Computers, Materials & Continua
DOI:10.32604/cmc.2022.024093
Article

Tech Science Press

Machine Learning-based Optimal Framework for Internet of Things Networks

Moath Alsafasfeh[1,*], Zaid A. Arida[2] and Omar A. Saraereh[3]

[1]Department of Computer Engineering, College of Engineering, Al-Hussein Bin Talal University, Ma'an, Jordan
[2]Abdul Aziz Ghurair School of Advanced Computing (ASAC), LTUC, Amman, P11118, Jordan
[3]Department of Electrical Engineering, Engineering Faculty, The Hashemite University, Zarqa, 13133, Jordan
*Corresponding Author: Moath Alsafasfeh. Email: moath.alsafasfeh@ahu.edu.jo
Received: 03 October 2021; Accepted: 16 November 2021

Abstract: Deep neural networks (DNN) are widely employed in a wide range of intelligent applications, including image and video recognition. However, due to the enormous amount of computations required by DNN. Therefore, performing DNN inference tasks locally is problematic for resource-constrained Internet of Things (IoT) devices. Existing cloud approaches are sensitive to problems like erratic communication delays and unreliable remote server performance. The utilization of IoT device collaboration to create distributed and scalable DNN task inference is a very promising strategy. The existing research, on the other hand, exclusively looks at the static split method in the scenario of homogeneous IoT devices. As a result, there is a pressing need to investigate how to divide DNN tasks adaptively among IoT devices with varying capabilities and resource constraints, and execute the task inference cooperatively. Two major obstacles confront the aforementioned research problems: 1) In a heterogeneous dynamic multi-device environment, it is difficult to estimate the multi-layer inference delay of DNN tasks; 2) It is difficult to intelligently adapt the collaborative inference approach in real time. As a result, a multi-layer delay prediction model with fine-grained interpretability is proposed initially. Furthermore, for DNN inference tasks, evolutionary reinforcement learning (ERL) is employed to adaptively discover the approximate best split strategy. Experiments show that, in a heterogeneous dynamic environment, the proposed framework can provide considerable DNN inference acceleration. When the number of devices is 2, 3, and 4, the delay acceleration of the proposed algorithm is 1.81 times, 1.98 times and 5.28 times that of the EE algorithm, respectively.

Keywords: IoT; distributed computing; neural networks; reinforcement learning

1 Introduction

In recent years, the Internet of Things (IoT) devices have become more and more common. According to Gartner data, the number of IoT devices is expected to reach 25 billion by 2021 [1–5]. As a typical representative of "Internet +", the IoT extends the traditional information communication to a wider physical world, which greatly expanding the coverage and composition of the Internet [6–9]. The wireless sensor network (WSN) is composed of a large number of sensor nodes with limited resources such as computing, communication, and energy in a multi-hop and self-organizing manner [10]. It is the core support of the perception layer of the IoT. Since the WSN was proposed in the 1990s, it has received extensive attention worldwide, especially in developed countries or regions such as the United States, Europe, Japan, and South Korea [11–13]. Continued to carry out related exploratory research, people usually think that: WSN technology has the ability to increase the existing network functions and improve the people's perception of the world. The IoT based on WSN has great potential, and its constantly emerging innovative application results will have a subversive impact on human life and social progress [14].

At present, deep neural networks (DNNs) are developing rapidly and have been widely used in various intelligent tasks (such as computer vision, video recognition and machine translation). The IoT devices are expected to perform DNN inference tasks to achieve real-time data processing and analysis. For example, in the smart home scene, the camera can perform video recognition and speech translation tasks based on the DNN model [15]. However, due to the limited resources of IoT devices, and the DNN task requires a lot of computing resources and memory usage, it is difficult for IoT devices to perform DNN inference tasks locally. In order to overcome the above-mentioned challenges, reference [16] proposed to split the DNN model between a single IoT device and cloud server to achieve task inference acceleration. However, limited by factors such as the large amount of transmitted data and the unpredictable network communication delay, the method of cloud assisting in the execution of DNN task inference is difficult to guarantee the efficiency of data processing, and it will increase the dependence on cloud services.

Aggregating the computing power of multiple IoT devices to perform DNN tasks together is an effective solution. The advantage of this approach is to reduce the dependence on cloud services, protect the privacy of IoT devices, and enable the distributed collaborative computing. Reference [17] is the first to use resource-constrained multiple IoT devices to collaborate to perform DNN tasks such as voice and video recognition. Reference [18] proposed the DeepThings framework to divide the convolutional layer to reduce the overall execution delay and memory usage. However, the existing research work only considers the isomorphism of IoT devices, and cannot achieve real-time dynamic DNN task splitting. How to efficiently split DNN tasks and collaborative inference in dynamic heterogeneous scenes is a key issue to be solved urgently.

The above-mentioned research problems face two important challenges. First, different parameter configurations (layer type, number of layers, convolution kernel size, input feature size, etc.) and heterogeneous device capabilities lead to significant differences in inference delays. It is impractical to perform DNN inference tasks on demand to obtain the inference delay under each system setting and task splitting strategy. Therefore, it is necessary to predict the current system state and the inference delay caused by the split collaboration strategy in advance. The existing DNN delay prediction model is based on the single-layer prediction, and the multi-layer prediction delay is obtained by adding the single-layer prediction delay. However, reference [19] found through experiments that, the difference between the sum of the delays of the individual execution of each layer and the overall execution delay becomes more obvious with increasing number of convolutional layers, and the existing DNN delay

prediction model cannot be within the acceptable error range to perform effective evaluation and prediction of inference delay. Moreover, the existing delay prediction model only considers specific parameter configuration, and does not consider the impact of equipment capabilities on DNN inferred delay. Therefore, it is of great significance to study the accurate multi-layer delay prediction model in the case of multiple parameter configurations and heterogeneous equipment.

DNN task splitting will generate communication overhead while distributing the amount of computation. Although, increasing the number of devices that cooperate to perform DNN tasks will reduce the calculation delay of a single device, it will also increase the communication delay between devices. Therefore, the collaborative splitting strategy needs to efficiently weigh the calculation and communication delays. Because the DNN structure, network status, and device capabilities are dynamically changing and highly heterogeneous, the DNN task splitting and collaborative inference strategies need to be dynamically adjusted and efficient decision-making based on the current system state, determine the number of devices to perform tasks, and select the split of DNN tasks according to the location and the computing tasks assigned to each device, in order to obtain the optimal DNN inference acceleration and make full use of the computing power of the IoT device [20]. In view of the above problems, traditional optimization methods have high computational complexity and long solution time, making it difficult to apply. The data-driven artificial intelligence methods can establish automated decision-making models through data processing and analysis, training and learning, and making decisions directly based on the learned decision-making model when the system status changes, thereby achieving adaptive, intelligent and real-time decision-making. This paper uses a data-driven learning algorithm to develop real-time intelligent DNN task splitting and collaborative inference strategies under the diversification of device capabilities, network status, and DNN tasks.

This paper proposes a novel IoT device collaborative execution DNN task inference (IoT-CDI) framework. Based on various factors such as DNN structure, device capabilities, and network status, it can adaptively adjust the DNN splitting and task allocation strategies, which can be used when resources are limited. It realizes the DNN collaborative inference between heterogeneous IoT devices, and makes full use of the computing power to minimize the inference delay of DNN tasks. The main contributions of this paper include three aspects:

1) Fine-grained characterization of DNN model layer types, parameter configuration and equipment capabilities, etc., mining complex mapping relationships between features and execution delays, generating interpretable multi-layer delay prediction models, and evaluating a variety of common predictions through a large number of experiments. Then, it obtains an accurate model suitable for multi-layer delay prediction.
2) Convert the original DNN split and collaborative inference problem into the shortest path discovery problem, and reduce it to an NP-hard problem. An adaptive DNN splitting and collaborative inference algorithm based on Evolutionary Reinforcement Learning (ERL) is proposed to realize the real-time intelligent DNN inference acceleration among heterogeneous devices.
3) Use real experiments to verify. Five common DNN models and various types of Raspberry Pi devices are selected to verify the effectiveness of the proposed IoT-CDI framework. The experimental results show that the proposed IoT-CDI can significantly improve the inference speed and is better than the benchmark algorithms.

The remaining of the paper is organized as follows. In Section 2, the literature review is discussed. In Section 3, the background introduction and research motivation are elaborated. In Section 4, the proposed IoT-CDI model is explained. In Section 5, the task splitting mechanism of DNN is discussed.

Section 6 provides the proposed framework. In Section 7, the experimental results analysis is described. In Section 8, the discussion on numerical results of the algorithms is given while Section 9 concludes the article.

2 Literature Review

2.1 Research on End-Cloud Collaboration Inference

Limited by the memory limitations and computing resource constraints of IoT devices, existing work is mainly devoted to the research of DNN task collaboration inference strategies between the IoT devices and cloud servers. Reference [21] proposed a DNN inference delay prediction algorithm based on a tree regression model. In [22], the author designed a flexible and efficient two-step pruning algorithm. According to multiple factors, such as hierarchical data transmission and calculation delay, tolerable accuracy loss, wireless channel and device computing power, etc., the pruning model and the optimal DNN splitting position are determined. While reducing the load of calculation and communication transmission, it also satisfies the inference accuracy requirements of DNN tasks. The authors in [23] designed an adaptive DNN splitting algorithm, which can find the optimal splitting strategy under dynamic and time-varying network load conditions.

Although the collaborative inference of IoT devices and cloud servers can use the computing power of cloud servers to reduce the inference delay, there are still problems such as high dependence on cloud servers, unscalable inference, long communication delay, and device privacy protection.

2.2 IoT Device Collaboration Inference

As cloud assists DNN task inference facing the above-mentioned problems, an emerging research trend is to aggregate the computing capabilities of resource constrained IoT devices, and multiple IoT devices collaborate to perform DNN inference tasks. Reference [24] used multiple IoT devices to perform the DNN inference for the first time, and achieved task inference acceleration by reducing the computational cost and memory usage of a single device. However, the existing research work does not consider the heterogeneous capabilities of IoT devices, dynamic changes of environmental conditions, and is difficult to achieve real-time adaptive decision-making under the diversified environment configuration and high computational complexity of problem solving. It is worth noting that, the above work is orthogonal to the compression and acceleration methods that use weight pruning [25,26], quantization [27,28] and low-precision inference [29,30] to reduce the computational cost of DNN models. At the same time, these two technologies are used to accelerate the DNN inference. Reference [31] proposes a novel system energy consumption model that considers the runtime, switching, and processing energy consumption of all involved servers (cloud and edge) and IoT devices. Then, utilizing a Self-adaptive Particle Swarm Optimization algorithm with Genetic Algorithm operators (SPSO-GA), a novel energy-efficient offloading approach is developed. With layer partition procedures, this innovative technique can efficiently make offloading decisions for DNN layers, reducing the encoding dimension and improving SPSO-GA execution time. The authors in [32] provide a technology framework that supports fault-tolerant and low-latency AI predictions by combining the Edge-Cloud architectural concept with BranchyNet advantages. The benefits of running Distributed DNN (DDNN) in the Cloud-to-Things continuum may be assessed thanks to the deployment and evaluation of this architecture. Reference [33] proposes a new convolutional neural network structure—BBNet—that speeds up collaborative inference on two levels: (1) through channel-pruning, which reduces the number of calculations and parameters in the original network; and (2) by compressing the feature map at the split point, which reduces the size of the data transmitted even more.

Tab. 1 compare the summary of related works and proposed method.

Table 1: Comparison of the related works and proposed work

Parameter	Ref. [18]	Ref. [19]	Ref. [20]	Ref. [23]	Ref. [29]	Ref. [31]	Proposed
KPI	Hierarchical	Fast inference	Fast inference	Fast inference	Hierarchical	Fast inference	Fast inference
Model	DNN	DNN	CNN	Energy	Logistic regression	SVM	DNN
Application	Distributed computing	Image recognition	Video analysis	Battery lifetime estimation	Human activity recognition	Code execution	Distributed computing and regession

3 Background Introduction and Research Motivation

This section first introduces the types and characteristics of DNN layers, and then leads to the research motivation of this article based on real experimental analysis.

3.1 DNN Layer Type

DNN tasks include multiple layer types, such as convolutional layer (conv), fully connected layer (fc), pooling layer, activation layer and Softmax layer. Among them, the computational cost and memory usage of the convolutional layer and the fully connected layer are the most. The fully connected layer has the largest memory overhead for more than 87%. Therefore, this article only focuses on the convolutional and fully connected layer in the DNN model.

3.2 Real Problem

1) Model prediction. The current research work only considers the single-layer delay prediction models with different layer types under different configuration parameters. However, the authors show that, there are obvious prediction errors in evaluating the multi-layer delay through the single-layer delay accumulation method. We conduct real experiments to conduct a comprehensive analysis of the multi-layer delay prediction problem, and reveal the true relationship between the delay sum of each layer executed separately and the actual delay of the entire multi-layer execution on DNN models with different channel types. As the number of different channel types gradually increases, the similarity of DNN models gradually decreases. As shown in Fig. 1, the abscissa represents the number of different channel types and the ordinate represents the reduction ratio of the overall execution delay compared to the individual execution delay summation. In the case of the same convolutional layer channel type, the overall execution delay is reduced by 50% compared with the delay summation executed separately. If the number of different channel types is large, it means that the convolutional layer has low similarity, and the delay of separate execution sum is approximately equal to the overall execution delay. This experiment provides persuasiveness for the development of a multi-layer delay prediction model, which is used to better guide the DNN task splitting and collaborative inference.

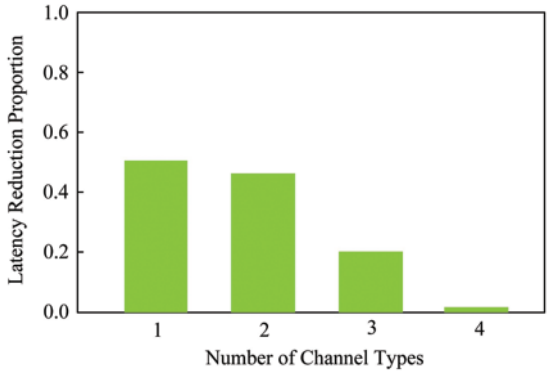

Figure 1: Comparison of latency

2) Equipment heterogeneity. First, measure the inference delay of five common DNN models on three variants of Raspberry Pi (Raspberry Pi 2B, Raspberry Pi 3B and Raspberry Pi 3B+). Five DNN models are executed on each model of Raspberry Pi device. The experimental results are shown in Fig. 2. The bar graph represents the inferred latency, and the line graph represents the ratio of the execution latency of different devices. For example, the AlexNet model is used on the Raspberry Pi 2B. The inference delay required for the above execution is 1.66 s, while the execution delay on the Raspberry Pi 3B is reduced to 1.06 s, which is only the inference delay of the Raspberry Pi 2B 64%. It can be seen that; the difference of equipment capabilities will significantly affect the inference delay of DNN tasks. Moreover, as the amount of calculation of the DNN model increases, the difference in the inference delay caused by the execution of DNN tasks by different devices becomes more prominent. The inference delays of the VGG16 model executed on Raspberry Pi 2B and 3B are 11.68 and 5.24 s, respectively, and the execution speed is increased by about 2.23 times. This experiment shows that, the DNN splitting should consider the heterogeneous capabilities of the device, and make full use of the computing resources of the device to achieve the approximate optimal inference acceleration. For this reason, it is necessary to design an accurate model to analyze the impact of equipment heterogeneous capabilities on the DNN inference delay.

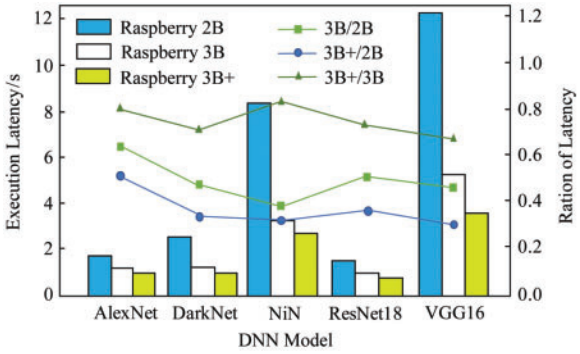

Figure 2: Comparison of latency of various deep neural networks models

4 IoT-CDI Model

4.1 System Model

The schematic diagram of the IoT-CDI scenario is shown in Fig. 3. It is assumed that, there is a group of IoT devices with heterogeneous capabilities $N = \{1, 2, \ldots, N\}$. Each device dev_i generates a DNN inference task m with a certain probability. The DNN task inference is carried out layer by layer, the output of the previous layer is the input of the next layer, and the task is terminated when all layers are executed. Suppose a DNN inference task m contains K layers, and each layer is considered a subtask. For a DNN inference task such as video recognition, usually a series of data frames are continuously input to the DNN model for inference, and the sampling rate is assumed to be Q frames/second.

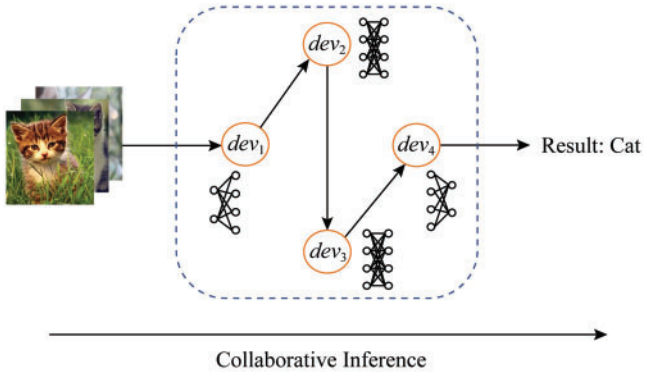

Figure 3: Proposed system model

Given the number of available devices N and the number of DNN subtasks (number of layers) K, the goal is to find the split position of the DNN task and the optimal task allocation of these devices. For each subtask k, find an IoT device *devi* to execute it. After each IoT device dev_i executes the assigned computing task (some layers of the DNN task), the output data generated is transmitted to the device that performs the next layer task until the DNN task inference is completed. The research goal is to minimize the overall execution delay of the DNN tasks. If all subtasks are executed on one IoT device, the limited resources of a single IoT device will cause a long calculation delay. However, if tasks are distributed to multiple IoT devices, the communication delay increases significantly. Therefore, it is necessary to split and allocate the DNN tasks reasonably, effectively weigh communication and calculation delays, and minimize the overall inference delay of DNN tasks.

4.2 Problem Description

We convert the DNN task splitting problem between IoT devices into a directed acyclic graph (DAG) expressed as $G = (V, L)$, where the vertex $v_{ik} \in V$ indicates that the k-th layer of the DNN model is assigned to the IoT device dev_i. The edge $l_{ijk} \in L$ indicates that the k-th layer of the DNN model is allocated to the IoT device dev_i, and the $(k+1)$th layer is allocated to the IoT device dev_j. The graph model representation is shown in Fig. 4. If the edge l_{ijk} $(i \neq j)$ is selected, the corresponding inference delay is represented as T_{ijk}. The calculation delay T^c_{ijk} of the k-th layer of the DNN model is executed for the IoT device dev_i and the output result of the k-th layer is transmitted from the sum of the communication delay T^c_{ijk} from IoT device dev_i to dev_j. If $i = j$, then the communication delay is zero, that is, T^c_{ijk}. Let C_{ik} denote the memory cost required by the IoT device dev_i to execute the k-th layer.

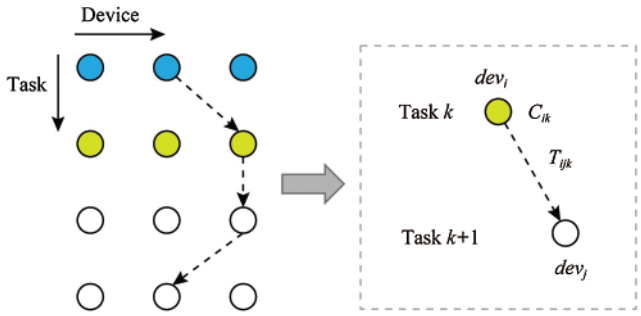

Figure 4: Schematic flow of the IoT-CDI

The IoT-CDI problem can be transformed into an optimal path problem from the first layer to the K-th layer. The problem is expressed as:

$$\min \sum_{i,j\in N} \sum_{k=0}^{K} l_{ijk} \times T_{ijk}$$

$$\text{s.t.} \sum_{i\in N} l_{ijk} = \sum_{h\in N} l_{ihk+1},\ 0 \le k \le K,\ \forall j \in N \tag{1}$$

$$\sum_{k=0}^{K} l_{ijk} \times C_{ik} \le B_i,\ 0 \le k \le K,\ \forall i \in N \tag{2}$$

$$\sum_{j\in N} l_{ijk} = 1,\ 0 \le k \le K,\ \forall i \in N \tag{3}$$

$$l_{ijk} \in \{0, 1\} \tag{4}$$

Eq. (1) indicates that, if the $(k+1)$th layer is allocated to the IoT device dev_j, an edge starting from the IoT device dev_j needs to be selected. Eq. (2) represents the memory limit of each device. Eq. (3) ensures that each layer is executed by only one device. In addition, the DNN inference is usually composed of multiple input data streams, so the optimization goal needs to be data stream-oriented. Once the DNN splitting strategy is determined, each frame needs to be processed in order according to the strategy. We introduce the concept of pipeline processing as shown in Fig. 5. Specifically, for two consecutive data frames, the IoT device dev_i first completes the task assigned by the data frame 1, and when the data frame 2 arrives, the IoT device dev_i will immediately execute the task of the data frame 2. Obviously, the bottleneck of pipeline processing is the maximum value of T_{ijk}, which is the device with the longest processing time for a single frame. This fact is verified through experiments. The VGG16 model is divided into three parts and executed on different devices. The time for each device to execute one frame is 2.374, 7.768 and 1.456 s, and a single frame is executed. The maximum inferred delay of the three devices during the task is 7.768 s, and the total execution delay of 100 frames in the experimental test is approximately equal to 100×7.768 s. In order to enable the DNN split and task allocation strategy to support the pipeline processing, the delay calculation formula is modified to the maximum value of the individual execution delay of each IoT device. The method of multi-device cooperative execution of DNN tasks proposed in this paper aggregates the computing power of multiple devices and makes full use of the concurrent processing capabilities, which can effectively

improve the overall throughput. It is achieved by adaptively splitting the DNN tasks among multiple IoT devices in real time with the goal of minimizing the total inference delay after processing all data frames.

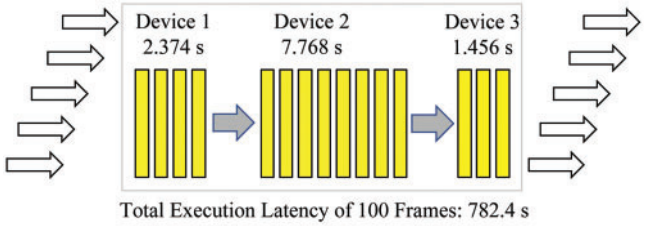

Figure 5: Processing illustration of deep neural network model

4.3 Problem Solution

First it is proved that, the IoT-CDI problem is NP-hard, and then use the known NP-hard problem—general assignment problem (GAP) to prove it [34,35]. The GAP assumes that, there are M items and N boxes, put item i into box j, and get the income $M_{i,j}$. The goal is to pack each item into an appropriate box, and maximize the overall revenue under the constraints of the cost of each box. Through parameter mapping and conversion, the IoT-CDI problem is reduced to a GAP problem, which proves that the problem is NP-hard.

Since the IoT-CDI problem is NP-hard, it is difficult to obtain the optimal DNN splitting and collaborative inference strategy in polynomial time. Therefore, accurate algorithms such as enumeration are not suitable for solving this problem. In addition, due to the diversity of DNN model structures, heterogeneous equipment capabilities and dynamic changes in communication status, it is necessary to adjust the collaborative inference strategy in real time. To this end, we adopt a data-driven artificial intelligence method to solve the problem, which can make real-time automated decision-making based on environmental information. Reinforcement learning (RL) is an effective data-driven method that continuously learns and guides behavior by interacting with the environment to obtain rewards to obtain the maximum benefits. In this paper, an enhanced learning algorithm is used to determine the optimal DNN splitting strategy, and to perform collaborative inference between heterogeneous devices to achieve inference acceleration.

5 DNN Task Split Strategy

In this section, we first elaborate and analyze the proposed accurate multi-layer delay prediction model through specific parameter configuration and a variety of typical prediction models. On this basis, the ERL algorithm is used to intelligently and adaptively determine the cooperative inference strategy between heterogeneous devices.

5.1 Parameter Configuration of Convolutional Layer and Fully Connected Layer

The convolutional layer includes input feature dimensions (input height *in_height*, input width *in_width*), convolution kernel size (*kernel_height*, *kernel_width*), channel size (*in_channel*, *out_channel*), stride and padding. The parameter configuration of the fully connected layer includes the input feature dimension (*in_dim)* and the output feature dimension (*out_dim)*. The parameter configuration range is shown in Tab. 2. The configurable parameters of each layer are generated by random combination, and the execution delay Y of each parameter combination is measured. Similar

to [36], the interpretable parameter vector $\boldsymbol{X}$ is determined according to the above model parameters, including floating point operations (FLOPs), memory footprint and parameter scale. The specific definition of the interpretable parameter vector $\boldsymbol{X}$ is: $X = (FLOPs,\ mem,\ param\_size)$, where $mem = mem\_in + mem\_out + mem\_inter$, mem_in represents the input data occupancy scale, mem_out represents the memory occupancy scale of output data, mem_inter represents the memory occupancy scale of temporary data, detailed definitions of memory and parameter characteristics can be found in [37]. The CPU operations and memory operations affect the execution time of the program to a certain extent. In the DNN model, the CPU operations and memory operations are reflected in floating-point operations, memory footprint and parameter scale. A large number of [$\boldsymbol{X}$, Y] data pairs are obtained through various parameter configuration combinations for delayed model training and prediction.

Table 2: Layers parameters

Type	Parameter
conv	$in - height \in [7, 299],\ in - width \in [7, 299],\ kernel_{height} \times kernel_{width} \in \{1 \times 1,\ 2 \times 2,\ 3 \times 3,\ 4 \times 4,\ 1 \times 3,\ 1 \times 4\},\ in_{channel} \in [3,\ 2048],\ out_{channel} \in [3,\ 2048],\ padding \in [valid,\ same],\ stride \in \{1, 2\}$
f_c	$in_{dim} \in [1,\ 4096],\ out_{dim} \in [1,\ 4096]$

5.2 Multi-Layer Delay Prediction Model

In this section, we conduct a comprehensive study on the multi-layer delay prediction model of the convolutional and the fully connected layer. The interpretable parameter vector $\boldsymbol{X}$ of the multi-layer delay prediction model includes the number of layers, the sum of floating-point operations, memory footprint and parameter scales. In order to perform multi-layer predictive analysis, first generate a DNN model of any number of layers, and generate a characteristic parameter combination, execute on IoT devices with different computing capabilities to obtain the execution delay Y in the case of any number of layers and different parameter configurations. After obtaining the [$\boldsymbol{X}$, Y] data pair, establish the correlation model of equipment capabilities, task characteristics and execution delay, study a variety of common predictive models to fit multi-layer input data and execution delay, and mine a variety of characteristic parameters and execution mapping relationship between delays. The coefficient of determination R^2, mean squared error (MSE) and mean absolute percentage error (MAPE) are used as the evaluation indicators of the accuracy of the prediction model. Also, study the linear regression (LR), RANdom SAmple Consensus regression (RANSAC), kernel ridge regression (KRR), k-nearest neighbor (KNN), decision tree (DT), support vector machine (SVM), random forest (RF), AdaBoostADA, gradient boosted regression trees (GBRT) and artificial neural network (ANN) models.

Compared with the convolutional layer, the fully connected layer has a shorter execution time, fewer parameters and a small number of layers. For example, the AlexNet model only contains three fully connected layers, and the ResNet model only contains one fully connected layer. We prove through experiments that the error of the sum of the overall execution delay and the individual execution delay of the fully connected layer is less than 2%. Therefore, we only study the single-layer

prediction model executed by the fully connected layer on different devices, and compare the prediction performance of different prediction models on the fully connected layer. From Tab. 3, it can be seen that a variety of prediction models can predict fully connected execution delay of the layer. For the convolutional layer, due to the many types of input feature parameters, the wider configuration range, the number of execution layers and the complex coupling relationship between feature parameters, the delay prediction is relatively increased. Adding the ANN prediction model, because the neural network can effectively obtain the nonlinear relationship and has strong generalization and fitting ability, and can obtain an approximate actual model without assuming the mapping relationship between the feature variable and the result.

Table 3: Comparative performance of different algorithms for single-layer

Algorithm	Parameter		
	R^2	*MSE*	*MAPE*
LR	0.999221	0.000295	0.043131
RANSAC	0.998915	0.000293	0.036892
KRR	0.962820	0.002878	0.579788
KNN	0.997838	0.000334	0.041826
DT	0.998446	0.000349	0.048217
SVM	28.632466	0.091143	17.990913
RF	0.998723	0.000333	0.047756
ADA	0.990998	0.001435	0.262618
GBRT	0.998865	0.000297	0.040126

Taking Raspberry Pi 3B as an example, Tab. 4 compares the performance of different multi-layer delay prediction models for the convolutional layer. It can be seen from Tab. 4 that, the performance of the three prediction models of RF, GBRT and ANN is better than other models. For example, compared with the RANSAC model and the ADA model, the MAPE index of the ANN model is reduced by 43% and 81%, respectively. The experiment in Section 6 further verify the accuracy of these three multi-layer prediction models.

Table 4: Performance comparison of the algorithms for conv multi-layer

Algorithm	Parameter		
	R^2	*MSE*	*MAPE*
LR	0.884172	0.164958	0.294132
RANSAC	0.878270	0.174933	0.215770
KRR	0.872121	0.182157	0.269433
KNN	0.947610	0.119536	0.139216
DT	0.930440	0.131583	0.144607
SVM	0.919145	0.158629	0.318136

(Continued)

Table 4: Continued

Algorithm	Parameter		
	R^2	*MSE*	*MAPE*
RF	0.965833	0.103031	0.119834
ADA	0.950137	0.169817	0.633392
GBRT	0.969417	0.102545	0.155241
ANN	0.973797	0.092394	0.123457

5.3 DNN Task Splitting Strategy Based on Evolutionary Reinforcement Learning

1) Description

Reinforcement learning (RL) is an effective machine learning algorithm for decision making. Agents can observe the state of the environment and learn which behaviors can obtain better returns. At each time step t, the agent observes the current environment state s_t, and chooses a behavior a_t according to the strategy $\pi \in (a_t|s_t)$. The instantaneous profit r_t is obtained after the execution of the behavior, and the state transition is performed according to the state transition probability environment, and the state is adjusted to $s_t + 1$. The goal of the agent is to obtain the optimal strategy to maximize the cumulative discounted income $R_t = \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)$, and the discount factor is γ^t. Strategy learning is based on the behavior value function, which is defined as the expected value of the cumulative discounted income that each state behavior can obtain, and is calculated as:

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s,\ a_t = a\right] \tag{5}$$

The goal of reinforcement learning is to find the optimal strategy to maximize the behavior value, which can be expressed as $\pi^* = \underset{a}{\arg\max} Q^*(s,a)$.

Deep reinforcement learning (DRL) [38] is proposed to solve the curse of dimensionality. DRL uses a DNN to approximate the Q function $Q(s_t, a_t) \approx Q(s_t, a_t|\theta)$, where θ represents the model parameters of the neural network. Deep Q-network (DQN) is a typical DRL method [39]. DQN stores the experience tuples in the experience pool, each time a batch of samples are randomly selected from the experience pool for training, and then the parameter θ is updated to minimize the loss function. However, the DQN method based on back propagation cannot be optimized for a long time, and it is difficult to learn the optimal behavior when the reward is sparse (a series of behaviors can be used to obtain benefits). In addition, in the face of high-dimensional action and state spaces, efficient exploration is still a key challenge that needs to be solved urgently. In this case, there is a challenge of difficulty in convergence. In summary, DQN is a traditional DRL algorithm which faces important challenges such as sparse rewards, lack of effective exploration, and difficulty in convergence. Therefore, traditional DRL algorithms (such as DQN) cannot be directly applied to solve the IoT-CDI problem, because the problem behavior is decomposed into continuous sub-behaviors, there are problems such as sparse rewards and huge behavior state space, and convergence is very difficult. For this reason, the evolutionary ERL algorithm [40] is proposed to realize the DNN splitting and collaborative inference among heterogeneous devices.

2) DNN Task Splitting Strategy Based On ERL

From the perspective of DRL, the device used to determine the DNN splitting strategy is modeled as an agent. In order to reduce the dimensions of the state and behavior space, the DNN split task is decomposed into hierarchical sequence subtasks, and each layer is treated as a subtask. In each decision-making, you only need to select the appropriate execution equipment for each layer of the model. The behaviors of each layer obtain the overall behavior set, and perform DNN task splitting and collaborative inference according to the behavior set. The DNN task execution delay is used as the benefit to measure the performance of the behavior set. First define the basic elements of the state, behavior, and return of the problem.

1) State. At each time t, the state s_t contains 5 parts:
   i) f_t represents the current number of layers;
   ii) com_t represents the current network status, that is, the communication rate
   iii) $c_t = \{c_{1,t}, c_{2,t}, \ldots, c_{N,t}\}$ represents the capability of each IoT device;
   iv) $l_t = \{l_{1,t}, l_{2,t}, \ldots, l_{N,t}\}$ represents the cumulative delay required for each IoT device to complete the pre-allocated subtask;
   v) $e_t = \{e_{1,t}, e_{2,t}, \ldots, e_{N,t}\}$ represents the inferred delay caused by the execution of the current subtask assigned to each IoT device. From the above description, we can see that $s_t = (f_t, com_t, c_t, l_t, e_t)$, The state dimension is $3N + 2$.
2) Behavior. a_t means to select a device from N IoT devices to perform the current subtask.
3) Revenue. If the current subtask is the last one, the revenue is the overall inferred delay of the DNN task (for the data flow situation, the revenue is the maximum value of the delay required for each IoT device to perform its own task), otherwise the revenue is zero.

The DRL algorithm based on back propagation is difficult to obtain the optimal strategy for this problem, because this problem faces challenges such as sparse rewards and difficult exploration. Compared with the traditional DRL method, the ERL integrates the population-based method in the natural evolution strategy, which makes diversified exploration possible, and uses fitness indicators to learn and generate better offspring, so that multiple strategies can be effectively explored, and continue to evolve towards high returns.

The ERL process is as follows: Apply evolution to the candidate sample population, and continuously generate new offspring by increasing the random deviation. By performing the selection operation, the offspring with a higher fitness value have more chances to retain and produce new offspring. The higher the fitness value, the better the performance, and the next generation by the selection operation will provide better performance. In this article, each sample represents a set of parameters of the neural network, and the random deviation added to the offspring represents random disturbance to the weight of the neural network.

The overall algorithm flow is shown in Algorithm 1.

Algorithm 1: ERL DNN algorithm

Input: random weight θ of behavior value function Q, parent weight θ, number of children C, learning rate η;
Output: Parent weight θ^p.
1: for episode $:=1, 2, \ldots E$ do
2: Initialize state s
3: For i in range C do

(Continued)

Algorithm 1: Continued

4: $\theta^i = \theta^p + noise$
5: Select the behavior a^i and observe the revenue r^i
6: Calculate the average return r and calculate the gain of each offspring $g^i = ri - \bar{r}$
7: $\theta^p = \theta^p + \eta \times \sum_{i=1}^{C} g^i \times \theta^i$
8: End for
9: End for

In Algorithm 1, the parameters are initialized at the beginning. Then describe how to update the neural network during training. Specifically, the parent neural network generates C child neural networks by perturbing the parameters of the neural network, and evaluates the income value obtained by each child during each iteration, that is, the fitness value. If a child has a higher fitness value, then the child is selected with a higher probability and the offspring is generated. Calculate the gain value of each child by normalizing the difference between the income value obtained by each child and the average income value of all children. Update the parameters of the parent neural network according to the gain value g of C children (steps 3∼9).

6 Proposed Framework

The overall process diagram of the IoT-CDI framework is shown in Fig. 6, which includes two stages of offline training and online execution. The offline stage generates a multi-layer delay prediction model and completes the training process of the ERL algorithm. The online stage dynamically determines the split location and based on the system state. Task allocation, multiple devices cooperate to perform DNN tasks together. The topological structure of different DNN tasks is different, the calculation amount of each layer and the amount of intermediate data transmission generated are different, network status changes directly affect the data transmission delay, and the heterogeneity of equipment capabilities significantly affects the calculation delay. So it needs to be based on these dynamic factors, automatically adjust the DNN task splitting and allocation strategy to effectively reduce the inference delay. The IoT-CDI framework can determine the split location of the DNN model and the task assignment of each device according to the current system status, including communication status, device capabilities, and DNN task requirements, and realize distributed and collaborative DNN task inference among heterogeneous devices. It deploys a master device (IoT device or gateway) to manage and control the entire process.

6.1 Offline Training Phase

In this stage, the training of multi-layer delay prediction model and ERL split strategy training are mainly carried out. For the two types of convolutional and fully connected layer, the delay prediction model under the condition of arbitrary multi-layer different parameter configurations is described, which allows accurate evaluation of the actual execution delay of the inference task without executing the DNN task. Due to different layer types, layer parameter configurations and the number of layers will have obvious delay differences. So build different layer types of prediction models (convolutional layer and fully connected layer), change the number of layers and each layer parameter of each layer type configure, use these parameters to determine the calculation scale and data transmission scale, and analyze the impact of different device capabilities on execution delay when the parameter configuration is the same. Real measurement data of parameter configuration, equipment capability and execution

delay are obtained through experiments, and the prediction model training is carried out based on the data. A variety of common prediction models, involving regression, k-nearest neighbors, decision trees, combination and artificial neural network models and other types of models are analyzed. Through experiments, it is found that there are fewer types of parameters in the fully connected layer, and the prediction is relatively simple, and many models can obtain accurate prediction performance. The convolutional layer has many parameter types and complex configurations, so the performance of the prediction model with strong generalization and nonlinear fitting capabilities is more accurate. It is worth noting that, by mapping the model parameters to the calculation and transmission scale and analyzing the impact of different device capabilities on the execution delay, the proposed prediction model is independent of the DNN model and related to the device capabilities and can be adapted to heterogeneous devices. When the DNN model structure and parameters change, it can quickly obtain accurate execution delay based on the prediction model, avoiding additional execution overhead. Based on the generated multi-layer delay prediction model, the ERL algorithm is trained in order to obtain the approximate optimal DNN task splitting and collaborative inference strategy when the DNN model, network status and device capabilities dynamically change. The status information of the ERL model includes model parameters, number of layers, communication status, and device capabilities. The behavior strategy is to determine the execution equipment of each layer of the DNN model. After training 2,000 times to reach convergence, the ERL model after training is stored on the main device, and then the best split strategy is determined based on the input system state.

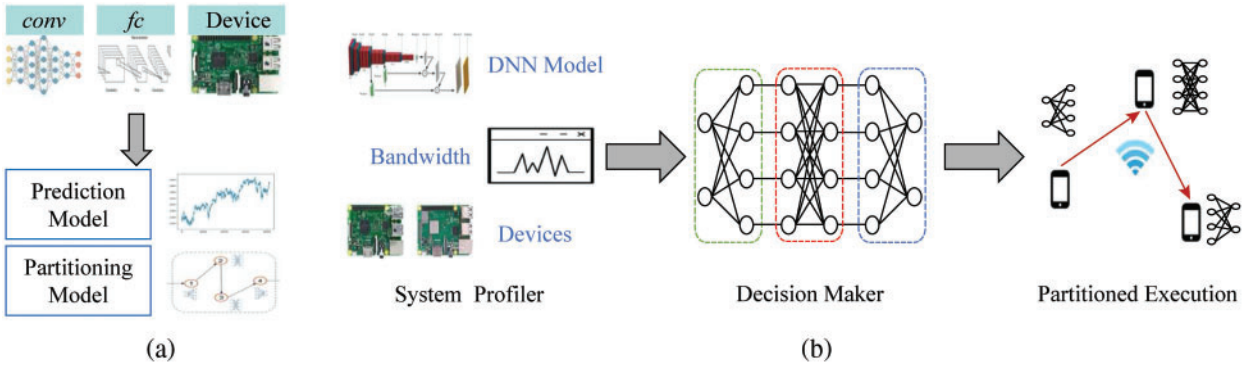

Figure 6: Proposed framework

6.2 Online Execution Phase

This stage includes three steps: 1) The system profiler obtains the current system status, including DNN inference tasks, current communication status and device capabilities, etc.; 2) This information is fed back to the decision maker, and it uses offline training to complete the completed multi-layer delay prediction model evaluates the inference delay of each candidate decision, and uses the ERL split model that is also trained in the offline phase to obtain the optimal split strategy to achieve DNN inference acceleration and device resources among heterogeneous multiple devices. 3) Each device executes its assigned tasks according to the split strategy.

IoT devices need to communicate with each other to transmit commands and data. In order to effectively identify the device, each IoT device needs to register an IP address. After knowing the DNN task splitting and allocation strategy, maintain each device's own IP processing table, which records the inference tasks assigned to it and the predecessor and successor nodes of its own task. The master

device maintains the overall IP processing table, which records the execution tasks of each device. Once the system status changes, for example, the communication rate changes or new equipment joins or exits, it will trigger the adjustment of the split strategy, and the master node will update the record. Then the master node distributes the updated information of the IP processing table to all devices, and each device modifies its own IP processing table according to the updated information.

The DNN inference process is executed according to the IP processing table. An IoT device will receive the input data required for calculation from the predecessor device, and send the generated output result to the successor device after completing the assigned task. In order to realize the above process, the remote procedure call (RPC) is deployed to realize the interaction between devices, which can communicate and transmit data between two devices. Taking the VGG model as an example, suppose that device 1 implement the 1∼5 layers of the VGG model, and its successor device implements the 6–10 layers of the VGG model for device 2. After device 1 completes the assigned number of layers, it sends the generated output result to the subsequent device 2, and the two devices jointly execute the DNN tasks according to the strategy. When the environment status changes, adjust the split and allocation strategy according to the ERL algorithm. For example, device 1 executes layer 1∼7, device 2 executes layer 8∼10, you need to update the IP processing table of each device, modify the allocation task and the predecessor and successor node.

7 Experimental Verification

We use real experiments to verify the proposed IoT-CDI framework. First, it is proved that the proposed multi-layer delay prediction model is accurate. Then, compared with the benchmark algorithm, it is found that the proposed ERL method can significantly reduce the inference delay and realize the acceleration of inference. In addition, we also evaluate the influence of factors such as communication status and the number of devices on the performance of the experiment.

7.1 Experimental Setup

1) Device type. Three types of Raspberry Pi devices are used as heterogeneous IoT devices, namely Raspberry Pi 2B, Raspberry Pi 3B and Raspberry Pi 3B+, using Raspbian GNU/Linux10 buster operating system. Different models of Raspberry Pi have different computing capabilities, providing differentiated inference performance. The specifications of different models of Raspberry Pi are shown in Tab. 5. In order to perform DNN tasks on the Raspberry Pi, we install basic software and platforms such as Python 3.7.3, Keras 2.2.4 and Tensorflow 1.13.1.

Table 5: Configuration of Raspberry Pi

	Device		
	Raspberry Pi 2B	Raspberry Pi 3B	Raspberry Pi 3B+
CPU	900 MHz Quad Core ARM Cortex-A7 Broadcm BCM2836 64 bit	1.2 MHz Quad Core ARM Cortex-A53 Broadcm BCM2837 64 bit	1.4 MHZ Quad Core ARM Cortex-A53 Broadcm BCM2837B0 64 bit
Memory RAM size	1 GB	1 GB	1 GB

2) DNN model. Five common DNN models are used, namely AlextNet, DarkNet, NiN, ResNet18 and VGG16. The VGG16 represents the long DNN model (with more layers), and AlexNet represents the short DNN model (with fewer layers). The AlexNet model and the ResNet18 model are less computationally intensive, while the VGG16 model and the NiN model are more computationally intensive. The type of calculation is relatively large, but the communication volume of the VGG16 model is relatively small, and the communication volume of the NiN model is relatively large.
3) Communication method. The average transmission rate between IoT devices is used to simulate different wireless networks. The experiment sets up 3 kinds of network environments, 3G network, WiFi and 4G network, the transmission rate is 1.1, 18.88 and 5.85 Mbps respectively.
4) Benchmark algorithms. We consider four comparison algorithms. The device-execution (DE) algorithm refers to the execution of DNN tasks only on the local device that generates the task. The maximum-execution (ME) algorithm refers to assigning DNN tasks to computing the most capable equipment. The equal-execution (EE) algorithm refers to the equal distribution of DNN tasks to all available devices. The classic shortest path Dijkstra algorithm obtains the shortest execution delay from the first to the last layer of the DNN model, and uses a single-layer prediction model to determine the weight of each edge, which is represented as short-execution (SE). The DE algorithm is used here as a benchmark and the proposed ERL algorithm is evaluated accordingly.

7.2 Forecast Model Accuracy

Delay prediction data set: For the two layer types of convolutional and fully connected layer, different parameter ranges are set respectively. The layer parameters and parameter ranges are shown in Tab. 2. Various configurable parameter sets are generated through random combination, and the configuration parameters are converted for floating-point operations, memory footprint and parameter scales and other related interpretable variables. Then, obtain the execution delay of three models of Raspberry Pi devices under different parameter settings, and determine the parameter settings and execution of the convolutional layer and the fully connected layer real time delay measurement data set. For multi-layer delay prediction, the multi-layer parameter configuration is generated according to the actual principles of DNN model. The number of layers' ranges from 1 to 40. The parameter configuration of each layer is converted into an explainable variable, and the accumulated explanatory variable is obtained by adding layer by layer, and through the tree execution of Raspberry Pi gets multiple inference delays. Based on the multi-layer parameter configuration and inferred delay data set obtained by real measurement, a variety of common prediction models are trained, and the convolutional layer and the fully connected layer are respectively predicted. The prediction performance of different prediction models is shown in Tabs. 3 and 4.

The following verifies the accuracy of the multi-layer delay prediction model of the convolutional layer. Take VGG16 and AlexNet as examples, as shown in Figs. 7 and 8, respectively, the histogram represents the actual execution delay of the experimental measurement. For example, when the abscissa is 7, it means that the delay required to execute the first seven layers of the VGG16 model is 6.08 s. The line graph shows the prediction performance of different prediction models, and the MAPE is used as the evaluation index. It can be seen from Figs. 7 and 8 that, the three prediction models of RF, GBRT and ANN can accurately predict the inference delay of any number of layers, and the average percentage error of the prediction results of any layer of the three models is less than 4%. The main reason for accurate prediction is to accurately describe the model parameters that can affect the inference delay, and map these parameters into explanatory variables such as calculation

scale and communication scale. The above three prediction models have good hierarchical fitting and generalization capabilities, and can effectively obtain the complex nonlinear relationship between feature variables and delay. In addition, further consider the impact of equipment capabilities on the inference delay, and obtain each type of equipment. The real data set of parameter configuration and execution delay, and the prediction model training for each device, so as to accurately predict the inferred delay of various devices under different parameter settings.

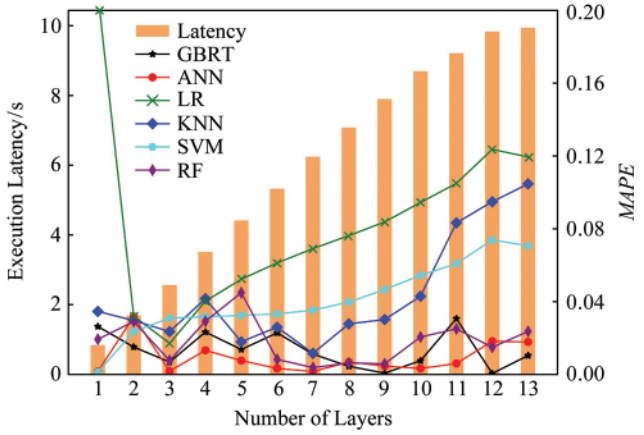

Figure 7: Comparison of latency and accuracy using VGG16

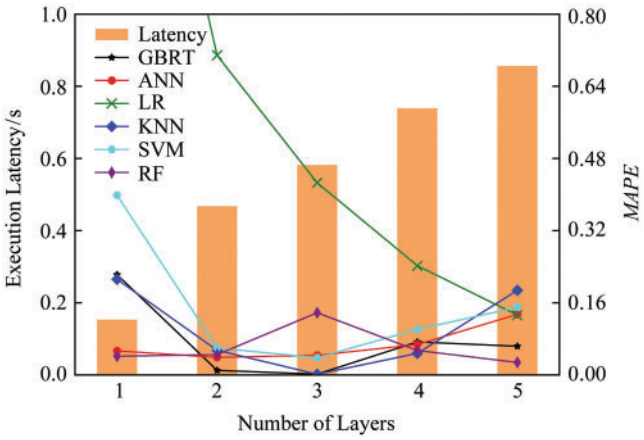

Figure 8: Comparison of latency and accuracy using AlexNet

7.3 Performance Comparison

1) DNN split. Fig. 9 shows different splitting strategies of three typical DNN models. It can be seen from Fig. 9 that, the DNN splitting strategy varies with the change of the DNN model and the number of devices. The VGG16 model has a large amount of calculation and a small amount of data transmission, so it tends to use more IoT devices to obtain better performance. The NiN model has a large amount of calculation and data transmission and excessive communication overhead cause performance degradation. Therefore, the NiN model tends to adopt fewer devices to cooperate to reduce the communication overhead. The ResNet18

model has a small amount of calculation, and it is necessary to consider whether the reduced computational overhead of collaborative inference can offset the increased communication overhead. Therefore, the split strategy of the ResNet18 model needs to weigh the computational gain and communication overhead. From this, it can be concluded that the DNN splitting strategy needs to be adaptively adjusted according to the characteristics of the DNN model and the environmental state.

2) Delayed acceleration. We compared the delay acceleration of five algorithms for different DNN models, set the number of devices to three, and the communication mode to WiFi. It can be seen from Fig. 10 that, compared with the DE, ME, EE and SE algorithms, our proposed ERL algorithm has different degrees of improvement.

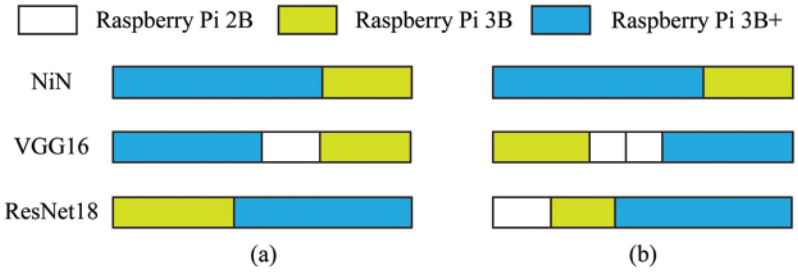

Figure 9: Partitioning comparison of algorithms

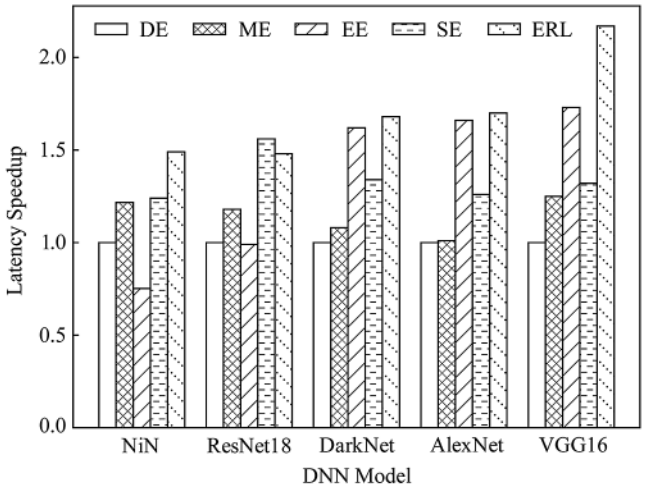

Figure 10: Comparison of latency of the proposed and existing algorithms for different neural networks models

As the computing demand increases, the performance improvement becomes more obvious. For example, the VGG16 model uses the ERL algorithm and the delay acceleration is about twice that of the DE algorithm. Mainly because of the limited resources of IoT devices, the performance of separate execution is poor when the amount of calculation is large, and the demand for DNN task splitting is stronger. However, when the amount of data transmission is large, the higher communication delay caused by DNN task splitting will seriously reduce the advantage of cooperative execution, so the delay acceleration is not obvious in the NiN model.

Since a single IoT device cannot bear the heavy computational burden, the performance of DE and ME algorithms are not ideal. Although the EE algorithm can benefit from collaborative inference, the average decision is not the optimal split strategy. Due to the inaccuracy of single-layer prediction, the performance of SE algorithm is not ideal. The proposed ERL algorithm can effectively balance the computational and communication costs, make full use of the heterogeneous capabilities of the device, and can achieve better DNN inference acceleration.

7.4 Adaptability to Environmental Conditions

1) Influence of communication status. This experiment evaluates the influence of communication status on delay acceleration. Under 3G, 4G and WiFi communication conditions, the performance of the five algorithms is compared with the VGG16 model as an example, as shown in Fig. 11. It is worth noting that, when the communication rate increases, the performance of the proposed ERL algorithm improves more significantly than the benchmark algorithms. When using a 3G network, the communication conditions are poor, and the computational gain generated by the cooperative execution is difficult to offset the communication cost generated by the data transmission. Therefore, the performance of VGG16 model EE algorithm is lower than DE algorithm. When using a 4G network, the delay acceleration of the ERL algorithm is 2.07 times that of the DE algorithm. When using WiFi for communication, the latency is increased to 2.36 times. The main reason is that, when the communication conditions are good, the data transmission delay required for DNN splitting is reduced, so the cooperative execution advantage is more obvious.

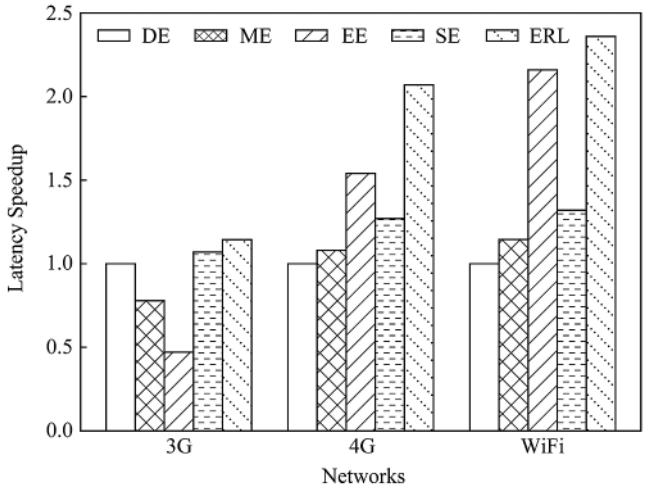

Figure 11: Comparison of latency of various communication networks of the schemes

In order to further verify that the proposed ERL algorithm can adapt to various communication states, the communication rate is set from 1 to 20 Mbps, and the VGG16 model is taken as an example to compare the delay acceleration performance of the different algorithms. Through experiments, it is found that, the performance of the proposed ERL algorithm is optimal at any communication rate, and it is inferred that the delay is reduced by more than two times. It can be seen from Fig. 12 that with the increase of communication rate, the delay acceleration becomes more obvious. This is because, the increase of communication rate can effectively reduce the communication cost caused by splitting, thereby reducing the overall execution delay. The DE, ME and EE algorithms cannot adjust

the split strategy according to the network status, so as the communication rate increases, the delay acceleration performance improvement is not obvious. The proposed ERL and the SE algorithms can effectively balance the communication and calculation overhead according to the current network state, thereby achieving significant inference acceleration as the communication rate increases, and effectively reducing the inference delay of the DNN task.

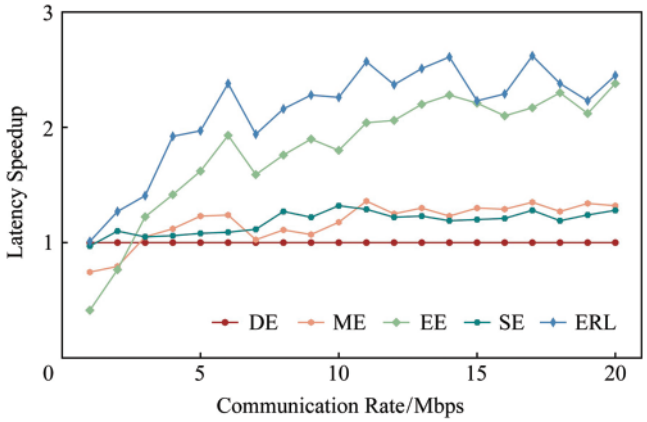

Figure 12: Latency *vs.* data rate of comparison of the proposed and existing schemes

2) Influence of the number of equipment. We deploy different numbers of IoT devices to evaluate the performance of five algorithms. Taking the NiN model as an example, it can be seen from Fig. 13 that, the proposed ERL algorithm has the best performance in terms of delay acceleration. When the number of devices is 2, 3, and 4, the delay acceleration of the proposed algorithm is 1.81 times, 1.98 times and 5.28 times that of the EE algorithm, respectively. Since the communication cost of the NiN model cannot be ignored, the EE algorithm cannot flexibly adjust the split strategy, and it is difficult to effectively weigh the calculation and communication cost. The proposed ERL algorithm can intelligently determine the splitting strategy to obtain the approximate optimal performance.

7.5 Complexity Analysis

Fig. 14 compares the computational complexity of the proposed and existing algorithms. It can be seen from Fig. 14 that, the complexity of all the algorithms increases with increasing the number of neurons in the layer. However, the complexity of the proposed algorithm is lower than all algorithms which makes it practicable and effective in the IoT-DNN deployment. As a result, the proposed algorithm outperforms the typical neural network in terms of complexity optimization.

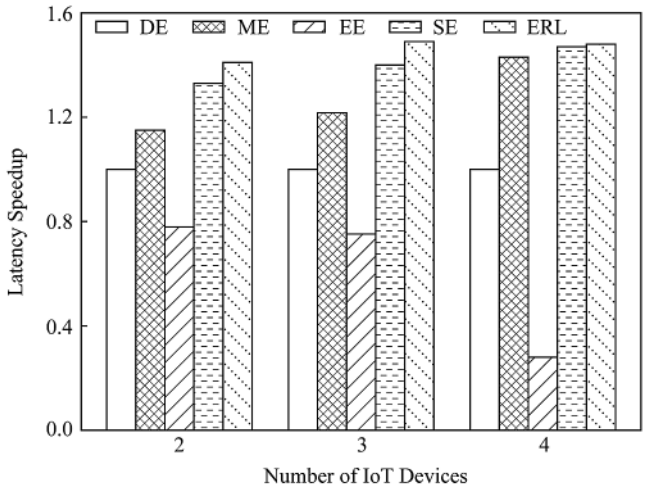

Figure 13: Comparison of latency *vs.* number of IoT devices

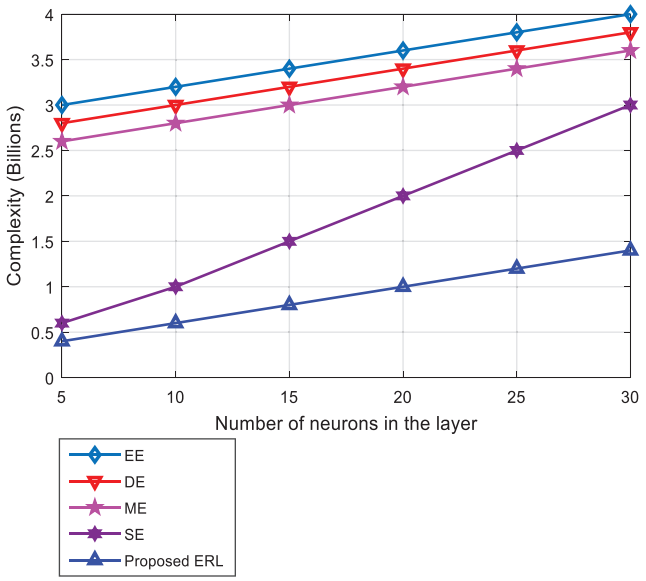

Figure 14: Complexity analysis of the algorithms

8 Discussion

1) Equipment heterogeneity. The experiment uses different models of Raspberry Pi devices to reflect the heterogeneity of the device. The performance differences of the three models of Raspberry Pi are shown in Tab. 4. Five DNN models, including AlexNet and DarkNet, are run on the three models of Raspberry Pi. Experiments of the measurement data are shown in Fig. 2. Through experiments, it can be seen that, there are obvious performance differences between the three types of equipment, which can reflect the heterogeneity of equipment. In the follow-up, we will consider various types of devices such as Raspberry Pi, mobile phones and wearable devices, and analyze the differences. The performance difference of different types

of equipment, the fine-grained modeling of equipment capabilities, on this basis, the study of cooperative inference issues between multiple types of equipment.

2) The number of equipment. The capacity of a single IoT device is insufficient, and the cooperative execution of multiple devices can effectively reduce the inference delay. However, increasing the number of cooperative devices will reduce the computational delay and increase the communication delay overhead. In order to prevent communication bottlenecks, the number of devices for cooperatively executing the DNN model will not be too much. From Fig. 9, it can be found that, for DNN models with a large amount of data transmission (such as the NiN model), even if there are more available devices, they tend to use a few devices. Even if the DNN model (such as the VGG16 model) with a small amount of data transmission and a large amount of calculation, the number of cooperative execution devices will not be too much.
3) The practicality of the IoT-CDI framework. The IoT-CDI framework mainly solves two problems: i) Aiming at the problem that the error of the existing single-layer prediction method cannot be ignored, a fine-grained multi-layer prediction method is designed, which can accurately evaluate the inference delay of any layer DNN task. ii) For equipment capabilities, the DNN task characteristics and dynamic changes in network status and heterogeneous conditions, an intelligent decision-making algorithm based on reinforcement learning is adopted. In order to overcome problems such as sparse returns and convergence difficulties, the evolutionary reinforcement learning is used to quickly obtain splitting strategies. The proposed IoT-CDI framework uses a data-driven approach to achieve accurate predictive analysis and real-time intelligent decision-making. However, compared with traditional methods, there are more obvious system overhead (requires storage models), scalability, and online adjustment. Future work will focus on the practicality of the framework to solve the problems existing in actual deployment to improve the feasibility.

9 Conclusion

This paper proposes a novel IoT-CDI framework for IoT devices to collaborate and perform DNN tasks. According to the DNN task requirements and device capabilities, a variety of factors, such as power and network status, realize real-time adaptive DNN task collaboration inference among heterogeneous IoT devices. Specifically, a multi-layer delay prediction model with different layer types, parameter configurations and device capabilities is proposed, which can accurately predict the inference delay of DNN tasks in different split situations. In addition, an intelligent DNN task splitting and collaborative inference algorithm based on evolutionary reinforcement learning is proposed, which can obtain an approximate optimal strategy in the case of heterogeneous and dynamic changes in device capabilities, network status, and task requirements. The experimental results show that, the proposed algorithm can effectively balance the communication and the calculation delay, make full use of the computing power of the equipment, and significantly reduce the DNN inference delay. In the future, we will further study the optimal value of cooperation equipment required by different DNN models. When the number of equipment is sufficient, the number of cooperation equipment can be adaptively adjusted according to the DNN task requirements to achieve the optimal inference acceleration. Future work is to address the challenges such as the scenario when the number of IoT devices is enormous and the inference delay is variable and latency is accumulating.

Acknowledgement: The authors would like to thanks the editors and reviewers for their review and recommendations.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Doss, J. Paranthaman, S. Gopalakrishnan, A. Duraisamy, S. Pal *et al.,* "Memetic optimization with cryptographic encryption for secure medical data transmission in IoT-based distributed systems," *Computers, Materials & Continua*, vol. 66, no. 2, pp. 1577–1594, 2021.

[2] S. Verma, S. Kaur, D. B. Rawat, C. Xi, L. T. Alex *et al.,* "Intelligent framework using IoT-based WSNs for wildfire detection," *IEEE Access*, vol. 9, pp. 48185–48196, 2021.

[3] A. A. Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] M. A. Garadi, A. Mohammed, A. K. Ali, X. Du, I. Ali *et al.,* "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[5] S. Li, L. Li, B. Xu, Y. Feng and H. Zhou, "Research of a reliable constraint algorithm on MIMO signal detection," *International Journal of Embedded Systems*, vol. 12, no. 2, pp. 13–26, 2020.

[6] J. Gojal, E. Monteiro and J. S. Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[7] K. Tange, M. D. Donno, X. Fafoutis and N. Dragoni, "A systematic survey of industrial internet of things security: Requirements and foq computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[8] S. Bashir, M. H. Alsharif, I. Khan, M. A. Albreem, A. Sali *et al.,* "MIMO-Terahertz in 6G nano-communications: Channel modeling and analysis," *Computers, Materials & Continua*, vol. 66, no. 1, pp. 263–274, 2020.

[9] F. Jameel, T. Ristaniemi, I. Khan and B. M. Lee, "Simultaneous harvest-and-transmit ambient backscatter communications under Rayleigh fading," *EURASIP Journal on Wireless Communications and Networking*, vol. 19, no. 1, pp. 1–9, 2019.

[10] Q. Alsafasfeh, O. A. Saraereh, A. Ali, L. A. Tarawneh, I. Khan *et al.,* "Efficient power control framework for small-cell heterogeneous networks," *Sensors*, vol. 20, no. 5, pp. 1–14, 2020.

[11] K. M. Awan, M. Nadeem, A. S. Sadiq, A. Alghushami, I. Khan *et al.,* "Smart handoff technique for internet of vehicles communication using dynamic edge-backup node," *Electronics*, vol. 9, no. 3, pp. 1–17, 2020.

[12] W. Shahjehan, S. Bashir, S. L. Mohammed, A. B. Fakhri, A. A. Isaiah *et al.,* "Efficient modulation scheme for intermediate relay-aided IoT networks," *Applied Sciences*, vol. 10, no. 6, pp. 1–12, 2020.

[13] B. M. Lee, M. Patil, P. Hunt and I. Khan, "An easy network onboarding scheme for internet of things network," *IEEE Access*, vol. 7, pp. 8763–8772, 2018.

[14] O. A. Saraereh, A. Alsaraira, I. Khan and B. J. Choi, "A hybrid energy harvesting design for on-body internet-of-things (IoT) networks," *Sensors*, vol. 20, no. 2, pp. 1–14, 2020.

[15] T. Jabeen, Z. Ali, W. U. Khan, F. Jameel, I. Khan *et al.,* "Joint power allocation and link selection for multi-carrier buffer aided relay network," *Electronics*, vol. 8, no. 6, pp. 1–15, 2019.

[16] K. Yiping, J. Hauswald, G. Cao, A. Rovinski, T. Mudge *et al.,* "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. of the ACM 22nd Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, New York, USA, pp. 615–629, 2017.

[17] T. Lawrence and L. Zhang, "IoTNet: An efficient and accurate convolutional neural network for iot devices," *Sensors*, vol. 19, no. 24, pp. 1–17, 2019.

[18] Z. Zhuoran, M. K. Barijough and A. Gertlauer, "Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2348–2359, 2018.

[19] A. E. Eshratifar, M. S. Abrishami and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2019.
[20] S. Dey, J. Mondal and A. Mukherjee, "Offload execution of deep learning inference at edge: Challenges and insights," in *IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kyoto, Japan, pp. 1–6, 2019.
[21] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu *et al.,* "Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices," in *Proc. of the ACM Conf. on Embedded Networked Sensor Systems*, New York, USA, pp. 278–291, 2018.
[22] W. Shi, Y. Huo, S. Zhou, Z. Niu, Y. Zhang *et al.,* "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, pp. 1–7, 2019.
[23] C. Hu, W. Bao, D. Wang and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE Int. Conf. on Computer Communications (INFOCOM)*, Paris, France, pp. 1–9, 2019.
[24] H. Mao, X. Chen, K. Nixon, C. Krieger and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," *Design, Automation & Test in European Conference and Exhibition*, pp. 1–6, 2017. https://ieeexplore.ieee.org/document/7927211.
[25] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das *et al.,* "Scalpel: Customizing dnn pruning to the underlaying hardware parallelism," in *Proc. of the Int. Symposium on Computer Architecture (ISCA)*, Toronto, Canada, pp. 548–560, 2017.
[26] S. Han, H. Mao and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Computer Vision and Pattern Recognition Conf.*, New York, USA, pp. 3–17, 2016.
[27] L. Wang, W. Liu, X. Liu, G. Zhong, P. Roy *et al.,* "Compressing deep networks by neuron agglomerative clustering," *Sensors Journal*, vol. 20, no. 21, pp. 1–18, 2020.
[28] R. Wang, W. Zhang, J. Ding, M. Xia, M. Wang *et al.,* "Deep neural network compression for plant disease recognition," *Symmetry Journal*, vol. 13, no. 10, pp. 1–19, 2021.
[29] J. Park, S. Lee and D. Jeon, "A neural network training processor with 8-bit shared exponent bias floating point and multiple-way fused multiply-add trees," *IEEE Journal of Solid-State Circuits*, vol. 8, no. 3, pp. 874–883, 2021.
[30] R. Wu, X. Guo, J. Du and J. Li, "Accelerating neural network inference on fpga-based platforms—A survey," *Electronics Journal*, vol. 10, no. 9, pp. 1–16, 2021.
[31] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter *et al.,* "Energy-efficient offloading for DNN-based smart iot systems in cloud-edge environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 683–697, 2021.
[32] D. R. Torres, C. Martin, B. Rubio and M. Diaz, "An open source framework based on kafka-ml for distributed dnn inference over the cloud-to-things continuum," *Journal of Systems Architecture*, vol. 118, no. 4, pp. 973–984, 2021.
[33] H. Zhou, W. Zhang, C. Wang, X. Ma and H. Yu, "BBNet: A novel convolutional neural network structure in edge-cloud collaborative inference," *Sensors Journal*, vol. 21, no. 13, pp. 1–26, 2021.
[34] L. Fleischer, M. X. Goemans, V. S. Mirrokni and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," *Mathematics of Operations Research*, vol. 36, no. 3, pp. 416–431, 2011.
[35] J. Yi, Q. Zhuge, J. Hu, S. Gu, M. Qin *et al.,* "Optimizing task assignment for heterogeneous multiprocessor system with guaranteed reliability and timing constraint," in *IEEE 19th Int. Conf. on Embedded and Real-Time Computing Systems*, Taipei, Taiwan, pp. 193–200, 2013.
[36] K. Arukumaran, M. P. Deisenroth, M. Brundage and A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
[37] V. Mnih, K. Kavukcuoglu, D. Silver and A. Rusu, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[38] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.
[39] M. Long, N. Dao and M. Park, "Real-time assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing," *Sensors Journal*, vol. 18, no. 9, pp. 2830–2848, 2018.
[40] D. D. Fan, E. A. Theodorou and J. Reeder, "Model-based stochastic search for large scale optimization of multi-agent uav swarms," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, Banglore, India, pp. 915–921, 2018.