Data Gathering System Based on Multi-layer Edge Computing Nodes

Shuzhen Xiang, Huigui Rong(✉), and Zhangchi Xu

Hunan University, Changsha, China
{shuzhenxiang,ronghg,zhangchixu}@hnu.edu.cn

Abstract. The development of Internet of Things technology brings new opportunities for the development of edge computing. As an emerging computing model, edge computing makes full use of the equipment resources at the edge of the network and creates a new network computing system at the edge of the network. At the same time, the emergence of edge computing solves the problem of high latency in WAN which cannot be solved for a long time in the field of cloud computing, and brings users with low latency, fast response and good service experience. This article will use the edges computing architecture to construct a multi-layer data collection system. In this system model, sensors upload data to the designated edge nodes for processing, rather than remote cloud computing centers. Data collection and sample training tasks of sensor nodes in different ranges are realized through the design of multi-layer edge nodes. This system reduces the energy consumption of data uploading and the delay in network communication. As a result, it provides a better network experience for the end users. And it tries to solve the problem that the edge node in the edge system cannot satisfy multiple training task requests at the same time.

Keywords: Data gathering · Edge computing · Compress sense

1 Introduction

In general, after collecting the data generated by the terminal devices, the cloud data center uses powerful computing power to train the AI service model, and then stores the trained model on the cloud and the service devices at the edge. However, using cloud data center to collect data will cause two very tricky problems. First, cloud data center is usually far away from the end devices, and the end devices need to be connected through wide area network (WAN), resulting in a large delay. Secondly, when users' data is transmitted in a WAN, data privacy cannot be guaranteed due to the fact that WAN covers a wide range of regions and there may be many intermediate nodes [1]. In the edge of computing architecture, the edge nodes have proximity. That is to say, the geographic distance between the edge node and the end device is very close, which is much smaller than the distance between the end device and the cloud data center. At the same time, using wireless LAN for network connection, the propagation delay is much smaller than in WAN. Based on the wireless local area network to transmit

H. Jiang et al. (Eds.): ICECI 2020, LNICST 368, pp. 51–64, 2021.
https://doi.org/10.1007/978-3-030-73429-9_4

data, there is no need to worry about some relatively confidential data being leaked. So the network system based on edge computing architectures constructing can well solve the two problems. In addition, the analysis engine is close to the data source and only a small amount of condensed information is sent back to the central system, which is more effective.

However, in the edge system, there may be many different AI service models that need to be trained, but the relatively scarce computing resources and storage resources of the edge devices cannot meet the requests of all training tasks at the same time. As a result, this article puts forward a kind of multi-layer edge data collection system, which makes the data can be collected and trained on the edge of the network side, in order to complete the training tasks of the AI service model as much as possible with limited resources.

2 Related Work

With the emergence of new computing models such as IoT, edge computing, and fog computing, some research efforts have attempted to implement distributed machine learning model training in network edge devices with more geographical locations [1–4]. These network edge devices are usually connected to each other through wireless LAN. Moreover, compared with the server cluster in the data center, the computing and storage capacity of a single network edge device is more limited. These factors have brought challenges to the training of distributed machine learning in the edge network. Reference [2] considers a distributed machine learning model training algorithm for parameter aggregation through a central parameter server. This method does not need to transfer the original data to the central server. Instead, it uploads the local model parameters on each distributed node after the aggregation. Based on the theoretical analysis of the convergence rate of distributed gradient descent, a control algorithm is proposed to determine the optimal tradeoff between local update and global parameter aggregation to minimize the loss function under a given resource budget. Reference [3] proposes to divide each layer of the deep neural network model (DNN) into several parts, which are mapped to various hierarchical structures in the edge computing architecture. Through joint training of each part, the network edge can quickly generate small neural network model (that is, a neural network model with fewer parameters) for quick task inference, while generating a larger neural network model (that is, a neural network model with more parameters) on the cloud data center side for more accurate tasks inference. It is inferred that this joint training method minimizes device communication and resource utilization, and improves the practicality of feature extraction in a cloud computing environment. Reference [4] introduces edge computing into IoT applications based on deep learning, and proposes a new scheduling strategy to optimize the performance of deep learning applications in the IoT through the edge computing architecture. However, in the case of training distributed machine learning models to attract edge computing architecture, it is not covered that how to effectively collect data from terminal devices and offload tasks and correctly allocate limited resources for each task. Since machine learning is usually resource-intensive and time-consuming, the data collection mechanism and resource allocation mechanism will have a great impact on the accuracy and training efficiency of the training model. Therefore, it is essential to tailor an effective data collection mechanism and an appropriate resource allocation mechanism for the training of AI service model in the edge.

For data collection in distributed networks, many scholars at home and abroad have also proposed the principle of using compressed sensing to sample and compress data. Reference [5–8] studies the distributed source coding technology of multi-sensor collaboration. The source coding algorithm based on compressed sensing is adopted to reduce the repetitive coding of data, improve the compression ratio and save energy. In recent years, a large number of research achievements have been made on compressed sensing itself and its application in medical detection, radar imaging, image processing and other fields.

3 Theoretical Basis

3.1 Compressed Sensing

With a large amount of information growing now, the compressed sensing method breaks through many limitations of traditional methods in data sampling, storage, and signal bandwidth. Sampling and compression are completed at the same time without losing the original information, which saves a lot of resources. Compressed sensing theory proves that if the signal satisfies the sparse characteristics in an orthogonal transform space, the original signal can be reconstructed accurately or with high probability through fewer sampling points. Suppose there is a signal $f^{(N\times 1)}$, the length is N, and the basis vector is Ψ_{i} ($i = 1, 2, \ldots, N$) to transform the signal.

$$f = \sum_{i=1}^{N} \chi_i \psi_i \ \text{ or } f = \Psi \mathrm{X} \tag{1}$$

From the above, the current theory of compressed sensing mainly involves three aspects:

(1) Sparse representation of signals;
(2) Design of measurement matrix;
(3) Design of reconstruction algorithm.

3.2 Random Walk

In computer, physics, biology and other fields, random walk has been widely used. More and more attention has been paid to the random walk model. “Random Walk” describes a situation in which a person standing on a straight line in three dimensions has only two directions to choose from, and now he can only choose to go left or right. In probability theory, he takes one step to the left as much as he takes one to the right, and when he takes enough steps he must come back to where he started.

First, the introduction of random walk on the graph: define the graph G (V, E, ω), G (V, E, ω) is a right undirected graph with n vertices and m edges, where V is the vertex set, E is the edge set, and ω: $V \times V \rightarrow R$ is the connection weight function. As shown in Fig. 1:

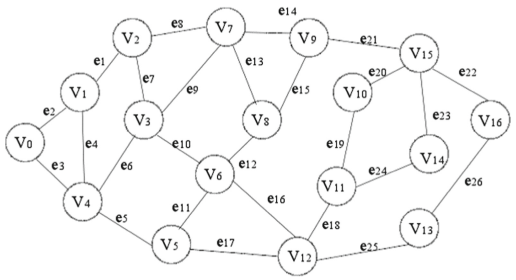


Fig. 1. Undirected graph

Figure 1 an undirected graph with 17 vertices {V_0, V_1 ....., V_{16}}, and 26 edges { e_1, e_2, e_3...... e_{26}}. Each edge is given a certain weight ω_i..

If particle A goes from vertex V_i to vertex V_j with probability $P_{ij,}$ then the vertices visited by particle A form a random sequence X_n, $n = 0,1,2,\ldots$

4 Multi-layer Edge Data Collection System

This system combines the topology of the network with the compressed sensing and the edge server, and sets the cluster and cluster head in the edge server in a fixed structure. This paper mainly describes how to set up the edge network structure and how to realize data collection on this structure. The logic diagram of the specific framework is shown in Fig. 2.

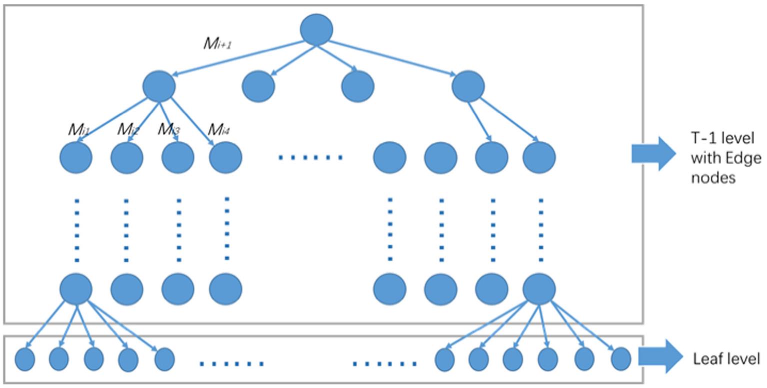


Fig. 2. System framework logic diagram

4.1 System Model and Problem Description

The network architecture in Fig. 2 is adopted in this paper. The first layer is leaf level, where each node is a data source node and corresponds to an area with a geographical area of s. The i $(i >= 2)$ layer is the edge computing layer, which is composed of multiple edge computing nodes.

Each edge computing node of the i-th layer can collect and process the data of the n nodes of the specified $I - 1$th layer and upload it to the specified edge node of the $i +$ 1th layer.

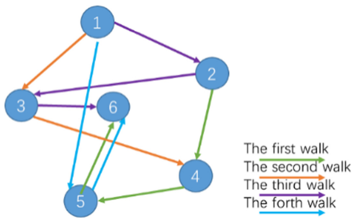
$C_{i,n}$ represents the nth node in the i-th layer, and $M_{i,n}$ represents the measured number of the nth node in the i-layer ($n = 1, 2, \ldots N_i$, N_i is the number of nodes shared in the i-th layer). $C_{i,n}$ has a data amount of $D_{i,n}$ and generates a training task request $q_{i,n,k}$.

When faced with the assignment of training tasks. Each edge node of the system can regard the upper node in its own cluster as the source node that can send training requests. At the same time, in the training model of AI, for the same model, the greater the amount of input data during training, the higher the accuracy of the training model, and otherwise the effect is very poor. Therefore, our model has the potential to improve the accuracy of the model by actively collecting more data.

4.2 Data Collection Process

Level 1: Source Node Data Collection

Random Walk. We correspond the vertices of the undirected graph to the sensor nodes in the wireless sensor network, and apply the random walk model to the first-level leaf nodes of the system. We can define the starting node of the Random Walk randomly or according to some properties of the signal, set the number of steps to be taken by the Walk, and then generate a random sequence of access paths *Xn*. In a vector of $1 \times N$ with all zeros, set the position of the path to 1, which means that the information of this node is collected. If the node has been passed in this path, the value in the vector is not changed, indicating that the information of this node is not collected. After a walk, the last node will pass the collected data to the cluster-head node. In this way, a path can be transformed into a 0/1 vector, and the 0/1 vector of M paths can be generated by repeating M random walks. These vectors are combined together to form an $M \times N$ matrix, which is the measurement matrix of compressed sensing, so as to collect and measure information. The logical diagram is shown in Fig. 3:



$$M = \begin{pmatrix} 0,1,0,1,1,0 \\ 1,0,1,1,0,0 \\ 1,1,1,0,0,1 \\ 1,0,0,0,1,1 \end{pmatrix}$$

Fig. 3. Random walk roadmap

There are 6 nodes in the figure, and each color represents a node that a walk passes through. A total of 4 walks are made. The resulting measurement matrix is M.

In the measurement matrix formed by Random walk, each row vector is independently generated, and the data collection process is also collected through one by one path. These characteristics make the use of stopping rule to ensure the quality of data with higher efficiency. Increasing the number of samplings at a time can be achieved by simply increasing the number of other walks without re-sampling.

Stopping Rule to Ensure Data Quality. Since the data information of the node is unknown, the sparsity of the information cannot be accurately estimated, which brings certain problems to the setting of the sampling rate. If the sampling rate is set too high, both communication overhead and computational overhead are wasted. And if the sampling rate is set too low, the quality of the reconstructed data cannot be guaranteed.

First, let's look at the sampling rate and the quality of data reconstruction. Figure 4 shows the relationship between the change of sampling rate and the quality of signal reconstruction when sampling a certain signal. As the sampling rate increases, the quality change of signal reconstruction tends to be stable. However, different types of signals tend to stabilize at different sampling rates.

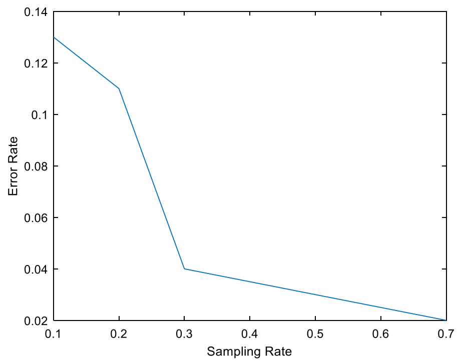


Fig. 4. The relationship between the change of sampling rate and the quality of signal reconstruction when sampling a certain signal.

Stop Rule: compare the reconstructed *x(m)* and *x(m + 1)* when the number of measurements is *m* and the number of measurements is *m + 1*, if $x(m) \underset{\approx}{\overset{\Delta}{}} x(m+1)$, stop increasing the number of measurements and upload the measurement results of *m + 1* to the next layer; if the conditions are not met, increase the number of measurements once, update the value of *m*, and compare again, until the conditions are met. Then, stop increasing the number of measurements. $x(m) \underset{\approx}{\overset{\Delta}{}} x(m+1)$ is expressed as:

$$\frac{\sqrt{\sum (x'(m-1)_{ij} - x'(m)_{ij})^2}}{\sqrt{\sum (1/2(x'(m-1)_{ij} + x'(m)_{ij}))^2}} \leq \varepsilon \tag{2}$$

To judge whether the two matrices have reached the stability of the reconstruction effect, ε is a small constant.

Saliency Influence the Choice of the Starting Point of the First Floor Walk. Saliency is the recognition of local mutations in the relevant overall data. Therefore, we introduce the concept of saliency into the sensor data processing.

Since random walk is a randomly walks in the network structure of nodes, each newly added walk in the above termination criterion is arbitrary, which affects the efficiency of the entire reconstruction to a certain extent. If you can choose a walk with the best improvement in overall reconstruction efficiency in the next walk, you can further save the amount of calculation and upload data consumption. The implementation steps are as follows:

(A) Divide the nodes in a cluster into n blocks on average;
(B) According to the reconstruction results of the previous m measurements, calculate the saliency value S_i of each block separately;
(C) Measure the weight of each block according to the saliency value of each block $w_i = \frac{S_i}{\sum_{i=1}^{n} S_i}$;
(D) According to the size of the saliency value, assign the block from which the walk starts to walk, $P_i = w_i$;
(E) Generate a new walk.

Algorithm Design of the First Layer

Input: The sampling rate of pre-sampling is R_{pre}, the initial sampling rate R_c, step t, the number of new walks g each time, the stop condition threshold ε;
Step 1: Randomly select the starting point of the random walk of the leaf nodes with the pre-sampled sampling rate R_{pre}, to obtain the pre-sampled measurement matrix M_{pre};
Step 2: Reconstruct the data according to the pre-sampling and use the reconstruction algorithm to obtain the matrix X_{pre};
Step 3: Divide X_{pre} into n blocks and calculate the saliency value S_i of each block;
Step 4: Assign the weight w_i of each block according to the saliency value, and calculate the starting point probability P_i of each block;
Step 5: Select the starting point of random walk according to P_i, and then generate the measurement matrix M_{add} of $(R_c - R_{pre})$ sampling rate to obtain the final initial measurement matrix $M_c = \begin{pmatrix} M_{pre} \\ M_{add} \end{pmatrix}$;
Step 6: According to the measurement matrix M_c, the reconstructed X_c is obtained, and then g new walks are generated by Step 3 and Step 4, and the new walk is added to the measurement matrix to update the measurement matrix M;
Step 7: Compare the reconstructed $X(m)$ and $X(m + 1)$ when the measurement times are m and the measurement times are $m + g$, if $X(m)$ and $X(m + 1)$ meet the conditions formula 2, then stop increasing the number of measurements, and upload the data to the cluster head of the second layer; if the conditions are not met, update $m = m + g$, convert $Xpre$ in step 3 to $X(m)$ and repeat after generating g walks.

Layer i: Data Collection Edge Layer

Saliency Affect the Sampling rate Distribution of the i ($2 < i < N$) Layer. In the actual situation, the data uploaded by the sensor node is not known, and it is not smooth, but with mutations. When faced with an unsmooth signal, the number of measurements may exceed the total number of nodes, resulting in an increase in the amount of communication data. Therefore, in the *i*-th layer $(2 < i < n)$ of the entire system, this paper introduces a weighting factor γ to evaluate the value of the data carried by the edge nodes, and updates the measurement value of each edge node to $M'^{(l)} = \gamma * M^{(l)}$, so that each cluster can be adaptively measured.

Because the fixed sensor node is limited by regional time, within a certain period of time, the data acquired by the sensor has certain similarity. All can calculate γ based on the data obtained last time.

Suppose that the data of four edge nodes in layer *i* will be transferred to an edge node in layer $i + 1$. The data of these four nodes are represented by vectors *x1, x2, x3,* and *x4*, respectively, and the corresponding measured values are *M1, M2, M3,* and *M4.* From *x1, x2, x3,* and *x4*, using the features of saliency, the saliency values of the four vectors are calculated, and the corresponding saliency values are obtained as *S1, S2, S3,* and *S4.* The γ value corresponding to the *j*th value is

$$\gamma_{\mathrm{j}} = \frac{S_j}{S_{\mathrm{avg}}} \tag{3}$$

Where S_{avg} is the saliency average of this layer,

$$S_{\mathrm{avg}} = \frac{\sum_{\mathrm{n=1}}^{\mathrm{k}} S_{\mathrm{i}}}{\mathrm{k}} \tag{4}$$

Where *k* is the number of clusters contained in this layer, so that the corresponding weighted value for the next retransmission can be obtained.

Algorithm Design of Layer *i*($2 < i < N$)

Input: the original assigned $M_i{}^{(l)}$ of the measured value of each cluster;
Step 1: Calculate the saliency value *Si* of each cluster in the *i*-th layer under this cluster head according to the data reconstructed by the cluster head in the $i + 1$th layer;
Step 2: According to the saliency value *Si*, calculate the weighting factor γi of the value of the data carried by the node;

$$\gamma_{\mathrm{i}} = \frac{k \cdot S_i}{\sum_{n=1}^{k} S_n} \tag{5}$$

Step 3: Update the measured value of each cluster $M_i'^{(l)} = \gamma_i * M_i{}^{(l)}$;
Step 4: Use the updated measurement value to perform compression measurement on the data of each cluster.

4.3 Train the Task Assignment Process

Since the training task does not necessarily have frequent periodicality along with the data collection, in addition to the function of data uploading, edge computing nodes can also make use of idle time and collected data to conduct offline training for existing models. This can improve the real-time performance of the request and reduce network pressure.

Then, when the edge computing node receives the training request, it may not accept and respond immediately. First of all, it is necessary to judge whether the training of the request is feasible by combining the requested data volume with the current computing resources. Training to each edge node C_{in} $(i > 2)$ the maximum storage capacity and computing power of rs_{in} and rc_{in}, respectively, at the same time, under the assumption that each data source node generated training mission need to request a unit of the CPU to calculate, we can get about edge training node storage capacity and computing capacity constraints, the sum of all the tasks of computing power and storage capacity of no more than the sum of node threshold.

After the training task of the edge computing node is completed and the updated model is obtained, the training request information is summarized and uploaded to the designated edge computing node at the next level of the system for model training with a larger area. Therefore, the closer the system is to the source node, the more real-time tasks it may have to face. Therefore, it is very likely that there will be insufficient computing capacity of edge nodes and insufficient resources. Therefore, we need to adjust our thought. In case of insufficient computing and resources, the request should be transferred to the next layer. In order to ensure real-time performance and low latency, the number of layers should not be more than 3. The specific algorithm is as follows (Fig. 5):

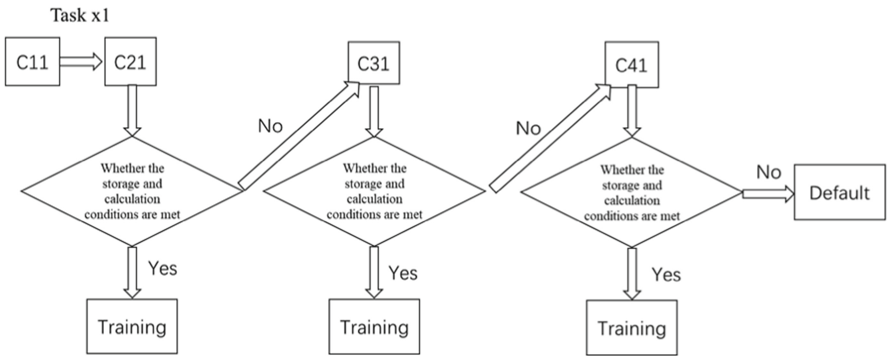

Task x1
C11
C21
C31
C41
No
No
No
Whether the storage and calculation conditions are met
Whether the storage and calculation conditions are met
Whether the storage and calculation conditions are met
Default
Yes
Yes
Yes
Training
Training
Training

Fig. 5. Train the task assignment process

5 Analysis of Experimental Results

5.1 Performance Measures

(1) Quality of collected data
The collected data quality will directly affect the accuracy of the training model. In this paper, the error rate and SNR are used to represent the data quality.

(2) Communication overhead
Sensors in a small area can generate as much as 8 GB of data a day, and that doesn't include data generated by mobile devices, surveillance cameras, and Web services. Therefore, in modern cities, the amount of data generated in one day will be very large, and the amount of data collected will directly affect the communication cost. The larger the amount of data, the greater the cost. In this paper, the data volume of the first layer is related to walk number. Each walk generates one unit of data. The generation of walk is directly determined by in the stopping rule, so the communication overhead is indirectly expressed as the relationship between walk number and ε.

(3) Training task volume
For each training task request, the system proposed in this paper will determine whether the training task is accepted by the edge node and which edge node should collect the task and its data for training. Depending on the goal of the problem, we should consider the number of tasks the system accepts as a performance measure. Different variables such as the storage capacity of edge nodes and the number of data source nodes will affect the number of training task requests received by the system.

5.2 Experimental Setup

In this paper, the experiments use the experimental data sources (https://tao.ndbc.noaa.gov/tao/datadownload/searchmap.SHTML) of sea surface temperatures. In addition, two groups of signals collected under other conditions are simulated for the experiment. The second group of signals is sparse signal, and the third group of signals are mutated non-sparse signal.

In this paper, a total of 1024 nodes are set, and the total structure is divided into 4 layers. The first layer is divided into 16 clusters, each with 64 nodes. The second layer is divided into 4 clusters, each cluster contains 4 cluster-head nodes. The four cluster-head nodes in the third layer upload the data to the data processing center in the fourth layer. In the first layer, the pre-sampling rate R_{pre} is set to 0.1, the initial sampling rate R_c is set to 0.3, the random walk skip is set to 30, and stopping rule's only selects 1 walk at a time. Measure the m value of the amount of data uploaded to the second layer when the recording stops. In the second layer compression of the algorithm in the paper [9], the same measured value M is used to compress and upload to the third layer, and the sampling rate after the third layer is set at the same sampling rate of 0.9.

We assume that the size if data uploaded by a node is 0.25 MB each time, and the task data collected by a training task is 4 GB, in this paper, data storage capacity of the edge of the tree nodes are assumed to be [10,100] of the GB uniform distribution, the amount of computing power to vCPU decision, to obey [50, 150] vCPU evenly distributed.

5.3 Comparison of Experimental Results

(1) Influence of ε

In the experiment on the influence of ε, we use 1024 data, the predetermined R_{pre} is 0. 3, and the corresponding $M_{pre} = 307$.

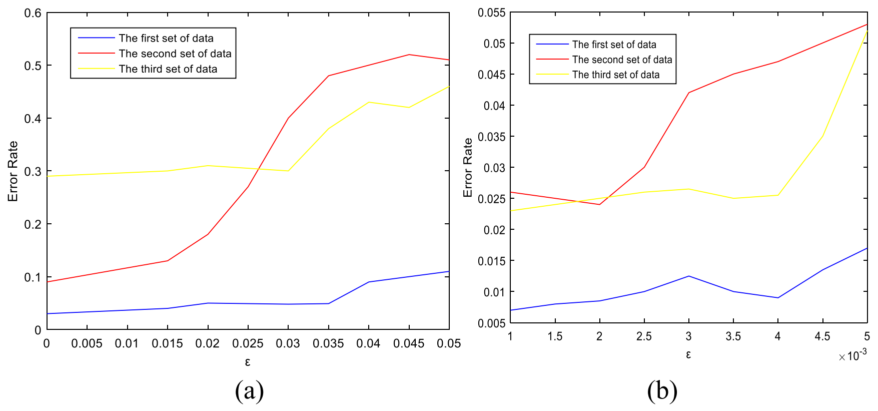


Fig. 6. The effect of ε on the reconstructed data Error Ratio

For different signals, the setting of ε affects them to different degrees, but they all directly affect the quality of the reconstructed data when the algorithm stops. In Fig. 6(a), the setting range of ε is 0.01–0.05, and the error ratio is higher than 0.1. The larger the ε setting, the larger the error ratio, which indicates that the quality of the data increases with ε. In the figure (b), the setting range of ε is 0.0001 to 0.0005, and the error ratio is lower than 0.01, achieving high-quality reconstruction.

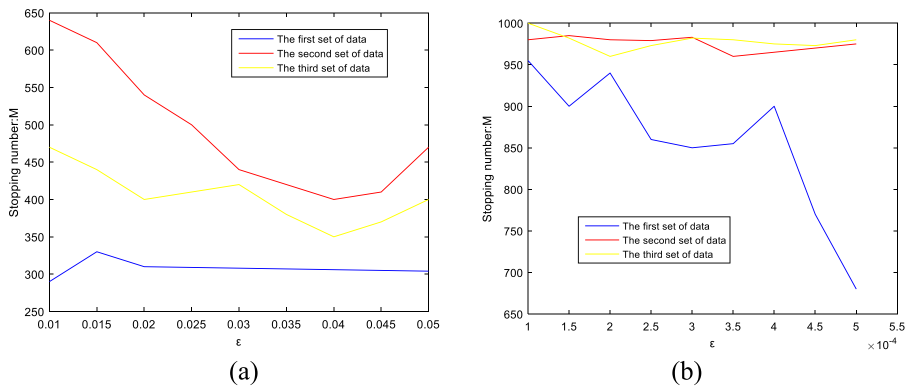


Fig. 7. The effect of ε on the value of M

As ε increases, the value of M at the time of stopping also becomes smaller. The smaller the value of M, that is, the less data need to be uploaded, which directly affects the overall communication overhead. The setting range of ε in Fig. 7(a) is 0.01–0.05, and the range of M value is basically between 300–650, which is equiva-lent to the sampling rate between 0.3–0.6; (b) In the figure, the setting range of ε is 0.0001–0.0005, the M range is basically between 750–1000, which is equivalent to the sampling rate between 0.7–0.9.

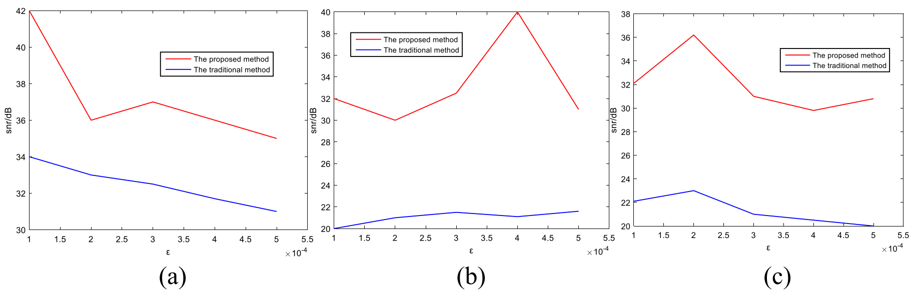


Fig. 8. Snr comparison chart of this algorithm and traditional algorithm

It can be seen from Fig. 8 that, overall, the algorithm in this paper significantly improves the data quality compared to the traditional algorithm. Especially for the two sets of signal data after the mutation, the improvement of the effect is more obvious.

(2) Impact on training business volume

This is mainly for comparison with the most intuitive way of collecting greedy thoughts. According to greedy theory, in order to collect more training task requests, the system should first select those training task requests that provide the least amount of data. Under the greedy strategy, we first select the training task request with the smallest amount of data, and then randomly put this training task request and the data it provides into an edge node. The total number of edge nodes in the greedy thought and the number of all edge nodes in the system.

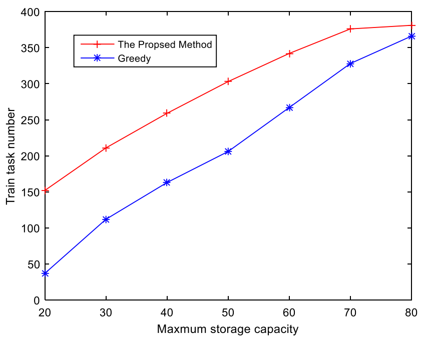


Fig. 9. Impact on training business volume

Figure 9 shows the impact of the storage capacity of the edge training node on the number of training task requests received by the system. As the upper limit of the storage capacity of the edge training node increases, the performance of the system is getting better and better, probably more than the greedy strategy collect about 30% of training mission requests.

6 Summary

This paper proposes a data collection system based on distributed edges, which uses random walk and stopping rule and selects the starting point of walk based on saliency, and applies it to the data collection of the leaf layer structure, so that each cluster can use edge nodes to adapt. To collect data, save the amount of uploaded data while ensuring data quality. In the compressed sampling process of the i-th layer ($2 < i < n$), the sampling rate of each cluster is allocated again according to the reconstructed data obtained in the previous time using saliency, so as to better identify the sudden change area and avoid the situation that the measured value in the abrupt region is greater than the number of cluster nodes, thus wasting the amount of uploaded data. In addition to the cost savings in the data collection process, the system proposed in this paper can better deal with the data collection of edge nodes for AI data training. The data and model of each edge node are closer to the data characteristics within the range of the node. And when the computing power or storage capacity of the edge node is insufficient, the training request can be passed to the edge node of the next layer to process as many training requests as possible. In the future research, the problem of unloading and scheduling of edge training tasks will be analyzed based on more specific scenarios.

Acknowledgments. This research was partly supported by National Natural Science Foundation of China under Grant No. 61672221, and by National Natural Science Foundation of Hunan Province under Grant No. 2020JJ4008.

References

1. Mao, Y., You, C., Zhang, J., et al.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutorials **PP**(99), 1 (2017)
2. Wang, S., Tuor, T., Salonidis, T., et al.: When edge meets learning: adaptive control for resource-constrained distributed machine learning. In: IEEE INFOCOM 2018 IEEE Conference on Computer Communications. IEEE, pp. 63–71 (2018)
3. Teerapittayanon, S., Mcdanel, B., Kung, H.: Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp. 328–339 (2017)
4. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Netw. **32**(1), 96–101 (2018)
5. Xu, X., Ansari, R., Khokhar, A., Vasilakos, A.V.: Hierarchical data aggregation using compressive sensing (HDACS) in WSNs. ACM Trans. Sens. Netw. **11**(3), 1–25 (2015). Article 45
6. Chong, L., Jun, S., Feng, W.: Compressive network coding for approximate sensor data gathering. In: Global Telecommunications Conference, IEEE Press, pp. 1–6 (2011)

7. Luo, C., Wu, F., Sun, J., Chen, C.W.: Compressive data gathering for large-scale wireless sensor networks. In: Proceedings of MobiCom (2009)
8. Luo, J., Xiang, L., Rosenberg, C.: Does compressed sensing improve the throughput of wireless sensor networks. In: Proceedings of the IEEE International Conference on Communications (2010)
9. Zheng, H., Yang, F., Tian, X., Gan, X., Wang, X., Xiao, S.: Data gathering with compressive sensing in wireless sensor networks: a random walk based approach. IEEE Trans. Parallel Distrib. Syst. **26**(1), 35–44 (2015)
10. Wang, L., et al.: CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing. In: UBICOMP 2015, Osaka, Japan, 7–11 September 2015
11. Kang, K.D., Chen, L., Yi, H., et al.: Real-time information derivation from big sensor data via edge computing. Big Data Cogn. Comput. **1**(1), 5 (2017)
12. Shi, W., Cao, J., Zhang, Q., et al.: Edge computing: vision and challenges. IEEE Internet Thing J. **3**(5), 637–646 (2016)
13. Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., et al.: Estimating smart city sensors data generation. In: 2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net). IEEE, pp. 1–8 (2016)
14. Sivakumaean, M, Iacopino, P.: The mobile economy 2018. GSMA Intelligence, pp. 1–60 (2018)