Scholar Publishing Group
Machine Learning Theory and Practice
https://doi.org/10.38007/ML.2026.060102
ISSN 2790-0983 Vol. 6, Issue 1: 10-17

Research on Edge Computing Deep Neural Network Task Unloading Based on Resource Collaboration Framework and Multi Strategy Optimization

Hangjie Zheng

52 Skytop Street, San Jose, 95134, US

Keywords: Edge computing; Deep neural network; Task unloading; Resource collaboration; Multi strategy optimization

Abstract: In the era of the Internet of Things, the exponential growth of IoT devices and explosive expansion of data have driven deep neural networks (DNNs) to become the core of intelligent data processing. However, their high computational complexity conflicts with the performance and energy consumption limitations of terminal devices. Existing schemes have limitations such as high hardware cost, lightweight model accuracy loss, high cloud computing latency, and lack of single path collaboration in edge computing. This research proposes the edge computing resource collaboration framework CoDNN, which optimizes throughput by executing DNN tasks through multiple computing resource pipelines; Design a particle swarm optimization based partition offloading algorithm DBPSO for single task delay optimization, dynamically allocating resources to meet deadline constraints; Aiming at the scenario of multi terminal sharing heterogeneous resources, a multi task energy consumption optimization strategy and the Levy flight particle swarm algorithm DBLPSO-MD are proposed to synchronously achieve deadline constraints and minimize system energy consumption. Research has found that the DBPSO algorithm has better response time than existing strategies, and DBLPSO-MD has significant advantages in achieving deadlines and energy consumption when resources or tasks increase. Through resource collaboration framework and multi strategy optimization, this research has promoted the efficient execution and energy efficiency improvement of DNN tasks in edge computing scenarios, and provided an effective solution for intelligent applications of the Internet of Things.

1. Introduction

With the advancement of the Internet of Everything era, the Internet of Things [1] has become the core pillar of the Internet in the future. The current global number of connected devices is growing exponentially. According to authoritative institutions' predictions, the access volume of smart terminals will soar from 50 billion in 2020 to 150 billion in 2025, accompanied by an explosive daily increase in data volume, expected to reach 73.1 ZB by 2025. Against this backdrop, deep learning, with its outstanding performance in computer vision, natural language processing,

and other fields, has become a key technology for achieving intelligent data processing in the Internet of Things. However, the high computational complexity of deep neural networks conflicts sharply with the limited hardware performance, battery life, and cost of terminal devices, making efficient processing of massive amounts of data a core challenge. The existing solutions have multiple limitations: at the hardware level, although deep neural network accelerators can improve computing performance, they will increase costs and energy consumption; Although lightweight models can reduce computational pressure, they may lead to loss of accuracy. In terms of uninstallation strategy, traditional cloud computing faces high latency and privacy risks due to wide area network data transmission; Although edge computing reduces latency through distributed nodes, most studies only focus on end-to-end or end-to-end cloud single path collaboration, which does not fully exploit the computing power potential of distributed devices in edge environments, and does not consider the throughput optimization and real-time response requirements of continuous data streams of deep neural network tasks. In addition, the strategy for unified task offloading and energy optimization in multi terminal concurrent scenarios also needs to be improved. In order to overcome the above limitations, it is necessary to make full use of the distributed characteristics of heterogeneous computing resources in the edge computing environment to build a pipeline architecture for collaborative execution of deep neural network tasks. While reducing the response time of a single task, it is also necessary to realize the optimization of system energy consumption in a multi task scenario. Specifically, the goal is achieved through the design of resource collaboration framework, the development of single task unloading strategy, and the optimization of multi task unloading strategy.

2. Correlation theory

2.1 Overview of edge computing

As a distributed computing model, edge computing[2]originated from the deployment of CDN technology[3] in the 20th century, and was formally defined by Professor Shi Weisong in 2016. It aims to deploy computing and storage resources at the edge of the network, closer to the terminal device or data source, to meet the challenges of cloud computing. Compared with traditional cloud computing, edge computing significantly reduces network bandwidth pressure and transmission delay by reducing long-distance data transmission, improves the user experience of time sensitive applications (such as automatic driving, medical real-time diagnosis), and protects users' private data through localized processing to reduce the risk of cloud data leakage. Its general architecture adopts a three-layer model of "end edge cloud": the terminal layer is composed of resource constrained devices such as sensors and wearable devices, continuously generating physical world measurement data; The edge layer is located between the terminal and the cloud, including public edge servers (such as base stations and routers) and terminal devices with strong computing capabilities (such as smartphones and tablets), providing distributed computing resources and supporting task offloading and collaborative processing; The cloud computing layer serves as a powerful backing, handling complex tasks that cannot be completed by the edge layer, storing long-term data, and dynamically adjusting edge resource deployment. At present, edge computing has been widely used in video analysis, smart city, medical care and other fields. According to the prediction of authoritative institutions, more than 50% of IoT device data will be processed at the edge in the future, becoming an important support for IoT, intelligent manufacturing, 5G, artificial intelligence and other technologies, and promoting the innovation and development of various industries.

2.2 Overview of Computing Task Offloading Technology

The technology of computing task offloading [4]optimizes task processing latency, energy consumption, and cost by transferring computing tasks from terminal devices to other computing resources for execution, ensuring quality of service (QoS). The core process includes five steps: task division needs to ensure the complete functionality of subtasks; The uninstallation decision needs to comprehensively consider factors such as network latency, computing resource performance, energy consumption, cost, and user privacy; Task deployment assigns subtasks to target resources; Task processing needs to ensure execution quality and efficiency; If the result is returned, the calculation result needs to be transmitted back to the terminal. According to the uninstallation process, it can be divided into three categories: local computing is suitable for lightweight tasks or high-performance terminals, such as simple calculation of vehicle speed by onboard sensors; Complete uninstallation transfers all tasks to the cloud or edge nodes, such as remote rendering in cloud games; Partial uninstallation achieves collaborative execution between local and edge/cloud through task partitioning, such as in intelligent systems where terminals preprocess data and transmit it to edge nodes to reduce network traffic. The optimization of task offloading strategies often faces NP hard problems and requires finding approximate optimal solutions in polynomial time. The current mainstream methods include heuristic algorithms (such as greedy algorithm, genetic algorithm, particle swarm optimization), convex optimization (solving non convex problems by reconstructing linearization or continuous convex approximation to transform them into convex problems), and machine learning (using historical data to construct predictive models, such as the unloading algorithm combining deep Monte Carlo tree search with neural networks). These methods promote the efficient application of computing offload technology in edge computing environment by balancing task completion time, energy consumption and cost.

3. Research method

3.1 Practical background and problem analysis of enterprise overseas warehouse location selection

Deep neural networks (DNNs)[5]originated from the study of artificial neural networks and can be traced back to Frank Rosenblatt's perceptron model[6] proposed in 1967, which constructed mathematical models by simulating the cooperative mechanisms of biological neurons. Driven by the trend of deep learning in the 21st century, DNNs have achieved stronger feature extraction capabilities by increasing network layers and parameter sizes. For example, in the evolution from AlexNet to ResNet, memory requirements have increased from 2.12G to 16.20G, and computational complexity has jumped from 727MFLOPs to 11GFLOPs. DNN inference is based on forward propagation mechanism, where input data is processed layer by layer from the input layer, hidden layer to the output layer. The computational characteristics of each layer provide a basis for task partitioning. The fully connected layer achieves data mapping and classification through matrix vector multiplication; The convolutional layer uses convolutional kernel sliding for feature extraction, and the kernel size (such as 1×1, 3×3) affects the receptive field and computational complexity; The pooling layer compresses the feature map size through maximum/average pooling to reduce the number of parameters; The activation layer introduces nonlinear functions such as Sigmoid and ReLU to enhance the model's expressive ability. The DNN structure presents diversity, and linear models such as VGG-16 consist of convolutional, activation, and pooling layers connected in series to form feature extraction and classification modules; Nonlinear models such as GoogLeNet [7] fuse multi-scale features through Inception modules, while DarkNet uses residual blocks to solve gradient vanishing problems. At present, DNN has been widely used in image

recognition (CNN such as VGG, ResNet), speech recognition (RNN such as LSTM, GRU), object detection (such as SSD, FCN) and other fields. Its directed graph (DAG) structure supports branching and merging operations, adapting to complex task requirements.

3.2 Research on Pipeline Unloading and Collaborative Framework of DNN Computing Tasks in edge computing

The current deep neural network (DNN) computation task offloading methods mostly focus on optimizing the total time of subtasks after model partitioning, but such methods are based on the assumption of low system load and do not fully consider the processing efficiency of continuous data streams, which is inconsistent with actual scenarios. Taking the last mile logistics delivery system of unmanned aerial vehicles as an example, the system deploys deep learning based applications such as video analysis and facial recognition, which need to process continuous video frames or speech segments and other data streams. Its core requirement is to improve overall throughput rather than single frame processing delay. Inspired by computer instruction pipeline, the DNN model can be divided into multiple subtasks and offloaded to different computing resources (such as terminal devices and edge servers) to form an execution pipeline, utilizing the advantages of parallel processing to improve throughput and reduce average execution latency. For example, in the scenario of unmanned aerial vehicle receiver recognition, the local execution of GoogLeNet model requires 215ms/frame (4.65fps), while the ideal sampling rate needs to reach 14fps, which is difficult to meet the quality of service (QoS) requirements; Completely uninstalling to edge servers can reduce to 136ms/frame (7.35fps); After adopting a pipeline approach that combines local and edge servers, the average execution time was further reduced to 102ms/frame (9.8fps), significantly improving efficiency. Based on this, the DNN execution time prediction model is introduced, and a deep neural network oriented resource collaboration framework CoDNN in the edge computing environment is proposed to achieve effective collaboration of various computing tasks and optimize the quality of service of DNN applications.

3.3 Deep neural network collaborative inference pipeline mechanism and optimization

The deep neural network (DNN) collaborative inference pipeline significantly improves the processing efficiency of continuous data streams by decomposing repetitive sequential tasks into subtasks and allocating them to dedicated computing resources for parallel execution. In computer architecture, the pipeline cycle Δ t is the maximum processing time of each segment, and the total processing time T is defined by the formula:

$$T = \sum_{i=1}^{k} t_i + (n - 1) \times \Delta t \quad (1)$$

Wheret_i is the processing time of each segment, n is the number of tasks, and k is the number of pipeline segments. The throughput TT represents the number of tasks completed per unit time, calculated by the formula:

$$TT = 1/\Delta t \quad (2)$$

in an ideal state, the average task execution time tends to approach the pipeline cycle Δ t. In practical applications, the continuous input of DNN tasks (such as video frames and speech clips) needs to focus on overall throughput rather than single frame delay. By dividing the DNN model into multiple partitions layer by layer and assigning them to different devices (such as devices A, B, C), the input data flows through each computing node to complete inference, forming a collaborative pipeline. The partitioning strategy needs to consider the matching of communication time and computing power: selecting a network layer with a small amount of data can reduce the

transmission overhead between devices, while meeting deadline constraints and minimizing the use of computing resources. For example, if a partition with a large workload is offloaded to low bandwidth or low computing power devices, it will reduce overall efficiency. Therefore, the partition size needs to be adapted to the computing/communication capabilities of each resource to optimize pipeline throughput and reduce average execution time.

4. Results and discussion

4.1 Research on DNN Execution Time Prediction and Resource Collaboration Framework in edge computing

Deep neural network (DNN) execution time prediction is the core basis for achieving efficient task offloading. Taking the VGG-16 convolutional layer as an example (as shown in Table 1),

Table 1. Parameter Configuration Table for Convolutional Layers of Deep Neural Networks

Layer Name	Input Size	Kernel Size	Channels	Stride	Output Size
Conv Layer 1-1	(3, 3, 224)	(3, 3)	64	1	(64, 224, 224)
Conv Layer 1-2	(64, 224, 224)	(3, 3)	64	1	(64, 224, 224)
Conv Layer 2-1	(64, 112, 112)	(3, 3)	128	1	(128, 112, 112)
Conv Layer 2-2	(128, 112, 112)	(3, 3)	128	1	(128, 112, 112)
Conv Layer 3-1	(128, 56, 56)	(3, 3)	256	1	(256, 56, 56)
Conv Layer 3-2	(256, 56, 56)	(3, 3)	256	1	(256, 56, 56)
Conv Layer 3-3	(256, 56, 56)	(3, 3)	256	1	(256, 56, 56)
Conv Layer 4-1	(256, 28, 28)	(3, 3)	512	1	(512, 28, 28)
Conv Layer 4-2	(512, 28, 28)	(3, 3)	512	1	(512, 28, 28)
Conv Layer 4-3	(512, 28, 28)	(3, 3)	512	1	(512, 28, 28)
Conv Layer 5-1	(512, 14, 14)	(3, 3)	512	1	(512, 14, 14)
Conv Layer 5-2	(512, 14, 14)	(3, 3)	512	1	(512, 14, 14)
Conv Layer 5-3	(512, 14, 14)	(3, 3)	512	1	(512, 14, 14)

There are significant differences in execution time among different layers due to variations in input size, convolution kernel size, number of channels, and other parameters. To this end, a linear regression model is used to predict the execution time of each layer: the input of the convolutional layer is $Sin \times K^2 \times C \times K^2/S$, the pooling layer is $Sin \times K^2 \times C \times K^2/S$, the fully connected layer is Sin, and the activation layer is Sin × Sout, where Sin/Shout is the size of the input/output feature map, K is the width of the convolution kernel, C is the number of output channels, and S is the step size. The accuracy of model prediction is evaluated by the coefficient of determination R^2, with a convolutional layer of 0.86, pooling layer of 0.88, fully connected layer of 0.96, and activation layer of 0.99, meeting the prediction requirements. Based on this, a resource collaboration framework CoDNN for edge computing is proposed, which includes three modules: task analysis, monitoring and unloading. The task analysis module obtains the model structure (such as a complex network represented by DAG) through a DNN parser, and uses the GRAMI algorithm to mine frequent subgraphs (such as GoogLeNet's Inception module), abstracting them into a single layer to simplify the linear model; The monitoring module collects real-time data on computing resources, computing power, and bandwidth between devices; The uninstallation module combines QoS requirements and divides the DNN into multiple partitions using partitioning algorithms, assigning them to appropriate computing nodes to form an execution pipeline. This framework optimizes

DNN task response time and system energy consumption through multi resource collaboration, making it suitable for real-time scenarios such as drone logistics and video analysis.

4.2 Model experiment

This section focuses on the optimization of the unloading strategy of deep neural network (DNN) computing tasks in the edge computing environment. The core goal is to improve the efficiency of task execution through task division and multi resource collaboration on the premise of meeting deadline constraints. Taking the DNN model as an example, its computational tasks need to consider three factors: resource selection, load allocation, and communication delay balancing. Table 2 shows the input sizes of each layer of DNN (such as 5MB for the first layer, 12MB for the second layer, etc.), while Table 3 presents the differences in execution time of different computing resources (devices A/B/C) for each layer (such as device A requiring 1 second to execute the first layer, and device C requiring only 0.35 seconds).

Table 2. Execution Time of DNN Layers on Three Computing Resources

DNN Layer	A (1GHz)	B (1.5GHz)	C (2.9GHz)
1	1 s	0.75 s	0.35 s
2	2.5 s	1.875 s	0.875 s
3	1.5 s	1.125 s	0.525 s
4	2 s	1.5 s	0.7 s
5	1 s	0.75 s	0.35 s

Through the comparison of six uninstallation schemes, it was found that multi resource collaboration can significantly reduce the average execution time (from 8s to 2.215-2.642s) compared to local execution (scheme 1) or single resource uninstallation (scheme 2), verifying the key role of resource collaboration and task partitioning. To solve the problem of DNN partitioning and offloading, a deep neural network partitioning and offloading algorithm based on particle swarm optimization (DBPSO) was designed. This algorithm iteratively searches for the optimal offloading solution by updating particle position and velocity, and the fitness function comprehensively considers task execution time, deadline constraint penalties, and computational resource usage costs. The experiment used Matlab R2020b simulation to set up execution scenarios of four classic DNN models (VGG-16/19, GoogLeNet, DarkNet-53) on heterogeneous computing resources (terminal devices, edge servers, cloud servers). The comparison of results shows that DBPSO reduces the average response time by 9.6% -53.1% compared to Neurosurger, DNNoff and other strategies, especially in multi resource pipeline processing. By dynamically adjusting the computational load and communication delay, the optimal throughput under deadline constraints is achieved. Further experiments have shown that task deadlines and network bandwidth have a significant impact on algorithm performance: under strict deadlines, algorithms tend to use more resources to meet deadlines, while bandwidth improvement prompts algorithms to increase task partitioning times to utilize sufficient bandwidth, ultimately reducing resource usage while meeting constraints.

4.3 Effect analysis

This section focuses on the complex scenario of multi terminal devices sharing heterogeneous edge computing resources, and proposes an optimization strategy to reduce system energy consumption under deadline constraints. The research background stems from the practical need for

concurrent execution of multiple DNN tasks in scenarios such as smart homes and smart security, which require a balance between response time constraints and energy efficiency. Quantify task execution characteristics by constructing a time energy dual model: the time model focuses on the average inference time of tasks, covering partition computation time, transmission time, and pipeline execution characteristics; The energy consumption model integrates computation, transmission, and idle energy consumption, with the optimization goal of minimizing the total system energy consumption while meeting all task deadline constraints. To solve this multi-objective optimization problem, a Levi's Flight Particle Swarm Optimization (DBLPSO-MD) algorithm is designed[8], which enhances global search capability by integrating Levi's flight strategy, and combines traditional particle swarm velocity position updates with random long jump step size adjustment to improve population diversity and avoid local optima. The experimental verification shows that the algorithm leads in the achievement rate of deadline when computing resources increase, and the system energy consumption advantage is more significant with the increase of the number of task groups, especially in complex multi task scenarios, showing efficiency, providing an effective solution for collaborative unloading and energy consumption optimization of multiple DNN tasks in edge computing environment.

5. Conclusion

With the development of IoT technology, deep neural networks (DNNs) are increasingly widely used in the intelligent application of terminal devices. However, their computing requirements are high, terminal performance is limited, and battery capacity is limited, facing dual challenges of response time and system energy consumption. The existing solutions have limitations: model compression sacrifices accuracy for lightweight; Cloud computing offloading results in high latency due to wide area network transmission; Edge computing studies multi focus single inference, ignoring the system throughput requirements of continuous tasks. Therefore, inspired by computer instruction pipeline, this research proposes edge computing resource collaboration framework CoDNN, which executes DNN tasks through multi computing resource pipeline to optimize the overall throughput.On the basis of CoDNN, a particle swarm optimization based partition offloading algorithm[9] DBPSO is designed for single task delay optimization requirements. By dynamically allocating computing resources to meet deadline constraints, its response time is experimentally verified to be better than existing strategies; In response to the complex scenario of multiple terminals sharing heterogeneous resources[10], a multi task energy consumption optimization strategy and the Levy flying particle swarm algorithm DBLPSO-MD are further proposed to minimize the total system energy consumption while meeting the deadlines of each task. Experiments show that they have significant advantages when resources or tasks increase.Future research will focus on three aspects: building clusters of IoT devices and verifying algorithm effectiveness in real edge environments; Explore cost optimization strategies based on the type, quantity, and usage time of computing resources; Aiming at the dynamic change of network bandwidth, a dynamic unloading algorithm is designed to adjust task allocation in real time to maintain user QoS and promote the efficient execution and energy efficiency improvement of DNN tasks in edge computing scenarios.

References

[1] *Zhang X, Wang M, Zhu X, et al. Collaborative Edge-Cloud Data Transfer Optimization for Industrial Internet of Things. IEEE Transactions on Parallel and Distributed Systems, 2025.*

[2] *Zhu B, Niu L. A privacy-preserving federated learning scheme with homomorphic encryption and edge computing. Alexandria Engineering Journal, 2025, 118(000):11-20.*

[3] *Chung J M. Content Delivery Network (CDN) Technology. Emerging Metaverse XR and Video Multimedia Technologies, 2023:251-277.*
[4] *Ahmed M, Raza S, Ahmad H, et al. Deep reinforcement learning approach for multi-hop task offloading in vehicular edge computing. Engineering Science and Technology, an International Journal, 2024, 59(000).*
[5] *Hong, Y. (2025). Architecture Design and Performance Optimization of a Large-scale Online Simulation Platform for Business Decision-making. Advances in Computer and Communication, 6(4).*
[6] *Hong, Y. (2026). Research on Warehouse Capacity Optimization Methods Based on Predictive Modeling. Engineering Advances, 6(1).*
[7] *Jin Li. Performance Analysis of Efficient Microservice Architecture in the Financial Industry. Machine Learning Theory and Practice (2026), Vol. 6, Issue 1: 1-9.*
[8] *Yixian Jiang. Performance Optimization and Improvement of Advertising Machine Learning Platform Based on Distributed Systems. International Journal of Big Data Intelligent Technology (2026), Vol. 7, Issue 1: 9-17*
[9] *Bukun Ren. Multimodal Learning Method for Cross-Modal Data Alignment and Retrieval. International Journal of Multimedia Computing (2026), Vol. 7, Issue 1: 1-8.*
[10] *Zhengle Wei. Research on Innovative Design of Financial Derivatives and Market Risk Management Strategies. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 19-27*
[11] *Yuhan Zhou. Green Bonds and Sustainable Financing Models in Energy Finance. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 28-35*
[12] *Yilin Fu. Research on the Application of Innovative Financial Technologies in Capital Market Risk Management. Socio-Economic Statistics Research (2026), Vol. 7, Issue 1: 1-9*
[13] *Linwei Wu. Data Visualization and Decision Support Analysis Based on Tableau. Socio-Economic Statistics Research (2026), Vol. 7, Issue 1: 10-18*
[14] *Xinran Tu. Resource Allocation Optimization and Cost Saving Analysis Based on Data Mining. International Journal of Business Management and Economics and Trade (2026), Vol. 7, Issue 1: 1-9*
[15] *Wang, C. (2026). Research on the Control of Uncertainty Risks in Investment Decision-making by Financial Modeling.*