compression and Deployment Method Based on Model Lightweighting

2.1 Structured Pruning and Knowledge Distillation Technologies for Recognition Accuracy

In edge computing scenarios, the computational complexity and storage requirements of deep neural network models often exceed the capacity of edge devices, thereby necessitating model compression while maintaining recognition accuracy. Structured pruning technology directly reduces the computational load and number of parameters of a model by removing redundant convolutional kernels, channels, or layer structures within the neural network. Unlike unstructured pruning, structured pruning preserves the regularity of the network structure, which allows it to fully leverage the parallel computing capabilities of hardware platforms and achieve inference acceleration without requiring special hardware support after compression. The pruning process typically employs importance-based evaluation metrics, such as the scale factor of the BN layer, first-order gradient information, or feature map activation values, to perform quantitative assessment and identify redundant structures for removal. In specific implementations, this approach can adopt an iterative pruning strategy, where a small number of unimportant structures are pruned in each iteration followed by immediate fine-tuning for recovery, thereby avoiding a sharp decline in recognition accuracy caused by one-time large-scale pruning. Through this progressive optimization, the system can find the optimal balance between compression rate and recognition performance, ensuring that the lightweight model meets the requirements for edge deployment.

Knowledge distillation technology transfers the knowledge contained in a large-scale teacher model to a lightweight student model by constructing a teacher-student network framework. The teacher model typically possesses high recognition accuracy, but its inference speed struggles to meet the real-time requirements at the edge. In contrast, the student model features a compact network structure and can operate efficiently on edge devices. During the distillation process, the student model not only learns the teacher model's outputs for hard labels but also matches the teacher model's class probability distributions through soft labels, thereby acquiring richer feature representation capabilities. Combining structured pruning with knowledge distillation can achieve a better balance between compression rate and recognition accuracy, enabling the lightweight model to maintain recognition performance comparable to that of the original model in edge environments[3].

2.2 Quantization-Aware Training and Inference Acceleration Adapted for the Edge

Model quantization is a technique that converts floating-point parameters into low-bit integer representations, which can significantly reduce the model's memory footprint and computational latency, and it serves as a key method for edge deployment. Quantization-aware training simulates the errors introduced by quantization operations during the model training process, allowing the network parameters to adaptively adjust to accommodate low-bit representations. By inserting fake quantization nodes into the training graph, this approach uses quantized values for computation during the forward pass while maintaining floating-point gradient updates during the backward pass, ultimately obtaining model weights that are more robust to quantization errors. The fake quantization nodes simulate the quantization and dequantization processes, mapping floating-point data to discrete quantization levels while preserving the gradient propagation path. Compared with post-training quantization, this method achieves higher recognition accuracy at low bit widths, making it particularly suitable for fine-grained recognition tasks that are sensitive to accuracy. During the quantization-aware training process, the quantization range and clipping thresholds must be set appropriately to avoid convergence difficulties

caused by gradient mismatch.

In accordance with the characteristics of heterogeneous hardware at the edge, the quantization strategy must be aligned with the computing capabilities of the underlying computing units. Different hardware platforms exhibit varying support for quantization data types. For example, some mobile NPUs only support 8-bit symmetric quantization, whereas GPUs may support 16-bit floating-point or integer computations. The adaptation mechanism automatically selects the optimal quantization bit width and quantization scheme based on the target hardware's instruction set and memory bandwidth, and it employs a mixed-precision quantization strategy for sensitive layers within the network. After quantization, the model leverages underlying hardware acceleration libraries to map computation-intensive operations such as convolution and fully connected layers to dedicated matrix arithmetic units, thereby achieving an order-of-magnitude improvement in inference speed while reducing the power consumption overhead of edge devices.

2.3 Online Deployment of Recognition Models Through Dynamic Computational Graph Optimization

The network conditions and device resources in edge computing environments exhibit dynamic characteristics, making it difficult for statically deployed recognition models to adapt to real-time changes in requirements. Dynamic computational graph optimization technology performs runtime reconstruction of a model's computational graph based on the current available computing power, memory capacity, and input data complexity of the edge node. For recognition tasks in simple scenarios, the system can skip certain redundant network layers or employ shallow branches to perform fast inference. For complex scenarios, it dynamically introduces deeper network structures to ensure recognition accuracy. This adaptive mechanism achieves runtime selection of inference paths by pre-designing multi-branch network structures or conditional computation modules.

The online deployment process must address the issues of model version management and hot updates. When the cloud completes model optimization or updates, it needs to deliver incremental parameters or computational graph description files to edge nodes and perform dynamic model replacement without affecting the current service. The deployment framework adopts a storage format that separates the computational graph from the weight parameters, transmitting only the parameter data of the changed parts to reduce network transmission overhead. The edge-side deployment engine supports just-in-time compilation and execution of the computational graph, dynamically adjusting operator scheduling orders and memory reuse strategies according to the status of hardware resources, thereby ensuring the stable operation of the recognition model in resource-constrained environments. Through dynamic computational graph optimization, the intelligent recognition system can adapt to environmental changes online, achieving a dynamic balance between performance and resource consumption[4].

3. Resource Management and Scheduling Implementation for Edge Intelligent Recognition Systems

3.1 Resource Demand Prediction Model Based on Recognition Task Characteristics

The edge intelligent recognition system handles a variety of task types. Different recognition tasks exhibit significant differences in input data size, model complexity, and inference depth, which leads to dynamic variations in their demands for computing and memory resources. The resource demand prediction model constructs a mapping relationship for task resource consumption by extracting the meta-features of recognition tasks, including network structure depth, operator type distribution, input resolution, and expected output dimensions. The prediction process employs a regression model or a lightweight neural network to learn the correlation patterns between task features and metrics such as CPU usage, GPU utilization, memory bandwidth, and inference latency based on historical runtime data.

The prediction model needs to possess online learning capabilities to adapt to changes in the task flow within the edge environment. When a new type of recognition task is added to the system, the prediction model uses a small amount of actual runtime data to perform incremental updates on its parameters, correcting initial prediction deviations. For recognition tasks that arrive periodically, the model introduces time series analysis methods to capture periodic fluctuations and trend changes in resource demands. The prediction results are output as multidimensional resource demand vectors,

including metrics such as the required amount of computing resources, memory usage, and storage access frequency, thereby providing a quantitative basis for subsequent resource scheduling and task allocation. Through accurate resource demand prediction, the system can avoid waste caused by over-provisioning of resources while preventing the degradation of recognition service quality resulting from insufficient resources.

3.2 Dynamic Priority Scheduling for Recognition Tasks Under Latency Constraints

Real-time intelligent recognition applications typically impose strict constraints on task completion time, with different tasks exhibiting varying degrees of latency sensitivity and importance levels. The dynamic priority scheduling mechanism calculates the scheduling priority of each recognition task in real time based on the task's remaining completion deadline, its waiting time, and the resource demand prediction results. The priority function comprehensively considers both the urgency of the task and its resource occupancy efficiency, assigning higher priority to tasks that are approaching their deadlines while requiring relatively few resources, thereby ensuring their priority completion under limited resource conditions. The scheduler adopts a preemptive strategy, allowing it to interrupt a currently running low-priority task and save its context when a high-priority task arrives[5].

In scenarios with intense resource contention, the scheduling mechanism introduces an elastic resource allocation strategy, which dynamically adjusts the resource share allocated to each task based on the system load. For computation-intensive recognition tasks, the scheduler can dynamically migrate computational loads between CPUs and GPUs, leveraging the complementary characteristics of heterogeneous resources to alleviate the bottleneck pressure on any single resource. The scheduling decision process also considers data dependencies among tasks. For recognition task chains with pipeline dependencies, the mechanism employs a coordinated scheduling strategy to ensure synchronous execution of upstream and downstream tasks, thereby avoiding additional latency caused by waiting for data. Through dynamic priority scheduling under latency constraints, the edge intelligent recognition system can meet the real-time requirements of diverse tasks in resource-constrained environments.

3.3 Adaptive Update Mechanism for Recognition Services Facing Data Drift

The distribution of input data in edge environments changes over time, which leads to a gradual decline in the performance of deployed recognition models. This phenomenon is referred to as data drift. The adaptive update mechanism for data drift first requires the establishment of a drift detection module, which identifies whether a significant change in data distribution has occurred by monitoring metrics such as model output confidence, feature distribution statistics, and the consistency of inference results. Drift detection employs statistical hypothesis testing or distance-based measurement methods. When the detection indicator exceeds a preset threshold, the system determines that the current data distribution deviates from the training set distribution and triggers the model update process[6].

The model update mechanism adopts a combination of online learning and incremental training, utilizing new sample data cached at edge nodes to quickly adapt the existing model. During the update process, the system freezes most parameters of the original model and fine-tunes only the final few layers or specific adaptation modules, thereby reducing computational overhead and the demand for training data. For resource-constrained edge nodes, the update mechanism supports differentiated downloading of model parameters, retrieving only the weight data for the updated portions from the cloud to lower network transmission load. After the update is completed, the system validates the recognition performance of the new model through A/B testing, ensuring that the updated model retains its original knowledge while adapting to the characteristics of the new data distribution. Through this adaptive update mechanism, the intelligent recognition system can maintain stable recognition accuracy in dynamic environments and extend the effective service life of the model on edge nodes.

Conclusion

This paper conducts systematic research on the implementation of intelligent recognition systems in edge computing environments from three aspects: computing architecture, model compression, and resource management. At the architectural level, this study designs a distributed computing architecture featuring end-edge-cloud collaboration, which improves system efficiency through task decoupling and heterogeneous resource adaptation. At the algorithmic level, this work integrates techniques such as

structured pruning, knowledge distillation, and quantization-aware training to achieve efficient lightweighting and online deployment of recognition models. At the resource management level, this research constructs a resource demand prediction model, a dynamic priority scheduling strategy, and an adaptive update mechanism for data drift, thereby ensuring system service quality in dynamic environments. The aforementioned research provides theoretical foundations and technical pathways for the practical deployment of edge intelligent recognition systems. Future work will focus on further exploration of more efficient automated compression techniques, cross-node collaborative learning mechanisms, and edge security and privacy protection, in order to promote the continuous development of edge intelligent recognition technology.

Fund Projects

A Research on the Condition Monitoring and Fault Pre-Diagnosis System for Electric Vehicle Charging Facilities Based on Lightweight Deep Learning, a General Program in Science and Engineering at Guangzhou Institute of Technology (Project No. YBL2025012).

References

[1] Wu, Y., et al. "Intelligent Security System Based on Facial and Multimodal Recognition." Intelligent Internet of Things Technology, vol. 58, no. 2, 2026, pp. 76-79.
[2] Wang, Y., Luo, H., and Zhao, X. "Classroom Situation Analysis System Based on Computer Vision." Automation and Information Engineering, vol. 46, no. 5, 2025, pp. 47-53.
[3] Ye, Q. "Design of Industrial Robot Recognition System Based on Computer Vision." Machinery Industry Standardization and Quality, no. 8, 2025, pp. 39-42.
[4] Li, Q. "Analysis of Computer Artificial Intelligence Recognition Technology." Software, vol. 46, no. 2, 2025, pp. 95-97.
[5] Lu, L. Research on Human Activity Recognition Methods Based on Smart Wearable Devices. 2024. Northwest Minzu University, MA thesis.
[6] Gao, W. Research on Workpiece Recognition and Positioning System Based on Machine Vision. 2023. Nanjing University of Information Science and Technology, MA thesis.