Journal of Computers Vol. 32 No. 4, 2021, pp. 211-224
doi:10.53106/199115992021083204017

Research on Cloud-edge Joint Task Inference Algorithm in Edge Intelligence

Yaping Zheng*

Shenshuo Railway Branch of Baoshen Railway Group, China Energy Investment, 719300, Shenmu Shanxi, China
wang4059018@126.com

Received 1 June 2021; Revised 1 July 2021; Accepted 8 August 2021

Abstract. With the advent of the era of Internet of things, edge computing is gradually becoming a new computing paradigm in the field of Internet of things. With the rapid development of artificial intelligence technology, edge intelligence (EI) will be the general trend. Compared with the traditional cloud computing mode, the edge computing mode with scattered and limited resources brings great challenges to the training reasoning, model deployment and resource allocation of artificial intelligence services. In this paper, the multi task reasoning scenario in edge intelligence is analyzed, and a multi task cloud edge joint reasoning optimization algorithm based on DNN model is proposed, which is modeled as the optimization problem of minimum average delay in multi task scenario. The binary genetic algorithm and the double nested optimization algorithm of augmented Lagrange algorithm are used to solve the problem.

Keywords: cloud edge alliance, deep learning, edge intelligence, task inference

1 Introduction

In recent years, with the popularity of mobile computing and Internet of things (IOT), hundreds of millions of mobile devices and IOT devices are connected to the Internet, thus generating billions of bytes of data at the edge of the network. Massive data transmission between terminal devices and cloud increases the burden of network bandwidth and consumes a large amount of equipment resources. The ultra long distance data transmission of cloud computing can not meet the demand of time delay sensitive services. Therefore, the traditional centralized computing model with cloud computing as the core has reached the bottleneck. For the exponential growth of a large number of data processing capacity of edge devices is limited. As a new example, edge computing is considered to be the best solution to solve these problems. Edge computing refers to the open platform deployed close to the data source, which integrates the core capabilities of computing, storage, network and application, and can provide nearby services for devices. It pushes computing tasks and services from the core of the network to the edge of the network. It can reduce the load of the cloud center server, relieve the pressure of the network core bandwidth, improve the level of service ability, protect the privacy of user data, reduce the power consumption of the device, and bring higher quality of service to users. Edge computing and cloud computing work together to provide a good solution to a series of existing problems in big data processing in the Internet of things era.

At the same time, with the improvement of hardware computing power and the breakthrough of deep learning algorithm, artificial intelligence (AI) applications and services have made rapid development. In the Internet of things and big data scenarios, mobile devices and Internet of things devices generate a large number of multimodal data. Artificial intelligence has the ability to quickly process massive data and analyze and extract the data, so the fusion of edge computing and AI is the general trend. Specifically, edge computing aims to coordinate a large number of edge devices to process the data generated nearby, while AI aims to process the data through task model. Therefore, AI can make use of the key

* Corresponding Author

infrastructure and massive data of edge computing, and edge computing can also be popularized in the rich application scenarios of AI applications.

As a result, a new interdisciplinary research direction, edge intelligence, has aroused great interest. Edge intelligence supports AI applications through the network edge side. Edge nodes calculate and analyze the data within the local physical scope, and train and update the algorithm model to realize the self-learning closed-loop, as shown in Fig. 1. Edge intelligence includes edge computing supported by various machine learning algorithms, which means that the algorithm model is transferred to the network edge. Edge intelligence technology makes use of the complementary advantages of computing locality of edge nodes and high computing performance of cloud server by coordinating the resources of edge node devices and cloud server. So as to achieve the purpose of significantly reducing the delay and energy consumption of machine learning model reasoning. It can be predicted that a series of AI applications related to mobile computing and Internet of things need a lot of computing and energy. These applications are sensitive to privacy and delay, and naturally keep the same idea with edge computing, which makes the research of edge intelligence become a hot topic today.

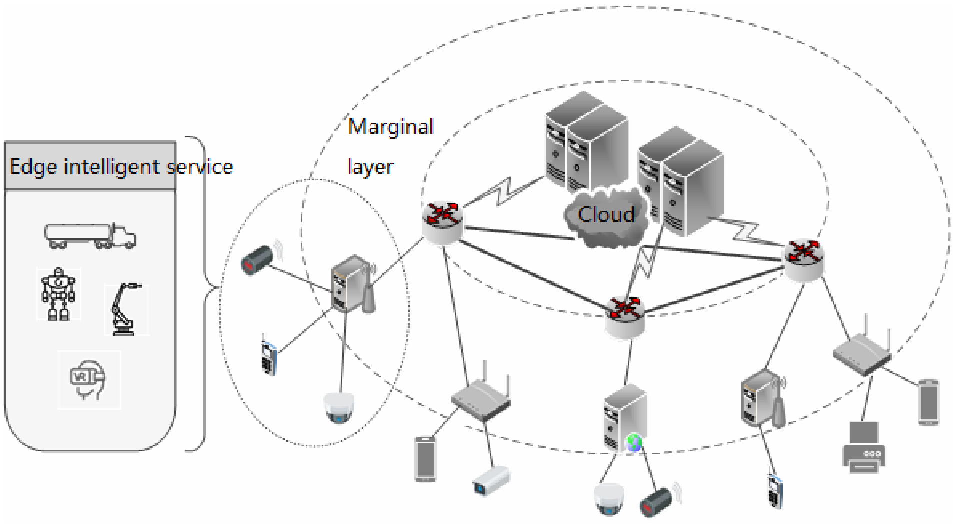

Fig. 1. Edge computing architecture

This paper studies the development status of edge intelligence, and then deeply studies the problem of task inference in edge intelligence, focusing on the resource optimization of task inference scenario in edge intelligence. Aiming at the modeling of minimizing the average delay of tasks, this paper proposes a DNN-based multitask cloud-edge joint inference algorithm (DMCJIA). Then, the unloading strategy of DNN model on edge nodes and the multi task resource allocation problem on cloud are described in detail. Finally, the simulation and experimental results are analyzed to prove the superiority of the model and algorithm.

2 Related Work

Low latency and energy-efficient task model offloading is very important to achieve high-quality EI service deployment in edge or terminal devices with limited algorithm and energy consumption. Now many technologies have been proposed. A pilot study [1] used model compression to reduce the complexity and resource requirements of the model, so as to realize inference on local devices, reduce response latency and reduce privacy issues. In order to reduce the execution pressure of EI application on terminal devices, model partition can be applied to unload the computing intensive part to the edge server or nearby mobile devices, so as to obtain better model reasoning performance. Model partitions can be divided into two types: partitions between servers and devices [2] and partitions between devices [3]. Both focus on latency, energy and privacy issues. At the same time, model partition can protect users' privacy by transferring some processed data instead of original data. In addition, edge caching [4] is a new method to speed up model execution, that is, to optimize the delay problem by caching task results. The core idea of edge caching is to cache and reuse task results at the network edge, such as image classification prediction, so as to reduce the query latency of EI applications. Input filtering [5] is also an effective method to accelerate model execution, especially for video analysis. The key idea of input filtering is to remove the non target frames of input data and avoid the redundant calculation of the model, so as to improve the task accuracy, shorten the task delay and reduce the energy consumption.

The core research problem of edge intelligence is how to deploy machine learning model to edge nodes, including model computing migration, model optimization of edge devices, collaborative scheduling between edge nodes and terminal devices, etc. As for edge intelligence, there are several core indicators, including the universality, scalability, low power consumption, hardware adaptability, task efficiency and so on. Because the research on edge intelligence is still in its infancy, task reasoning in edge intelligence is facing great challenges.

Due to the resource constrained characteristics of edge nodes, it is usually necessary to unload tasks to the cloud for joint computing. Model splitting is a promising technology. By splitting the DNN model used in artificial intelligence tasks, the DNN model is deployed in the edge node and the remote cloud server respectively. Its advantage is that it can reduce the total task delay and improve the quality of service. In the existing work of model splitting, researchers mainly focus on the static resource deployment and allocation and the optimization of DNN model itself, and seldom consider the task utility caused by delay in multi task scenario and the reasonable optimal allocation of cloud resources. These are the key issues to be considered in the future.

3 System Model

In this section, aiming at the task inference problem of DNN model splitting in multi node and multi task scenario, this paper takes minimizing the average task delay between nodes as the goal, establishes the queuing theory model for the task arrival on each node, and calculates the corresponding task delay according to the node resources and the different task processing delay under the cloud resource allocation. The task inference optimization problem is transformed into a nonlinear mixed integer programming problem. In order to solve this NP-hard problem, we decompose it into two sub problems: multi node model joint unloading optimization and cloud server computing resource allocation. We design a two-level optimization algorithm to solve this problem. The outer layer uses binary genetic algorithm to make the unloading decision of the task model, the inner layer relaxes the objective constraints, and uses Lagrange multiplier method to allocate the proportion of cloud server resources.

3.1 Problem Description

Artificial intelligence service is based on deep learning task, using DNN model as the core technology of task reasoning. When the DNN model reasoning task is executed, it usually needs to extract information from highly redundant datasets. If all task input data is sent to the cloud for processing, it will cause data transmission delay and network bandwidth pressure, which degrades service QoS and adds risk of privacy disclosure.

By deploying DNN model in edge node and performing reasoning task locally, edge computing can reduce the pressure of bandwidth and delay. DNN model reasoning is a computing intensive task. Because of the limited computing power and resources of edge nodes, it is unrealistic to rely on edge nodes totally.

Extending edge node reasoning to edge cloud joint reasoning is a reasonable solution. One of the key technologies is DNN model splitting technology [6], as shown in Fig. 2. Because DNN model has hierarchical structure characteristics, and the input and output between layers are relatively independent, if the DNN model is divided according to the network layer, the deployment and operation of the model are carried out separately, then the problem of insufficient computing resources of edge nodes can be solved. Specifically, the data intensive tasks with small amount of computation and large amount of data in the first half of the model are deployed to the edge nodes for processing, and the data intensive tasks with large amount of computation and small amount of data after feature extraction in the second half of the model are unloaded and deployed to the cloud for processing with low data transmission cost. The following mainly analyzes the cloud edge joint task reasoning scheme of multi node and multi task through task unloading in the edge intelligence scenario, which can well meet the system energy efficiency and improve the overall availability of the system.

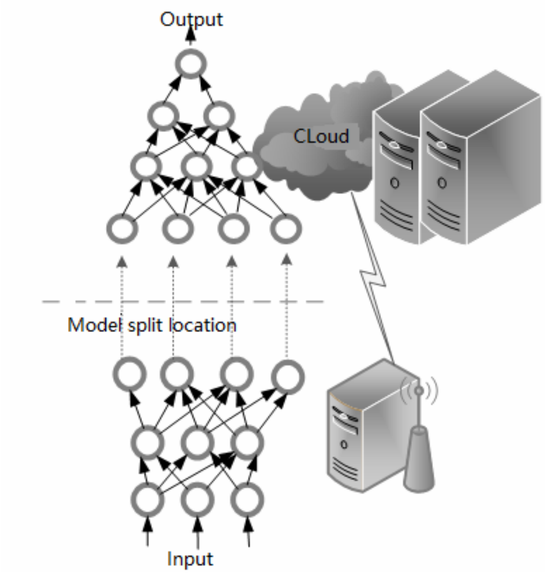

Fig. 2. Schematic diagram of DNN model separation

3.2 System Model

In the IOT scenario shown in Fig. 3, there is a central cloud server, multiple edge nodes and a large number of sensor devices. It is assumed that the edge node d is responsible for a class of tasks of a group of sensor devices in an area. All edge nodes can be represented as $d \in \boldsymbol{D}$, and the available computing power on the node d can be represented as $f_d \in \boldsymbol{F}$. The cloud server ε is connected with all edge nodes and provides edge intelligent services for them. The total computing capacity of the cloud is expressed as F_ε. Suppose that an edge node can only compute one kind of task at the same time, and multiple nodes can execute the same kind of task at the same time. Let the task type on the node d be $K_d \in \boldsymbol{K}$, and the arrival rates of different kinds of tasks obey Poisson distribution. The arrival rates is $\lambda_d \in \boldsymbol{\Lambda}$. The proportion of computing resources allocated to task K_d by the cloud server is $\delta_d \in \Delta$, meeting $\Sigma_{d \in D} \delta_d \leq 1$. The symbols and definitions in the system model are given in Table 2.

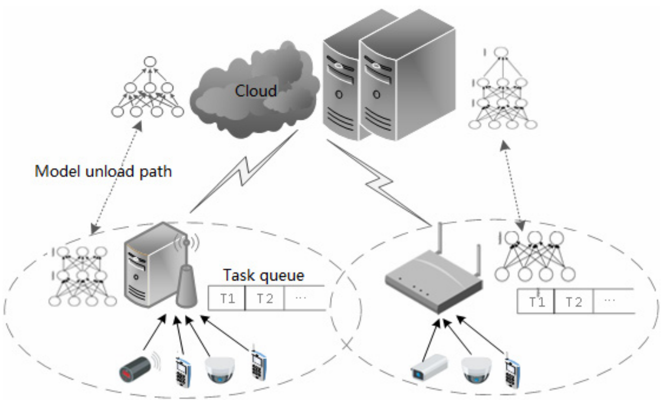

Fig. 3. Example of edge deployment

3.2.1 DNN Network Layer Computing Resource Requirements

DNN model has layered characteristics. The amount of calculation required by each layer model is relatively independent of the amount of data input and output between layers. The input data of each layer network is the output data of the upper layer network. According to the architecture of each DNN model, we can quantitatively calculate the amount of calculation of each layer model and the amount of input and output data of each layer in the process of single reasoning task execution. According to the derivation [7], the calculation amount of each layer network in the DNN model is jointly determined by the parameters of the layer, the size of input and output data, and the number of input and output data channels. Thus, the resource occupancy of each DNN model can be obtained [8], as shown in Table 1. The classic ResNet34 deep neural network [10] is analyzed. This DNN network model avoids gradient disappearance and degradation caused by too deep network through residual structure, and can achieve better training effect.

Table 1. Comparison of resource usage of different models

DNN network model	network layers	Total calculation	parameter
InceptionV3 [9]	46	6 GFLOPS	89MB
ResNet34 [10]	34	3.68GFLOPS	83MB
VGG19 [11]	19	19.6GFLOPS	548MB
AlexNet [12]	8	0.7GFLOPS	233MB

In the ResNet34 network structure, the amount of calculation required for each layer of DNN network and the delay required for each layer of network output data to be transmitted to the cloud under 1MHz bandwidth are calculated, as shown in Fig. 4. It can be seen that the amount of computation required for each layer of ResNet34 model remains basically unchanged, but the amount of output data decreases in turn. For edge cloud joint task offloading, with the task model offloading position moving backward, the total amount of computing data and computing delay of the model deployed by edge nodes in the execution of tasks increase linearly, but the delay of data output to the cloud decreases rapidly. Therefore, as the unloading position of the model moves back, the local computing delay of the edge node increases gradually, but the data transmission delay to the cloud decreases gradually. The data transmission delay varies greatly when unloading at different network layers, and the maximum difference can reach 2945 ms. This is basically consistent with the qualitative intuitive feeling of the model analysis. When the more parts of the model are unloaded to the edge node, the more the computing load of the edge node is, the less the transmission load is.

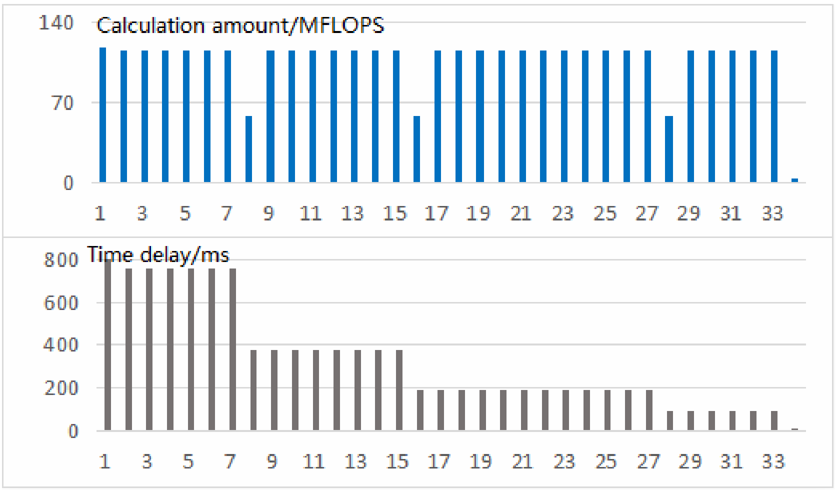

Fig. 4. Resource occupancy per layer of ResNet34

When the model is divided into edge nodes and cloud nodes for execution, it is assumed that the division position of DNN model is $\gamma \in \Gamma$, where Γ is the total number of layers of DNN model, that is to say, the front γ layer network model is unloaded on the edge nodes, and the rest of the network model is deployed on the cloud. Suppose that $C(\gamma)$ represents the total amount of computation that the edge node needs to perform when dividing the model at the unloading position γ, which is the sum of the computation of all layers before layer γ. At this time, the amount of computation to be performed in the cloud is $C - C(\gamma)$, where C represents the total amount of computation for a single reasoning task of the DNN model, i.e. $C = C(\max(\gamma))$. At the same time, at the best unloading location γ, assume that the amount of data output by the model is $M(\gamma)$, that is, the amount of data uploaded by a single task from the edge node to the cloud.

Table 2. Signs and definitions

Symbol	Definition
D	Set of edge nodes
ε	Cloud server
F	Set of computing power of edge nodes
F_ε	Cloud server computing power
$\boldsymbol{K}$	Collection of task types
Λ	Collection of task arrival rates
Δ	Collection of resource allocation ratios in the cloud
Γ	The set of DNN model partition positions
$C(\gamma)$	Total computation of DNN model front γ layer network
$M(\gamma)$	Output data of DNN model layer γ network
B	Link bandwidth from edge node to cloud

3.2.2 Delay Analysis

In the edge cloud joint reasoning decision, in order to minimize the average delay of all tasks, it is necessary to divide the DNN task model of each node. First, analyze the task delay on a single node. The total edge cloud delay of task execution mainly includes task calculation delay, queuing time and task data transmission delay. In this case, the delay needed for the inference result to return to the node is ignored, because the amount of data transmitted is very small, which is far less than the other delays mentioned above.

Assuming that the task arrival is a stationary process, obeying the Poisson distribution of arrival rate $\lambda(d)$, and the processing time of a single task is fixed, the M/D/1 Queuing theory model [13] can be used to model the queuing process. According to the resource computing power F_d of node d and the amount of computing in the first γ_d layer of the DNN model, it is calculated that the computing speed of the task on the edge node d is μ_d.

$$\mu_d = \frac{F_d}{C_d(\lambda_d)} \tag{3-1}$$

According to the steady-state distribution of M/D/1 queue, the average processing delay of tasks on edge node d is $T_{d,edge}$.

$$T_{d,edge} = \frac{\lambda_d}{2\mu_d(\mu_d - \lambda_d)} + \frac{1}{\mu_d} \tag{3-2}$$

The first term represents the average queuing delay of each task on edge node d, and the second term represents the calculation delay of each task. The sum of the two is the average processing delay of each task K_d on edge node. The transmission delay of edge node d transmitting data $M_d(\gamma_d)$ to the cloud server can be expressed as $T_{d,edge}$.

$$T_{d,trans} = \frac{M_d(\gamma_d)}{B_d} \tag{3-3}$$

Where B_d represents the uplink bandwidth from edge node d to the cloud, and the tasks on all edge nodes share the link bandwidth. On cloud server ε, the processing delay of task K_d is also divided into two parts: queuing delay and computing delay. Assuming that the proportion of computing resources allocated by the cloud server to the edge node d is δ_d, the computing speed of the cloud server for task

K_d is $\mu_{d,\varepsilon}$.

$$\mu_{d,\varepsilon} = \frac{\delta_d \cdot F_\varepsilon}{C_d - C_d(\gamma_d)} \tag{3-4}$$

According to the steady-state distribution of M/D/1 queue, the average processing delay of Task K_d on cloud server ε is $T_{d,cloud}$.

$$T_{d,cloud} = \frac{\lambda_d}{2\mu_{e,\varepsilon}(\mu_{d,\varepsilon} - \lambda_d)} + \frac{1}{\mu_{d,\varepsilon}} \tag{3-5}$$

The first represents the average queuing delay of task K_d on cloud server ε, and the second represents the computing delay of a single task on cloud server. The sum of the two represents the average processing delay of each task K_d in the cloud. Based on (3-3) (3-4) (3-5), for Task K_d on edge node dd, the total average edge cloud delay is T_d.

$$T_d = T_{d,edge} + T_{d,trans} + T_{d,cloud} \tag{3-6}$$

3.2.3 Multi-task Joint Optimization Modeling

Multiple edge nodes perform different kinds of tasks in parallel, and need to unload data to the cloud server at the same time. Due to the different resource computing power of each edge node, the DNN models used by different kinds of tasks may also be different. In the edge cloud joint reasoning, the optimal location γ for each node to unload the task model to the cloud server needs to be optimized.

On the cloud server, in addition to considering the bandwidth resources occupied by the model data transmission, because the edge nodes execute tasks synchronously and in parallel, each node needs to separate the computing resource environment in the cloud, so we also need to consider the allocation ratio δ_d of the total computing resources of the cloud to the tasks of different nodes to meet the parallel computing needs of each node. In order to minimize the average delay of tasks, the unloading process of each edge node to the cloud server task model is jointly optimized, and the following optimization equation is obtained.

$$\textbf{(P1)} \ \min T(\Gamma, \Delta) \tag{3-7}$$

$$= \Sigma_{d\in D} \frac{1}{|D|}\{\frac{\lambda_d}{2\mu_d(\mu_d - \lambda_d)} + \frac{1}{\mu_d} + \frac{M_d(\gamma_d)}{B_d} + \frac{1}{\mu_{d,\varepsilon}} + \frac{\lambda_d}{2\mu_{d,\varepsilon}(\mu_{d,\varepsilon} - \lambda_d)}\}$$

$$\begin{aligned} s.t.\quad & C_1 : \Sigma_{d\in D}\delta_d \le 1, \quad \forall_d \quad \delta_d \in R^+. \\ & C_2 : 0 \le \gamma_d \le \Gamma, \quad \forall_d \quad \gamma_d \in N^+. \\ & C_3 : \lambda_d \le \mu_d \le \mu_{d,\varepsilon}, \quad \forall_d \end{aligned}$$

C_1, C_2, C_3 are the constraint conditions of the optimization equation, constraint C_1 ensures that the resource allocation of each node is within the available range of the total computing capacity, constraint C_2 represents the unloading position of the task model on each edge node, constraint C_3 ensures the steady state of the task queue, and the processing speed of the cloud server is always faster than that of the edge node.

3.3 Algorithm Analysis

In problem P1, the joint optimization equation of edge cloud task offloading is obtained by modeling. Its physical meaning is the average delay of various tasks in all edge nodes. It can be seen that the objective function is a nonlinear function, including the independent variable constraints of constraint conditions C_1 and C_3 and the integer constraints of constraint condition C_2. Therefore, problem P1 is actually a nonlinear mixed integer programming problem, Since this kind of problem is Np-hard, the time complexity of solving it directly is unacceptable. In this section, the problem P1 is analyzed and summarized. By decomposing the problem P1 into two subproblems and designing algorithms to solve the two subproblems respectively, the approximate optimal solution of the original problem P1 is obtained.

3.3.1 Decomposition of Joint Optimization Problems

Because the problem P1 is difficult to solve directly, in order to solve the problem, the optimization method is used to decompose the problem. By analyzing the P1 expression and its constraints, we can find that the expression can be split. The three terms of $\frac{\lambda_d}{2\mu_d(\mu_d-\lambda_d)}+\frac{1}{\mu_d}+\frac{M_d(\lambda_d)}{B_d}$ are only related to the constraints C_2 and C_3, but not to C_1. the two terms of $\frac{1}{\mu_{d,\varepsilon}}+\frac{\lambda_d}{2\mu_{d,\varepsilon}(\mu_{d,\varepsilon}-\lambda_d)}$ are related to the constraints C_1, C_2 and C_3. In the first part, we extract the terms of P1 expression which are only related to the constraints C_2 and C_3, which can be expressed as P2.

$$\textbf{(P2)}\quad \min T^{(1)}(\Gamma)=\frac{\lambda_d}{2\mu_d(\mu_d-\lambda_d)}+\frac{1}{\mu_d}+\frac{M_d(\lambda_d)}{B_d} \qquad \textbf{(3-8)}$$
$$s.t.\quad C_2, C_3$$

By analyzing the physical meaning of P2 expression, it actually represents the joint optimization of the unloading position of the task model on all nodes. The independent variable is the unloading position $\gamma\in\Gamma$ of the task model on each node, and the objective function is the calculation delay and data transmission delay of all edge nodes. The second part extracts the remaining terms in P1 expression which are related to the constraints.

$$\textbf{(P3)}\quad \min T^{(2)}(\Gamma,\Delta)=\frac{1}{\mu_{d,\varepsilon}}+\frac{\lambda_d}{2\mu_{d,\varepsilon}(\mu_{d,\varepsilon}-\lambda_d)} \qquad \textbf{(3-9)}$$
$$s.t.\quad C_1, C_2, C_3$$

By analyzing the physical meaning of P3 expression, it actually represents the optimization of resource allocation for each node task in the cloud server. The independent variable is the proportion $\delta_d\in\Gamma$ of resource allocation of the cloud server and the unloading position $\gamma\in\Gamma$ of task model on each node. The objective function is the average computing delay of all tasks on the cloud server.

Since the independent variable Γ is a group of integer variables and the constraint C_2 is the limiting condition of its value range, the P2 expression is transformed into an integer programming problem. When the independent variable Δ is a group of decimals, the constraint C_1 is the limiting condition of its value range, and the objective function is a nonlinear function, the P3 expression is transformed into a constrained nonlinear programming problem. And because P3 expression also contains the independent variable Γ of P2, that is to say, the solution of P3 is related to the solution of P2. Therefore, a two-layer nested optimization algorithm is designed to solve the two problems after decomposition. First, the outer layer algorithm uses the heuristic genetic algorithm with good global search performance to solve the problem P2 to obtain the approximate optimal integer solution set Γ, then the inner layer algorithm substitutes the integer solution set Γ and uses the Lagrange optimization method to solve the planning subproblem P3 to obtain the solution set Δ, The global optimal solution is obtained by iterating the algorithm several times.

3.3.2 Implementation of Nested Algorithm

In this section, the sub problems are solved and designed based on the proposed bi level optimization algorithm. The first is the joint unloading subproblem P2 of the task model, which is solved by the outer heuristic genetic algorithm [14]. Its basic idea is to search the approximate global optimal solution by simulating the chromosome gene crossover and mutation in the process of genetic evolution. The basic steps are divided into coding, initialization, selection, crossover and mutation.

(1) Coding: because the independent variable $\gamma \in \Gamma$ of problem P2 is an integer variable, binary genetic algorithm can be used to code, that is, the unloading position $\gamma \in \Gamma$ of DNN model on all nodes can be digitally coded into binary gene array, so as to establish the sample space of individual genotype;

(2) Initialization: initialize the control parameters of the genetic algorithm, including the iteration number N of the genetic algorithm, the gene mutation probability P1 of the independent variable chromosome γ, the gene crossover probability P2 of the two variable chromosomes γ, and select the initial population $k \in K$ from the sample space, that is, all nodes randomly select the initial model unloading position $\gamma \in \Gamma$ to get an individual, Repeat this process to get the initial population of K individuals;

(3) Selection, crossover, mutation: in each cycle of the algorithm, the fitness of individuals in the population is calculated and evaluated to make individual selection, that is, the individuals obtained in this cycle are transformed into the unloading position of each node to calculate the value of the optimization function, and the results are evaluated to determine whether to retain or discard individuals, The new generation population is generated by gene crossover and mutation based on gene mutation probability and gene crossover probability, and the cycle is repeated until the termination condition is satisfied.

The individual fitness in step (3) of the algorithm can be transformed into the average delay of the task under the current solution value, so the fitness function is the negative value of the objective function of the original problem P1.

$$Fit(k) = -T(\Gamma, \Delta) \tag{3-10}$$

It can be found that solution set $\delta \in \Delta$ is needed to solve individual fitness, that is to say, inner layer algorithm is needed to solve problem P3 to get solution set $\delta \in \Delta$. For the subproblem P3, the value of the equation is determined by the independent variables Γ and Δ. In the outer layer algorithm, the solution set of Γ can be obtained from the sample space in each cycle, so the subproblem P3 is transformed into a nonlinear programming only related to the independent variable Δ. By expanding each item of problem P3 and moving the constraint conditions, the original problem can be further transformed into problem P4, and the following relationship can be obtained.

$$\begin{aligned}\textbf{(P4)}\quad \min T^{(2)}(\Delta) &= \Sigma_{d\in D}\frac{1}{|D|}\left(\frac{\lambda_d}{2\mu_{d,\varepsilon}(\mu_{d,\varepsilon}-\lambda_d)}+\frac{1}{\mu_{d,\varepsilon}}\right) \\ &= \Sigma_{d\in D}\frac{1}{|D|}\left(\frac{X^2(\delta_d)-1}{2\lambda_d(\delta_d)}\right) \\ s.t.\quad C_1 : X(\delta_d) &= \frac{\lambda_d}{\mu_{d,\varepsilon}}-1 = \frac{\lambda_d[C_d - C_d(\gamma_d)]}{\delta_d \cdot F_\varepsilon}-1<0 \\ C_2 : Y(\delta_d) &= \Sigma_{d\in D}\delta_d - 1 \le 0, \quad \forall_D \quad \delta_d \in R^+.\end{aligned} \tag{3-11}$$

For problem P4, the augmented Lagrange multiplier method can be used in the inner layer algorithm to solve the problem. Because the original problem is an inequality constraint problem and the constraint conditions are C_1 and C_2, the non negative relaxation variables s_x and s_y are introduced to transform the inequality constraint into equality constraint. Lagrange multiplier method is used to define two Lagrange multiplier vectors α and β as the coefficients of constraint conditions, and penalty factors ρ_x and ρ_y are introduced to it. The Lagrange function under unconstrained conditions is obtained as follows.

$$\begin{aligned}\textbf{(P5)}\quad \min L(\delta_d,\alpha,\beta,\rho) &= T^{(2)}(\delta_d)+\alpha[X(\delta_d)+s_x]+\beta[Y(\delta_d)+s_y] \\ &\quad +\frac{\rho_x}{2}\| X(\delta_d)+s_x \|^2+\frac{\rho_y}{2}\| Y(\delta_d)+s_y \|^2 \\ s.t.\quad & s_x, s_y > 0\end{aligned} \tag{3-12}$$

In the process of searching the optimal solution of the function, the current approximate optimal solution satisfying the condition of Lagrange multiplier method is solved by analytic method, then the values of Lagrange multiplier and penalty factor are changed by gradient rising method, and the iterative solution is repeated. Finally, the minimum value of the original Lagrange function is obtained, and the solution set of the original problem P3 is obtained by solving problem P5. $\delta_d \in \Delta(\forall d \in \boldsymbol{D})$.

The outer layer genetic algorithm can calculate the fitness function value of each individual according to the solution value δ_d obtained by the inner layer algorithm, so as to carry out individual evaluation and select a new generation of population, and obtain a new integer solution set $\gamma_d \in \boldsymbol{\Gamma}(\forall d \in \boldsymbol{D})$. Through the iteration cycle of the two-layer algorithm, until the result converges, the approximate optimal solution of the original problem P1 is finally obtained, $\delta^* \in \Delta$, $\gamma^* \in \boldsymbol{\Gamma}$. To sum up, DMCJIA (DNN-based Multitask Cloud-edge Joint Inference Algorithm) is proposed. According to the optimization problem solving method, the detailed description of DMCJIA is obtained, as shown in Table 3.

Table 3. DNN-based multitask cloud-edge joint inference algorithm

```
Algorithm 1. DMCJIA
1.  /* Outer layer algorithm */
2.  Begin:
3.  For node $\forall d \in \boldsymbol{D}$, calculate the calculation amount $C_d(\gamma_d)$ of each layer unloading and the output data amount $M_d(\gamma_d)$ of each layer;
4.  The integer variable $\gamma_d \in \boldsymbol{\Gamma}$ of unloading layers are binary coded to obtain the sample space of chromosome;
5.  Define the initial parameters: population size K, mutation probability P1, crossover probability P2 and iteration number N;
6.  Generate the initial population $k \in K$ randomly from the sample space;
7.  For epoch=1, 2, …, N:
8.  For each individual in the population, $\delta_d \in \Delta$ is obtained by the inner layer algorithm, and the fitness $Fit(k)$ is calculated;
9.  Keep the high fitness individuals and discard the low fitness individuals;
10. Individuals are crossed and mutated to be the next generation population;
11. epoch+1;
12. End for;
13. Output approximate optimal solution set $\delta^* \in \Delta$, $\gamma^* \in \boldsymbol{\Gamma}$;
14. End.

1.  /* Inner layer algorithm */
2.  Begin:
3.  Initialize relaxation variables $(s_x, s_y)$, penalty factors $(\rho_x, \rho_y)$, and Lagrange multipliers $(\alpha, \beta)$;
4.  Get the current unloading position $\gamma_d \in \boldsymbol{\Gamma}$ according to the outer layer algorithm, and establish Lagrange equation $L(\delta_d, \alpha, \beta, \rho)$;
5.  The derivation of $L(\delta_d, \alpha, \beta, \rho)$ is solved according to the condition KKT;
6.  The approximate optimal solution set $\delta_d \in \Delta$ is obtained by the gradient rising method;
7.  End.
```

It can be seen from Table 3 that when the available resources of the edge node and the cloud are known and the link state is determined, the unloading position of the DNN model on each edge node and the proportion of computing resources allocated by the cloud to the tasks on each edge node can be obtained in the initialization phase of the task running environment, and the task model on each edge node can be deployed accordingly, And the corresponding task model is unloaded to the cloud, and the cloud builds a parallel computing environment for each task according to the corresponding resource allocation proportion of the nodes, so as to realize the cloud side joint task reasoning of minimizing the average delay. In the next section, the performance of DMCJIA algorithm is analyzed in the simulation environment.

4 Scheme Assessment

In this section, we simulate the multi task reasoning scenario between the cloud server and a group of edge nodes with heterogeneous resources in the Internet of things environment, analyze the unloading process of the task model, evaluate the performance of the proposed dmcjia algorithm through simulation experiments, and analyze the experimental results.

4.1 Experimental Parameter Setting

In this experiment, in order to simulate the rich computing resources in the cloud and provide computing power for edge nodes as much as possible, four groups of computing devices equipped with Intel x86 high-performance processors are used as clusters to simulate the cloud server. Since the computing capacity of each group of computing devices in the cloud is about 100 GFLOPS, the total computing performance of the cloud server is about 400 GFLOPS. Due to the heterogeneous resource characteristics of these edge node devices, their computing power ranges from 5GFLOPS to 9GFLOPS. According to the experimental requirements, the number of edge nodes used in each group of experiments varies from 5 to 20.

In the experiment, it is assumed that there are three types of DNN reasoning tasks, all of which are artificial intelligence tasks of recognition and detection. The experimental data sets of Mnist, Cifar100 and ImageNet are used to simulate the data of three types of tasks, and the types of tasks are t_1, t_2 and t_3, respectively, as shown in Table 4. Each edge node randomly selects one of the three tasks as its own task K_d. during the operation of the system, each node is only responsible for this task. Suppose that one data in the data set is a task executed by the edge node. By making different nodes process different numbers of data in unit time, the arrival rate λ of the corresponding actual task in each node is simulated.

Table 4. Comparison of three data sets

Data set	Task type	Original size	Size
Mnist	t_1	28*28*1	60000
Cifar100	t_2	32*32*3	60000
ImageNet	t_3	224*224*3	1000000+

For the task model selection, different kinds of tasks have corresponding common task models in the actual scene. In the experiment, it is assumed that these three kinds of recognition and detection tasks use the same DNN model to perform reasoning tasks, and ResNet34 network, which is widely used in the field of computer vision, is selected as the task model, The relevant model parameters and output data have been analyzed in Fig. 4. The total calculation amount of the model is about 3.657GFLOPS, and the total parameter amount is about 21.8M. The models used in the simulation experiment have been trained well in advance, and the task accuracy is more than 95%. Through the proposed DMCJIA algorithm, the task model is deployed in each edge node and cloud to perform task reasoning. The relevant parameter settings of algorithm simulation are shown in Table 5.

Table 5. Simulation parameter setting

Parameter	Definition	Value
D	Number of edge nodes	{5~20}
F_d	Computing power of edge nodes	{5~9} GFLOPS
F_ε	Total computing power of cloud server	400GFLOPS
B_d	Cloud edge link bandwidth	{1~5} Mbps
λ_d	Mission arrival rate	{0.5,1,2}/s
Γ	Total layers of DNN model	34
N	Iteration times of genetic algorithm	32
K	Population size of genetic algorithm	50
P1	Variation probability	0.01
P2	Crossover probability	0.1

4.2 Result Analysis

Based on the above simulation environment, this paper analyzes the multi task DNN reasoning scenario, compares the performance of DMCJIA algorithm in different task conditions, analyzes the node cloud resource allocation of DMCJIA algorithm on heterogeneous nodes, and proposes some benchmark algorithms to compare the performance of different algorithms.

4.2.1 Iterative Convergence

We evaluate the iterative convergence of DMCJIA algorithm under different number of nodes, in which the arrival rate of each task is set to 1gs and the bandwidth of cloud side link is set to 1MHz. As shown in Fig. 5, the horizontal axis represents the number of iterations of DMCJIA algorithm, so that the total number of iterations is 32, and the vertical axis represents the average total delay of each task, including the sum of the delay of edge node, cloud and data transmission. It can be seen from the figure that under different number of edge nodes, the algorithm tends to converge to the average delay optimization, and the less the number of edge nodes, the less iterations the algorithm needs, and the faster the convergence speed. This is because with the increase of the number of edge nodes, the unloading position of each node model and the independent variables of resource allocation to nodes in the cloud increase linearly, which leads to the exponential increase of the sample space of the algorithm and increases the time complexity of the algorithm. It can also be found in Fig. 5 that the more the number of edge nodes, the greater the average delay of tasks solved by the algorithm. This is because the more the number of edge nodes, the less resources allocated to each node in the cloud, and the longer the task queuing time, resulting in the overall increase of task delay of all nodes, which affects the efficiency of task execution.

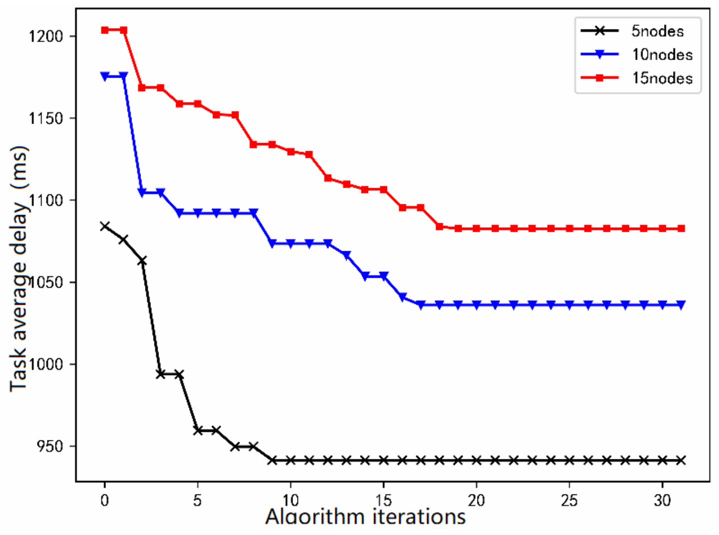

Fig. 5. Convergence performance of DMCJIA algorithm

4.2.2 Comparison of Other Algorithms

For the comparison of DNN model task reasoning effect, several other algorithms are analyzed as benchmark algorithm, and compared with DMCJIA algorithm in performance.

(1) Cloud-only algorithm: This algorithm does not split the model, but sends all the data to the cloud for reasoning. The cloud adopts the equal allocation mode for each node's task resource allocation;

(2) Greedy algorithm: Because the bottleneck of cloud task execution is the huge delay of data transmission, this algorithm optimizes the unloading position of the model through as little data transmission as possible. For the resource allocation in the cloud, each edge node task uses the cloud resources to calculate the reasoning task as much as possible under the restriction of minimum data transmission, and the amount of task calculation is positively proportional to the resource occupation;

(3) Neurosurgeon algorithm [15]: neurosurgeon algorithm is the only one that considers dynamic splitting in the existing research on DNN model splitting. Two factors are considered in the algorithm,

one is the static factor such as model structure, the other is the dynamic factor such as the load of cloud server, the number of connected devices in the network layer and the energy consumption of devices. By predicting the influence of different factors on task execution, the algorithm calculates the split position of the model. This scheme does not consider the resource allocation problem of the cloud, so in this experiment, the task of each edge node is equally allocated.

The experiment compares the different algorithms affected by the network link bandwidth. The total number of edge nodes is 10, the computing power of each node is 5GFLOPS, the task arrival rate is 1/s, and the link bandwidth from cloud to edge node changes from {0.5 ~ 5} MHz. As shown in Fig. 6, it can be seen that the cloud-only algorithm is most affected by the bandwidth fluctuation. When the bandwidth is very small, the effect of the algorithm is very poor. When the bandwidth gradually increases, the average task delay of the algorithm decreases rapidly. This is because the delay bottleneck of the cloud-only algorithm mainly depends on the data transmission delay. The smaller the bandwidth, the greater the transmission delay, which leads to the poor performance of the algorithm, Therefore, it is not suitable for cloud side task reasoning scenario with heavy bandwidth load. For the Greedy algorithm, because it implements the greedy strategy, the edge node will unload as many tasks as possible to the cloud computing, and ensure a relatively small data transmission delay, so the average task delay shows a downward trend, but the decline is small. The performance of Neurosurgeon algorithm is close to that of DMCJIA algorithm. With the increase of bandwidth, the average delay of tasks decreases linearly, and it has good performance in different link bandwidth, so it has better anti bandwidth fluctuation performance.

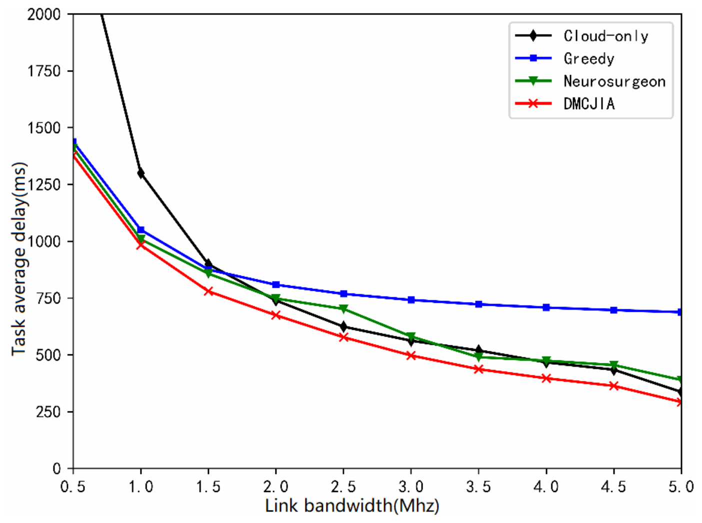

Fig. 6. Comparison of algorithm performance under different link bandwidths

Based on the above experiments, the iterative process and the results of decision variables are analyzed, and the performance of different algorithms is compared when the network link bandwidth changes. The results show that DMCJIA algorithm has good robustness in edge cloud multi node multi task joint reasoning optimization, and achieves good results for DNN reasoning task average delay optimization.

5 Conclusion

In edge intelligence, task real-time reasoning is an important application of artificial intelligence technology in the field of edge computing. This paper analyzes the reasoning task of DNN model in the environment of Internet of things. Considering the hardware characteristics of edge nodes and cloud and the actual physical scene, a multi task cloud edge joint reasoning optimization algorithm dmcjia based on DNN model is proposed to minimize the average delay of task execution. Then, the problem is transformed into a mixed integer nonlinear programming problem, which is decomposed into two subproblems: the joint optimization of model unloading and the optimization of cloud resource allocation. Finally, based on the experimental data set, the simulation results show that the delay cost of dmcjia algorithm is significantly reduced compared with the traditional algorithm, and the performance of dmcjia algorithm is better than other existing algorithms in the changing environment.

References

[1] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, J. Du, On-demand deep model compression for mobile devices: A usage-driven model selection framework, in: Proc. 16th Annual International Conference on Mobile Systems, Applications, and Services, 2018.

[2] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, L. Tang, Neurosurgeon: Collaborative intelligence between the cloud and mobile edge, ACM SIGPLAN Notices 52(4)(2017) 615-629.

[3] J. Mao, X. Chen, K.W. Nixon, C. Krieger, Y. Chen, MoDNN: Local distributed mobile computing system for deep neural network, in: Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017.

[4] T.Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, H. Balakrishnan, Glimpse: Continuous real-time object recognition on mobile devices, in: Proc. ACM Conference on Embedded Networked Sensor Systems, 2015.

[5] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, M. Satyanarayanan, Bandwidth-efficient live video analytics for drones via edge computing, in: Proc. IEEE/ACM Symposium on Edge Computing (SEC), 2018.

[6] S. Teerapittayanon, B. McDanel, H.T. Kung, Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices, in: Proc. 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017.

[7] S. Williams, A. Waterman, D. Patterso, Roofline: An Insightful Visual Performance Model for Multicore Architectures, Communications of the ACM 52(4)(2009) 65-76.

[8] S. Xu, Z. Zhang, M. Kadoch, M. Cheriet, A collaborative cloud-edge computing framework in distributed neural network, EURASIP Journal on Wireless Communications and Networking 2020(2020) 211.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[10] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[11] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, <https://arXiv.org/abs/1409.1556>, 2014.

[12] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proc. NIPS 2012, 2012.

[13] S.V. Subramanian, R. Dutta, Comparative Study of M/M/1 and M/D/1 Models of a SIP Proxy Server, in: Proc. 2008 Australasian Telecommunication Networks and Applications Conference, 2008.

[14] P. Guo, X. Wang, Y. Han, The enhanced genetic algorithms for the optimization design, in: Proc. 2010 3rd International Conference on Biomedical Engineering and Informatics, 2010.

[15] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, L. Tang, Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge, in: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’17), 2017.