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 development of the Internet of Things and artificial intelligence, intelligent recognition systems have gradually appeared, but there are still serious problems of resource limitations and changes in the operating environment at the edge. Research on intelligent recognition at the edge has been carried out, and in this paper, a distributed computing architecture with full-stack cloud cooperation is put forward. The two are far apart, but they are not the same and have different purposes. To address the problem of model deployment, structured pruning, knowledge distillation and quantization-aware training are employed to reduce the size of the model, and then dynamic computation graph optimisation is used for online model deployment. Based on the attributes of recognition tasks, a model to predict resource demand and a dynamic-priority scheduling method with latency constraints have been developed, and an adaptive update method for data drift has also been added to ensure the normal operation of the system in the face of all sorts of changes in resource conditions. All-round technical solutions have been put forward in the above studies to solve the problems of recognition accuracy, resource utilisation and real-time operation effectively in the construction of edge intelligent recognition systems.*

Keywords: *Edge Computing, Intelligent Recognition, Model Lightweighting, Resource Scheduling, End-edge-cloud Collaboration*

Introduction

Edge computing is near the data source, so there will be a small delay, but the bandwidth for real-time intelligent recognition will be large. Edge devices have some deficiencies in computation power, storage and power supply, so they are not suitable for meeting the high demands of computation and recognition accuracy of intelligent recognition. The two factors are in conflict and have become the main obstacles to the development of edge intelligence. The old model of cloud computing cannot meet the demands of real-time processing, and the transmission of a large amount of data is also a problem of privacy. Therefore, it is worth learning about intelligent recognition systems at the edge in terms of theory and application. Some progress has been made in the fields of model compression and edge scheduling, but a general cooperative optimisation method has not yet been developed. The three contents of this paper are Architecture Collaboration, Algorithm Lightweighting and Resource Scheduling. Collaboration architecture of end-edge-cloud has been used to distribute the processing of data, and in order to reduce the size of the model, structural pruning and quantization-aware training have been applied; along with changes in the environment, dynamic priority scheduling and adaptive update mechanisms have also been added, and full technical support will be provided for the construction of edge intelligent recognition systems.

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

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

First, we need to solve the problem of various computing powers and data coordination at the edge. The three stages of recognition in the end-edge-cloud collaborative architecture are terminal collection, edge preprocessing and cloud training; thus, a closed-loop system for near-end data processing and iterative model updating has been constructed. Terminal devices are used to get the first image or video

data and perform the first filter; edge nodes are used for inference tasks that need to be done in real time, and the cloud is used to train models with a large amount of data and distribute updated models. A few levels up can reduce the amount of data that needs to be sent to the main network, decrease the delay of the recognition result, and at the same time protect the privacy and security of local data.

The interface protocol and data format at the level of the actual implementation will be the same for all, so all layers can work normally and communicate. At the core of this structure is the edge node, which needs to perform real-time recognition inference; it will also filter and compress the data sent by the terminals, and only representative samples or abnormal data that are difficult to recognise need to be sent back to the cloud. Obtain the data sent by the edge, use the strong computing power of the cloud to gradually train or fine-tune the base model, and then send the updated model parameters back to the edge node. The structure of the intelligent recognition system can be highly available, work well in all kinds of network conditions, and also be scaled according to different demands at various levels of deployment[1].

1.2 Load Balancing Strategy for Recognition Inference Based on Task Decoupling

All the stages of an intelligent recognition task are quite different in terms of the required computing power, for example, image preprocessing, feature extraction, target classification and post-processing. According to the concept of task decoupling, all the modules in the recognition pipeline are independent computing units, and each module can be carried out by a different processing core or computing node according to its characteristics. For example, some of the convolution calculations can be carried out by GPUs or NPUs, and the relatively simple control logic for post-processing can be managed by the CPU. To make better use of all the computing power at the edge and avoid bottlenecks in any single resource, we can split this fine-grained task.

Decoupling is used to distribute the balancing work, and at different times, according to changes in resource utilisation and task queue length of the various computing nodes, the distribution of recognition tasks is dynamically adjusted. A distributed task scheduler will be used to distribute the decoupled subtasks evenly among the various execution units according to network bandwidth, node load, task priority, and so on. For recognition scenarios that need to operate in real time, both data parallelism and model parallelism can be employed to split different parts of the data or different parts of the model, and several edge nodes can be used to process them in parallel. To expand the processing power of the recognition system and handle many tasks at the same time, task load balancing and decoupling will be employed.

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

The computing hardware for edge computing is very diverse; there are ARM processors, GPUs, FPGAs, many other special AI chips, and so on. All of the above have different instruction set architectures, memory models and computing powers. To hide the difficulty of the underlying hardware, a unified resource abstraction layer can be established in this way to virtualise the physical computing resources into standardised computing units and provide a uniform call interface for the upper-layer application. Load the hardware driver in the Resource Abstraction Layer, switch to the computation context, and then allocate a memory area. A Hardware Abstraction Layer Design Pattern is employed in this part, and there are adapters for all kinds of hardware. A set of uniform computing primitives is exposed to the outside by the abstraction layer, such as tensor allocation, data transfer, kernel execution; thus, all kinds of computing resources can be utilised by the upper-layer applications without needing to know the specific details of the hardware. A Pool Management Module has been added to the abstraction layer to centralise the cataloging and status tracking of computing resources, dynamically allocate and free resources, and improve the utilisation efficiency of resources[2].

A Resource Abstraction Layer will be introduced to automatically select an appropriate hardware backend for the adaptation mechanism of the inference task according to the network structure, operator type and precision requirements of the recognition model. The two stages of the adaptation process are as follows: First is the matching of hardware at the operator level; that is, operations such as convolution, pooling and activation in the model are mapped to acceleration libraries or instruction sets for the corresponding hardware. Second, the compilation and optimisation of the computation graph are carried out to build an efficient execution engine for the hardware. The hardware resources of dynamic partitioning can be employed to distribute the computation of all tasks, and urgent recognition

tasks will be guaranteed to have sufficient resources. Abstraction and adaptation can be used to make full use of all kinds of computing resources at the edge, and a good trade-off between recognition accuracy and power consumption can be achieved for an intelligent recognition system.

2. Compression and Deployment Methods for Recognition Algorithms Based on Model Lightening

2.1 Structured Pruning and Knowledge Distillation Methods for Recognition Accuracy

Edge devices are generally weak in computing power and have a small amount of memory; thus, the deep neural network models are often too large, and model compression needs to be carried out to maintain a high level of recognition accuracy. Structured pruning can reduce the computational cost and the number of parameters of a model by removing redundant convolutional kernels, channels, etc., directly in the neural network. Unstructured pruning is not the same as structured pruning; that is to say, it does not maintain the regular structure of the network and cannot be sped up by parallel computation on hardware that lacks special hardware support after compression. Generally speaking, pruning methods are based on all kinds of importance indicators, such as the scale factor of a BN layer, first-order gradient information or feature map activation values, to determine the redundancy of a structure and then remove it. In some cases, an iterative pruning method is employed; that is, a small number of irrelevant structures are removed in each round, and immediately after, fine-tuning is carried out to correct any loss in recognition accuracy caused by the previous large-scale pruning. Gradual improvement will help us find a good trade-off between the compression ratio and recognition accuracy for the small model at the edge.

Knowledge Distillation is employed to construct a teacher-student network and spread the knowledge of a large-scale teacher model in a small student model. The teacher model has a good recognition accuracy, but it is too slow for the real-time requirements at the edge. At the same time, the student model is a small network and can be used at the edge. At the same time, during distillation, the student model can learn the outputs of the teacher model for hard labels and use soft labels to match the class probability distributions of the teacher model and obtain rich feature representations. Structured pruning and knowledge distillation can be combined to obtain an optimal trade-off between the compression ratio and recognition accuracy, so the lightweight model will still have the recognition performance of the original model in an edge environment[3].

2.2 Quantization-Aware Training and Inference Acceleration at the Edge

Quantisation of a model is a way to convert floating-point parameters into low-bit integers; thus, both the size of the model and the computation delay are reduced, and it has been widely used for the deployment of models at the edge. Quantisation-aware training determines the amount of error the model will have due to quantisation during training, and then adjusts the network parameters based on the low-bit representation according to this information. Add fake quantization nodes to the training graph, and then use quantized values for computation in the forward pass and keep the gradient updates in floating-point format during the backward pass; thus, it is possible to obtain model weights that are more robust to quantization errors. Fake Quantisation Nodes are used to imitate the quantization and dequantisation operations, and at the same time, the path of gradient flow is maintained. It is relatively easy to achieve a small reduction in bit rate and keep the accuracy of fine-grained recognition results high compared to post-training quantization. At a certain time in Quantisation-Aware Training, the Quantisation Range and Clipping Threshold should be set to avoid convergence problems due to gradient mismatch.

Since the hardware at the edge is heterogeneous, the selection of the quantization method should be in line with the computing power of the corresponding hardware. All the Hardware Platforms can handle all kinds of quantization data. For instance, some mobile NPUs only support 8-bit symmetric quantization, but GPUs can carry out 16-bit floating-point or integer arithmetic. According to the instruction set and memory bandwidth of the target hardware, based on the adaptation mechanism, an appropriate quantization bit width and quantization scheme will be automatically selected, and mixed-precision quantization will be applied to the sensitive layers in the network. Quantization can be employed to use hardware-acceleration libraries for computation, so the computationally intensive parts of convolution and fully connected layers will be handled by special matrix arithmetic units, and thus both the inference speed and power consumption of edge devices can be reduced by several times.

2.3 Online Deployment of Recognition Models by Dynamic Computational Graph Optimization

The network conditions and device resources of the edge computing environment are not uniform; therefore, one recognition model will not be suitable for all the different requirements at different times. Dynamically, the computation graph of a model can be changed at runtime according to the current computing power, memory availability, complexity of the input data and other reasons at an edge node. To increase the inference speed of recognition tasks in simple environments, some of the redundant network layers can be omitted or a shallow branch can be used. For serious cases, we will further expand the network to increase the recognition rate. At that time, one of the network branches or conditional calculation modules will be chosen for the response.

The problems of model version control and hot updates need to be addressed in the online deployment process. After the cloud finishes model optimisation or update, it needs to send the incremental parameters or the description of the computational graph to the edge node and then dynamically replace the model without stopping the current service. The deployment framework has been optimised to save the computational graph separately from the weight parameters, and only the parameter data of the modified part needs to be sent over the network. The edge-side deployment engine will compile and run the computational graph just in time, and according to the current state of hardware resources, it will dynamically change the order of operator scheduling and memory reuse to ensure stable operation of the recognition model in resource-constrained environments. Dynamically change the computation graph based on changes in the environment at any time to achieve a good balance of performance and resource utilization[4].

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

3.1 Resource Demand Prediction Model Based on the Characteristics of Recognition Tasks

The first few are generally divided into a few categories. Recognition tasks have different scales and are at various stages of model complexity and inference depth, so their demands on computation and memory will also differ. Construct a map of the resource demand for a recognition task by extracting meta-features of the recognition task, such as the depth of the network structure, the distribution of operator types, input resolution, expected output dimensions, etc. A regression model or a shallow neural network is used in the prediction stage to learn the relationship between the features of a task and indices such as CPU utilisation, GPU utilisation, memory bandwidth and inference latency based on historical runtime data.

Periodically update the prediction model in light of any changes to the task flow at the edge. When a new kind of recognition problem is added to the system, a small amount of actual operation data is used to gradually update the parameters of the prediction model and correct the initial prediction errors. Periodic fluctuations and trends in the demand for resources over a certain period can be observed in time series analysis. The result of the prediction is a multi-dimensional resource demand vector; that is to say, it has attributes such as the required amount of computing resources, memory usage, frequency of storage access, etc., and provides a quantitative basis for the following scheduling of resources and tasks. Predicting the demand for resources accurately can help to avoid having too much idle money and ensure that there are sufficient staff available to provide good recognition services.

3.2 Dynamic Priority Scheduling for Recognition Tasks Under Latency Constraints

Generally speaking, the requirements for the completion time of the real-time intelligent recognition application are relatively high, and all the various tasks have different sensitivities to latency and importance. Dynamic Priority Scheduling will change the priority of recognition tasks according to several reasons, such as how much time is left before their deadlines, how long they have been waiting, and forecasts of resource demand. The weight of the urgent factor and the extent of resource use determine the priority of the function; that is to say, tasks with approaching deadlines and relatively low resource demands will be given higher priority to ensure that they can be completed on time given the limited resources. The scheduler is preemptive; that is to say, it can stop the execution of a low-priority task, save its context, and then start running a high-priority task[5].

When resources are scarce, the scheduler can distribute these few resources among all the tasks in the system more flexibly according to the current conditions of the system. A scheduler can distribute

the work of high-computation recognition problems among CPUs and GPUs according to their strengths automatically, and it will only be limited by the weakest link. Schedule the tasks according to the data dependency of the tasks. For recognition task chains that have pipeline dependencies, coordinated scheduling is employed to have the upstream and downstream tasks run in parallel and not wait for the data. Dynamic Priority Scheduling with Latency Constraints can be employed to meet all the different real-time demands of various kinds of tasks at the edge in resource-constrained environments.

3.3 Adaptive Update Mechanism for Recognition Services Under Data Drift

With the change in the distribution of the input data at the edge over time, the performance of the deployed recognition model has been declining gradually. The above is referred to as data drift. First, a drift detection module needs to be added to the adaptive update mechanism to determine whether there has been a significant change in the distribution of the data by way of indicators such as model output confidence, feature distribution statistics and consistency of inference results. Drift detection is a kind of statistical hypothesis testing or distance measure. If the detection indicator is greater than the upper bound, the system will assume that the current data distribution is different from that in the training set and begin updating the model[6].

The new mode of the model is online learning and incremental training; based on the new sample data stored at the edge node, the current model is updated promptly. Most of the parameters in the original model are fixed during the update process, and only the last few layers or some adaptation modules are fine-tuned; thus, both the computational cost and the required amount of training data are reduced. For resource-constrained edge nodes, only download the model parameters that have been updated from the cloud to reduce the amount of data transmitted over the network, and obtain only the weight data of the updated part. Update and then carry out an A/B test of the recognition performance of the new model to ensure that it still has the original knowledge and can adapt to the characteristics of the new data distribution. Given the changes in the environment and to extend the service life of the model at the edge, an adaptive update method will be used.

Conclusion

The three research directions on the application of intelligent recognition systems in edge computing in this paper are computing architecture, model compression and resource management. According to the end-edge-cloud model, a distributed computing architecture will be employed in this paper to distribute tasks and use all kinds of resources for good system efficiency. The algorithms in this paper are structured pruning, knowledge distillation and quantization-aware training; thus, it is possible to reduce the size of the recognition model and run it online. Resource Management has also been added to this paper, and given the changes in the environment, a model to forecast resource demand, a dynamic priority scheduling method, and an adaptive update mechanism for data drift have been introduced. The above studies have offered some theoretical support and technical basis for the application of edge intelligent recognition systems. Next, we will present some more efficient automatic compression methods, cross-node collaborative learning mechanisms and edge security and privacy protection 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.