Article

A Partition Based Gradient Compression Algorithm for Distributed Training in AIoT

Bingjun Guo, Yazhi Liu * and Chunyang Zhang

Department of Computer Science and Technology, North China University of Science and Technology, Tangshan 063210, China; youngwind1995@gmail.com (B.G.); zchunyang571@gmail.com (C.Z.)

* Correspondence: liuyazhi@ncst.edu.cn

Abstract: Running Deep Neural Networks (DNNs) in distributed Internet of Things (IoT) nodes is a promising scheme to enhance the performance of IoT systems. However, due to the limited computing and communication resources of the IoT nodes, the communication efficiency of the distributed DNN training strategy is a problem demanding a prompt solution. In this paper, an adaptive compression strategy based on gradient partition is proposed to solve the problem of high communication overhead between nodes during the distributed training procedure. Firstly, a neural network is trained to predict the gradient distribution of its parameters. According to the distribution characteristics of the gradient, the gradient is divided into the key region and the sparse region. At the same time, combined with the information entropy of gradient distribution, a reasonable threshold is selected to filter the gradient value in the partition, and only the gradient value greater than the threshold is transmitted and updated, to reduce the traffic and improve the distributed training efficiency. The strategy uses gradient sparsity to achieve the maximum compression ratio of 37.1 times, which improves the training efficiency to a certain extent.

Keywords: AIoT; distributed training; gradient compression; training efficiency

Citation: Guo, B.; Liu, Y.; Zhang, C. A Partition Based Gradient Compression Algorithm for Distributed Training in AIoT. *Sensors* **2021**, *21*, 1943. <https://doi.org/10.3390/s21061943>

Academic Editor: Francesco Marcelloni

Received: 8 January 2021
Accepted: 6 March 2021
Published: 10 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid development of the Internet of Things (IoT), it has become a trend to apply AI technology to the IoT, and creates a new concept, AIoT (Artificial Intelligence & Internet of Things). AIoT refers to the application of artificial intelligence (AI) algorithms in the Internet of Things (IoT) to achieve a higher level of the intelligent system of the Internet of Things. In an AIoT system, it is usually composed of many sensors and cloud servers to collect, store and process data. At the same time, with the rapid development of the information age, higher requirements are put forward for the data processing and low delay communication of equipment in the AIoT system. However, due to a large number of sensors, small size, and limited computing power, the ability to access and communicate with each other is limited, and it is necessary to extend their service life and minimize the number of maintenance [1,2]. Therefore, how to improve the communication efficiency between the sensor nodes and improve the overall serviceability has become one of the urgent problems to be solved.

In this paper, we model the above problem as a distributed training problem in deep learning. As shown in Figure 1, the left half of Figure 1 describes the AIoT architecture, where multiple sensors are connected to Cloud Computing through the network layer. In the right half of the figure, we map AIoT to a distributed training model. Each sensor in AIoT is equivalent to a worker node in a distributed training scene, which is responsible for calculating the gradient and transmitting it to the server; and Cloud Computing in AIoT is equivalent to a server node in a distributed training scene, responsible for summarizing global gradients and updating model parameters. Due to the relatively weak communication capabilities of IoT devices, the gradient exchange efficiency between various nodes has become one of the important bottlenecks that restrict the quality of

AIoT. To solve this problem, this paper proposes a gradient partition-based compression algorithm, Partition Gradient Compression (PGC). PGC can improve the overall training efficiency by reducing the communication cost between distributed nodes and improving communication efficiency, thereby improving the efficiency of distributed training in AIoT.

In recent years, deep learning technology has attracted much attention due to its outstanding performance in many artificial intelligence fields such as image recognition, natural language processing, and speech recognition. With the increasing scale of deep learning and the increasing complexity of neural network structure, distributed deep learning becomes more and more important in research and practice. The communication efficiency of gradient exchange is a major obstacle to the improvement of distributed training efficiency [3], therefore, how to improve the communication efficiency of distributed training is also one of the hot spots in current research.

Figure 1. The training scene in IoT is modeled as distributed training.

One of the most direct ways to deal with communication costs is to reduce traffic, or gradient compression [4]. Solutions based on gradient compression can be divided into two categories: gradient quantization and gradient sparsity. Gradient quantization reduces communication costs by using fewer bits to reduce the number of gradients transmitted. However, the compression ratio based on the quantization method is limited and can only be compressed to 1-bit at most.

The sparse method is to select a threshold value and send gradient elements larger than the threshold value, while gradient values smaller than the threshold value are not sent. Although this method can obtain a high compression ratio, it often damages the accuracy of the model. For example, Han et al. [5] proposed the Deep Gradient Compression method, in which the Gradient was sparse to 0.1% of the original gradient and the Compression rate could reach a maximum of 600 times. However, due to the high proportion of sparsity, four strategies such as momentum correction, local gradient pruning, momentum factor masking, and warm-up training have to be adopted to compensate for the loss of accuracy.

Although this method achieves a high compression ratio, the correction cost is also very high, but the training task quantity is increased.

In this paper, we propose an adaptive compression method based on gradient partition to overcome the limitations of the above methods. We analyzed the distribution characteristics of the gradient in each layer of the neural network and concluded that the distribution of the gradient was regular and approximately normal. Similar conclusions were also found in references [6,7]. According to the gradient distribution characteristics, the gradient region is divided, to screen out more favorable gradient values for training, to realize the reasonable distribution of updated gradient values in different gradient division intervals. Based on gradient partition, we further introduce a method of adaptive threshold selection for gradient update. Different from the existing method of setting threshold by experience, this method combines the information entropy of gradient distribution to select a reasonable threshold to realize threshold filtering. It is also proved that the gradient descent method can achieve normal convergence based on gradient partition. It can be said that on the premise of ensuring the accuracy and convergence of training, our method improves the efficiency of training to a certain extent.

2. Related Works

At present, researchers have proposed many methods to reduce communication costs and speed up distributed training of neural networks. Among them, gradient compression is one of such methods. Compared with other solutions, gradient compression strategy only plays a role in the communication content between computing nodes but does not affect the communication process, let alone change the structure of the training model. Therefore, to a certain extent, gradient compression achieves the purpose of reducing communication costs. Solutions based on gradient compression strategy can be roughly divided into two categories: gradient quantization and gradient sparsity.

2.1. Gradient Quantization

Gradient quantization refers to the quantization of gradient and the reduction of its precision. The principle of quantization is to reduce the transmission of data by reducing the bit of the gradient value of floating-point type transmitted, to improve the communication efficiency between nodes. At present, the main challenge of gradient quantization is how to obtain satisfactory model accuracy with less gradient information. Researchers have proposed several methods for quantifying gradients.

Wen et al. [8] proposed the Ternary Gradients(TernGrad) algorithm, which quantified the gradient value into three values $\{-1, 0, 1\}$ to reduce the cost of gradient communication. It is also proved that the convergence of the algorithm can be guaranteed as long as the quantized gradient mean is equal to the original gradient. Through TernGrad algorithm, the gradient of the traffic can be reduced to 1/20 of the original.

Khairat et al. [9] studied quantization under deterministic setting conditions, that is, gradient values can be calculated noiselessly, and proved that parallel gradient descent with unbiased quantization will converge to the optimal solution of a strongly convex function. Seide et al. [10] proposed a 1-bit quantization method to quantify each gradient parameter as a symbol bit, that is, in the communication between computing nodes, each parameter is quantized as a bit, which can significantly reduce the size of communication data. At the same time, to avoid the decline of the accuracy of the model, the error in the quantization process is retained and added to the calculation gradient in the next step. This method achieves a 10-fold acceleration ratio in the classical speech recognition application.

In order to solve the problem of low model accuracy or low compression rate in the existing quantization schemes, Guo et al. [11] proposed a novel adaptive quantization scheme(AdaQS). AdaQS algorithm according to the ratio of the mean and standard deviation of the gradient (MSDR) automatically determine the quantitative level, and achieve a balance between model accuracy and quantitative level.

Mishchenko et al. [12] considered the stochastic setting in data parallelization and showed that Stochastic Gradient Descent (SGD) with unbiased quantization converges to near the stationary points of non-convex and strongly convex functions. However, the Quantized SGD (QSGD) method proposed by Alistarh et al. [13] is to achieve a balance between training accuracy and gradient quantization by rounding the gradient into a linear quantization value randomly, to maintain a certain training accuracy without quantizing the result of excessive damage to the training. At the same time, the validity and convergence of the method can be guaranteed in the process of gradient quantization.

2.2. Gradient Sparsification

Gradient sparsity reduces communication costs by setting a threshold and sending a small number of parameters. Aji et al. [14] proposed a distributed gradient descent method, in which the gradient is sparse by removing the gradient with the minimum absolute value of $R\%$ ($R\%$ is a fixed percentage). At the same time, to ensure convergence, a layer with a global threshold is used for normalization. This method can map 99% of the minimum gradient value to zero and improve the speed of translation by 11% when applied to neural machine translation without affecting the translation quality.

Shi et al. [15] discussed that under the influence of communication bandwidth in distributed training, both the gradient sparse method and the gradient merge method would be damaged to a certain extent, and therefore could not achieve a better training effect. They proposed an SGD-based gradient sparse optimization algorithm (OMGS-SGD). The algorithm considers the tradeoff between communication and computation to find the optimal solution. The experimental results show that the efficiency of the proposed algorithm is improved by 31% when the convergence is almost the same as that of the original SGD.

The Adaptive Residual Gradient Compression (AdaComp) method proposed by Chen et al. [16] divides the gradient residuals of each layer into several groups of fixed size, and use the gradient with the largest absolute value in each group as the threshold to screen the gradient, thus using local gradient activities to automatically adjust the compression rate. If the previous gradient residual combined with the gradient value calculated in proportion exceeds this maximum value, the gradient is considered important and the gradient is updated, otherwise, it is not updated.

Strom et al. [17] developed a threshold sparse method, in which only the gradient greater than the set threshold is sent, thus only a small part of the value of the transmission gradient is needed to be updated. Similar to 1-bit quantization, the residual gradient after sparsity is accumulated into the next gradient update. At the same time, how to determine the threshold value of the gradient is also a thorny problem. Dryden et al. [18] proposed a fixed proportion of positive and negative gradients to send gradients instead of fixed thresholds.

In this section, two methods to reduce communication costs were introduced, gradient quantization and gradient sparsity. Gradient quantization is realized by reducing the bit of gradient transmission, but the compression rate is limited. Gradient sparsity improves the compression rate by setting a threshold to screen the gradient, but it is a difficult problem how to select a reasonable threshold. In this paper, we propose an adaptive compression method based on gradient partition, which divides the gradient according to its distribution characteristics and selects an appropriate threshold value for gradient update in combination with the information entropy of gradient distribution. At the same time, our method will not affect the gradient communication process, let alone change the training model structure.

3. Gradient Distribution

To solve the problems mentioned in the previous section, we started from the distribution characteristics of the gradient, and after investigating relevant literature, we reached the conclusion that the gradient distribution in the neural network is approximately normal

distribution. This can be verified in references [6,7]. In references [6], Wang et al. set up experiments and trained on AlexNet network by using ImageNet, CIFAR10, and other data sets, respectively, concluded conclusion that the gradient was an approximately normal distribution. Similarly, Chmiel et al. In references [7] also achieved similar results by using the dataset ImageNet on ResNet18 network. These papers all proved the conclusion that the gradient distribution in neural network training is an approximately normal distribution.

Based on the above references, the characteristics of gradient distribution are re-counted and analyzed in this paper. In this paper, the deep neural network VGG16 and CIFAR10 datasets were used for training. Besides, the log files generated by training are visualized through the TensorBoard tool provided in TensorFlow, a deep learning framework, and the gradient distribution results are obtained (see the results in Figure 2).

Figure 2. Gradient distribution histogram of some layers sampled in VGG16.

In Figure 2, the gradient distribution of some layers of VGG16 is shown. Due to too many layers, only the gradient distribution characteristic map of some layers of VGG16 is selected. Where the upper left corner of each graph indicates the gradient distribution of which layer in VGG16. Meanwhile, the horizontal axis represents the values of the gradients and the vertical axis represents the statistical quantity of the gradient value in this layer.

To sum up, both the conclusions of relevant literature and the conclusions of experiments in this paper can fully explain that the gradient distribution in the neural network is approximately normal distribution. According to the property of normal distribution, it can be known that about 68.26% of the samples fall between $(\mu - \sigma, \mu + \sigma)$. This means that most of the gradient values are distributed around 0 (when $\mu = 0$), and this part of gradient parameters has little influence on the training. Therefore, the problem of redundancy is generated. Therefore, gradient sparsity is necessary.

4. Compression Based on Gradient Partition

According to the gradient distribution characteristics obtained in the previous section, the gradient distribution is approximately described as $x \sim N(\mu, \sigma^2)$. According to the gradient of the distribution of the following division (see the division in Figure 3).

Figure 3. Gradient partition.

According to the characteristics of gradient update, choosing a gradient value with a larger absolute value is more beneficial to update and has a greater influence on the weight. Therefore, we take the $(\mu - 3\sigma, \mu - \sigma) \cup (\mu + \sigma, \mu + 3\sigma)$ interval of the gradient distribution as the focus area of the gradient update and the interval of $(\mu - \sigma, \mu + \sigma)$ is the sparse area of the gradient update. The method that can be considered is to take more gradient values in the key interval of the gradient update to participate in the update, while in the sparse interval of the gradient update, tries to take fewer values (or even no values in special cases) to participate in the update.

Based on gradient division, this paper uses information entropy to measure the importance of gradient information in each interval, so as to serve as the basis for gradient selection.

4.1. Information Entropy of Normal Distribution

Information entropy is often used to measure the uncertainty of variable distribution. The following is the information entropy calculation formula under the probability distribution of discrete variables [19]:

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i) \quad (1)$$

The probability density function of normal distribution is:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (2)$$

Then the Shannon information of normal distribution is (the base of logarithm is e):

$$-\ln p(x) = \frac{1}{2} (\ln(2\pi) + 2\ln(\sigma) + \left(\frac{x-\mu}{\sigma}\right)^2) \quad (3)$$

The information entropy of normal distribution, that is, the expectation of Shannon information. Therefore, the information entropy of normal distribution is:

$$E = \frac{1}{2} (\ln(2\pi) + 2\ln\sigma + 1) \quad (4)$$

Considering that the gradient values in the neural network are all decimals, to ensure that the information entropy is positive, the expression is expressed as follows:

$$H = -\frac{1}{2}(\ln(2\pi) + 2\ln\sigma + 1) \quad (5)$$

It can be seen from the above formula that the information entropy of normal distribution is only related to its standard deviation.

4.2. Threshold Setting

In this section, we propose a gradient threshold screening algorithm based on information entropy to set the threshold screening gradient. The specific process of the algorithm is shown in Algorithm 1.

Algorithm 1 Sparse gradient based on information entropy algorithm.

Require: Gradients $G_l, l = 1 \dots L$ for l -th layer in the net;

Require: The number of gradient elements in the layer N ;

Ensure: $T_l, l = 1 \dots L$;

1: **for** layer $l = 1 \dots L$ **do**

2: Calculate the average of $G_l: \bar{g}_l = \frac{\sum_{i=1}^N g_l^i}{N}$;

3: Calculate the standard deviation of $G_l: \sigma_l = \sqrt{\frac{1}{N} \sum_{i=1}^N (g_l^i - \bar{g}_l)^2}$;

4: Calculate the information entropy of the normal distribution: $H = -\frac{1}{2}(\ln(2\pi) + 2\ln\sigma_l + 1)$;

5: Function: $H(\alpha) = -\frac{1}{2}(\ln(2\pi) + 2\ln\sigma_\alpha + 1)$;

6: $\gamma = \text{abs}[(G_{max}^l - G_{min}^l) / G_{sum}^l]$;

7: $S = H(\alpha) * \gamma$;

8: Threshold: $T_l = \text{Top} - k(G_l, S)$;

9: **end for**

The algorithm has the following important steps. Firstly, what we need to do is how to select the appropriate gradient threshold according to the information entropy H of gradient distribution. The specific description is as follows, let the gradient threshold be α , and the constructor:

$$H(\alpha) = -\frac{1}{2}(\ln(2\pi) + 2\ln\sigma_\alpha + 1) \quad (6)$$

where σ_α is the standard deviation of the gradient in the layer where the threshold is located, and $H(\alpha)$ is the information entropy of the gradient distribution in the layer.

$$\gamma = \text{abs}[(G_{max}^l - G_{min}^l) / G_{sum}^l] \quad (7)$$

Secondly, where γ is the trade-off factor, G_{max}^l is the maximum gradient value of each layer, G_{min}^l is the minimum gradient value in each layer, and G_{sum}^l is the sum of absolute gradient values of each layer.

$$S = H(\alpha) * \gamma \quad (8)$$

Thirdly, where S is the set gradient sparsity. According to the gradient sparsity, we use the *Top-k* sorting algorithm to find the first S large gradient values in this layer, and get the threshold α . G_l is the gradient value of l -th layer.

$$\text{Threshold} = \text{Top} - k(G_l, S) \quad (9)$$

Finally, the function $T_\alpha(g_l)$ represents the threshold selection process, where g_l represents the gradient elements in l -th layer and α is the set threshold value. Then the gradient screening process can be represented by the following piecewise functions:

$$T_\alpha(g_l) = \begin{cases} g_l & |g_l| > \alpha \\ \alpha & |g_l| = \alpha \\ 0 & |g_l| < \alpha \end{cases} \quad (10)$$

It is through this way to filter and update the gradients in training.

4.3. Convergence Analysis of Gradient Partition

In this section, because of the division of the gradient distribution interval, the convergence analysis is used to prove whether the gradient descent in the partition interval can achieve convergence without affecting the training. We get the following conclusion:

$$L(w_k) \leq L(w_*) + \frac{\eta}{2} \alpha^2 \quad (11)$$

Preparation condition. Given the training data set of size n , any one random sample of the training data set is represented as $\varphi(x, y)$, the model parameter w is defined, and the loss function is represented as $l(w, \varphi)$, then the objective function is minimized as:

$$L(w) = \frac{1}{n} \sum_{i=1}^n l(w, \varphi_i) \quad (12)$$

If φ_k is the sample input for k iterations, then the random gradient value is expressed as follows, the random gradient value can be abbreviated as $g(w_k)$.

$$g(w, \varphi_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} \nabla l(w, \varphi_{k_i}) \quad (13)$$

Theorem 1. Gradient function $\nabla L(w)$ is on the interval $(\mu - 3\sigma, -|\alpha|) \cup (|\alpha|, \mu + 3\sigma)$, and the Lipschitz condition is satisfied for the nonconvex object $L(w)$ and the Lipschitz constant $\lambda > 0$. Lipschitz continuity condition, is a more smooth condition than usual, which is often used to analyze the convergence of gradient descent algorithm. Let the step size be η , the initial value w_0 , and the number of iterations $k \in N$ (N is the set of natural numbers), $w_{k+1} = w_k + \eta \nabla L(w)$. If $L(w)$ converges, then the following equation is satisfied:

$$L(w_k) \leq L(w_*) + \frac{\|w_0 - w_*\|^2}{2\eta k} \quad (14)$$

Let $w_* = \operatorname{argmax} L(w)$ be the optimal solution, if the Lipschitz condition is satisfied for the nonconvex target $L(w)$, then for the step η_t :

$$\eta_t \leq \frac{1}{\lambda} \quad (15)$$

The λ is the Lipschitz constant, then (14) can be written as follows:

$$L(w_k) \leq L(w_*) + \frac{\lambda \|w_0 - w_*\|^2}{2k} \quad (16)$$

Proof. Suppose that the gradient function $\nabla L(w): R \rightarrow R^m$ is Lipschitz continuous and the Lipschitz constant $\lambda > 0$, for any $w_k, w_{k+1} \in R^m$:

$$L(w_{k+1}) \leq L(w_k) + \nabla L(w_k)(w_{k+1} - w_k) + \frac{\lambda}{2} \|w_{k+1} - w_k\|^2 \quad (17)$$

According to the gradient descent method $w_{k+1} = w_k - \eta \nabla L(w_k)$, the right side of Formula (17) can be written as:

$$\begin{aligned}
 &= L(w_k) + \nabla L(w_k)(-1)(\eta \nabla L(w_k)) + \frac{\lambda}{2} \eta^2 \nabla L^2(w_k) \\
 &= L(w_k) - \eta \nabla L^2(w_k) + \frac{\lambda}{2} \eta^2 \nabla L^2(w_k) \\
 &= L(w_k) - \eta(1 - \frac{\lambda}{2}) \nabla L^2(w_k) \\
 &\leq L(w_k) - \frac{\eta}{2} \nabla L^2(w_k)(\eta \leq \frac{1}{\lambda}) \\
 &\leq L(w_*) + \nabla L(w_k)(w_k - w_*) - \frac{\eta}{2} \|\nabla L(w_k)\|^2 \\
 &(\text{used } L(w_k) - L(w_*) \leq \nabla L(w_k)(w_k - w_*)) \\
 &= L(w_*) + \frac{w_k - w_{k+1}}{\eta}(w_k - w_*) - \frac{\eta}{2} \|\frac{w_k - w_{k+1}}{\eta}\|^2 \\
 &(\text{used } w_{k+1} = w_k - \eta \nabla L(w_k))
 \end{aligned}$$

By expanding and merging the above formula, we can get the following results:

$$\begin{aligned}
 &= L(w_*) + \frac{1}{2\eta}(w_k - w_*)^2 - \frac{1}{2\eta}(\|w_k - w_*\|^2 - 2\eta \nabla L(w_k)(w_k - w_*) + \|\eta \nabla L(w_k)\|^2) \\
 &= L(w_*) + \frac{1}{2\eta}(w_k - w_*)^2 - \frac{1}{2\eta}(\|w_k - w_*\| - \|\eta \nabla L(w_k)\|)^2
 \end{aligned} \tag{18}$$

According to $w_k - w_{k+1} = \eta \nabla L(w_k)$, $\nabla L(w_k), \nabla L(w_{k+1}) \in (\mu - 3\sigma, -|\alpha|) \cup (|\alpha|, \mu + 3\sigma)$, therefore $w_k - w_{k+1} \in \eta(\mu - 3\sigma, -|\alpha|) \cup \eta(|\alpha|, \mu + 3\sigma)$.

$$= L(w_*) + \frac{1}{2\eta}(\eta|\alpha|)^2 - \frac{1}{2\eta}(\|\eta|\alpha|\| - \|\eta|\alpha|\|)^2 \tag{19}$$

$$= L(w_*) + \frac{\eta}{2}\alpha^2 \tag{20}$$

In summary:

$$L(w_{k+1}) \leq L(w_*) + \frac{\eta}{2}\alpha^2 \tag{21}$$

$$L(w_{k+1}) - L(w_*) \leq \frac{\eta}{2}\alpha^2$$

Each iteration is expanded by formula:

$$L(w_1) - L(w_*) \leq \frac{\eta}{2}\alpha^2$$

$$L(w_2) - L(w_*) \leq \frac{\eta}{2}\alpha^2$$

... ..

$$L(w_k) - L(w_*) \leq \frac{\eta}{2}\alpha^2$$

Add up the above formula:

$$\begin{aligned}
 \sum_{k=1}^k L(w_k) - kL(w_*) &\leq \frac{k\eta}{2}\alpha^2 \\
 L(w_k) - L(w_*) &\leq \frac{\eta}{2}\alpha^2 \\
 L(w_k) &\leq L(w_*) + \frac{\eta}{2}\alpha^2
 \end{aligned} \tag{22}$$

So far, we get a conclusion (22). At the same time, the convergence condition is as follows:

$$L(w_k) \leq L(w_*) + \frac{\|w_0 - w_*\|^2}{2\eta_k k}$$

If $w_k - w_{k+1} \in \eta(\mu - 3\sigma, -|\alpha|) \cup \eta(|\alpha|, \mu + 3\sigma)$. They are as follows:

$$L(w_k) \leq L(w_*) + \frac{\|\eta|\alpha|\|^2}{2\eta_k k}$$

$$L(w_k) \leq L(w_*) + \frac{\eta}{2k} \alpha^2 \quad (23)$$

□

$\frac{\eta}{2} \alpha^2$ in Formula (22) $> \frac{\eta}{2k} \alpha^2$ in Formula (23), $\frac{\eta}{2} \alpha^2, \frac{\eta}{2k} \alpha^2$ are both positive numbers, the following formula must be established.

$$L(w_k) \leq L(w_*) + \frac{\|w_0 - w_*\|^2}{2\eta_k k}$$

Therefore, the gradient function must converge on the interval $(\mu - 3\sigma, -|\alpha|) \cup (|\alpha|, \mu + 3\sigma)$.

5. Experiment

In order to verify the effectiveness of the adaptive compression algorithm based on gradient partition proposed in this paper, an experimental simulation environment was constructed to simulate the distributed training scene of neural networks for experimental verification. Through simulation experiments we verified the performance of our gradient compression algorithm, from the convergence of the algorithm, compression ratio, and training throughput to analyze and evaluate.

The datasets used in this experiment were MNIST and Cifar10, and the parameter information of the neural network models used in training is shown in Table 1.

Table 1. Parameter information in training.

Dataset	Network Model	Batch Size	Iterations	Learning Rate
MNIST	LeNet	128	1000	0.01
	AlexNet	128	3000	0.01
	ResNet	128	300	0.01
Cifar10	AlexNet	128	6000	0.001
	ResNet	128	10,000	0.001

The experiment in this paper was completed by TensorFlow framework [20], which was implemented in the form of single machine multi-card. The Parameter Server (PS) node and Worker nodes in distributed training were simulated by three port addresses of the local machine. One port address was used to simulate the PS node, which was mainly responsible for summarizing and calculating the global gradient and updating the model parameters. The other two port addresses were used to simulate the Worker node to calculate the gradient and receive the latest model parameters from the PS node. The bandwidth of links in the network was set to 10 Mbit/s. At the same time, the data set used in this paper is MNIST, and the neural network models used included LeNet, AlexNet and ResNet.

5.1. Convergence Analysis

The convergence of the algorithm refers to the algorithm after many iterations, the numerical value tends to a certain value, the convergence of the algorithm reflects the ability

of the algorithm to find the optimal solution. The accuracy rate of the three algorithms under the AlexNet network model is shown in Figure 4, where Baseline represents the reference group, Gradient Variance represents the compression algorithm based on gradient variance [21], and Partition Gradient Compression (PGC) represents the adaptive compression algorithm based on gradient partition proposed in this paper.

Figure 4. (a) Accuracy in AlexNet network on MNIST; (b) Loss curve in AlexNet network on MNIST.

It can be seen from Figure 4 that the algorithm proposed in this paper had a similar accuracy to the Baseline, Gradient Variance, and other methods. With the increase of the number of training steps, the overall fluctuation of the curve was more stable, which indicates that the threshold sparsity played a certain role. Although there was some fluctuation in the early iteration, it was since in the initial stage of training, the threshold value of gradient was directly sparse, which would cause certain accuracy damage. In the loss curve, the loss was smaller than the other two methods.

The three algorithms were also trained in the ResNet network model. The accuracy and loss curves in ResNet network model are shown in Figure 5.

Figure 5. (a) Accuracy in ResNet network on MNIST; (b) Loss curve in ResNet network on MNIST.

ResNet network adopted the ResNet-50 and MNIST datasets. Under the same experimental conditions, the accuracy of model recognition could reach more than 95% after only 300 steps of training. As can be seen from Figure 5, compared with the other two methods, the overall fluctuation of the adaptive compression algorithm based on gradient partition was more stable, and the accuracy rate gradually tended to a stable range, indicating that the algorithm could achieve normal convergence, and the loss curve also showed that the algorithm could achieve normal convergence.

Similarly, training on the LeNet network could obtain similar results, as shown in Figure 6.

Figure 6. (a) Accuracy in LeNet network on MNIST; (b) Loss curve in LeNet network on MNIST.

According to the curve fluctuation in Figure 6, it can be seen that the adaptive compression algorithm based on gradient partition could also achieve normal convergence. Besides, we also verified the convergence of AlexNet and ResNet on the dataset Cifar10, as shown in Figures 7 and 8.

Figure 7. (a) Accuracy in AlexNet network on Cifar10; (b) Loss curve in AlexNet network on Cifar10.

Figure 8. (a) Accuracy in ResNet network on Cifar10; (b) Loss curve in ResNet network on Cifar10.

It can be seen from Figures 7 and 8 that the PGC method in this paper also achieved better accuracy and convergence effect on different datasets. To sum up, by analyzing the convergence of the algorithm on the three network models, the algorithm proposed in this paper could achieve good convergence under the same experimental conditions. At the same time, compared with the other two methods, it also achieved better contrast. In this section, the training accuracy of all network models are shown in Table 2.

Table 2. The accuracy of all network models.

Datasets	Network Model	Baseline	Gradient Variance	PGC
MNIST	LeNet	96.65%	96.59%	96.86%
	AlexNet	90.02%	89.95%	91.17%
	ResNet	95.90%	96.45%	96.75%
Cifar10	AlexNet	71.78%	71.89%	72.21%
	ResNet	89.97%	90.25%	90.43%

5.2. Compression Ratio Analysis

In this paper, the data size of the total gradient parameters in the training of three kinds of network models was counted, and the gradient data size after compression by related algorithms was counted as the measurement of algorithm compression ratio. The details are shown in Table 3.

Table 3. Compression ratio information.

Dataset	Network Model	Gradient Size	Baseline	Gradient Variance	PGC
MNIST	LeNet	0.45 M	1×	18.3×	24.2×
	AlexNet	1.39 M	1×	23.7×	29.1×
	ResNet	4.27 M	1×	30.4×	37.1×
Cifar10	AlexNet	4.56 M	1×	20.1×	27.3×
	ResNet	6.50 M	1×	25.6×	32.9×

It can be seen from the above table that the compression ratio of the algorithm increased with the increase of network layers by testing on three different network models. In the AlexNet model, the algorithm proposed in this paper could achieve a compression ratio of about 29.1 times, while for the network model ResNet with more layers, it could achieve a compression ratio of about 37.1 times. When training AlexNet and ResNet on Cifar10, the compression ratio could reach 25.6 times and 32.9 times. Compared with the other two methods, the compression ratio was also improved.

5.3. Training Throughput Analysis

According to the analysis of training throughput of adaptive compression algorithm based on gradient partition, the training throughput refers to the number of batch data read in unit time. The larger the batch value, the faster the reading speed and the higher the training efficiency. This experiment was carried out under the condition of two computing nodes and the bandwidth was 10 Mbit/s. The training throughput of the proposed algorithm on three different network models was tested. The training throughput results are shown in Figure 9.

Figure 9. Training throughput on three different network models.

Through the analysis of the above histogram, it can be seen that although the efficiency of the algorithm was limited in the LeNet network with few network layers, it may also have been affected by the bandwidth of communication and the size of computing tasks. However, compared with the other two methods, the training throughput was improved to a certain extent. In addition, the throughput of our method on the AlexNet network was increased by 1.18 times compared with the Gradient Variance method and 1.4 times higher than that of the Baseline method. When training on the network model ResNet with more layers, the efficiency of the algorithm was improved by 1.3 times compared with Gradient Variance method and 1.5 times higher than Baseline method.

6. Conclusions

In this paper, we analyze and study the distribution characteristics of gradient parameters in neural network training for AIoT systems, and propose an adaptive compression strategy based on gradient partition. Firstly, according to the distribution characteristics of the gradient, the gradient value is divided into the key area of gradient update and the sparse area. At the same time, the information entropy formula of gradient distribution feature is used to select the gradient value. To reduce the communication cost and improve the training efficiency, the gradient is sparse to a certain extent by selecting the updated gradient value in the partition by the threshold.

Through the analysis of relevant experimental results, our method can achieve normal convergence after the training of different network models, and compared with the method without compression scheme, this algorithm can improve the compression ratio to a certain extent, and under the premise of ensuring the accuracy and convergence, the maximum compression ratio can achieve about 37.1 times. Besides, the algorithm also improves the training throughput and speeds up the training speed to a certain extent. At the same time, the method proposed in this paper also provides some more useful ideas for data transmission between nodes in Mobile Edge Computing (MEC) and communication optimization in Federated Learning (FL). We also hope that this idea can be applied to more fields and provide more meaningful progress.

Author Contributions: Conceptualization, Y.L. and B.G.; investigation, Y.L. and B.G.; methodology, Y.L. and B.G.; validation, Y.L., B.G. and C.Z.; writing—original draft preparation, B.G. and C.Z.; supervision, Y.L.; writing—review and editing, Y.L., B.G. and C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This publication has emanated from a joint research conducted with the financial support of the S&T Major Project of the Science and Technology Ministry of China under the Grant No. 2017YFE0135700.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tang, X.; Wang, X.; Cattley, R.; Xianghong, W.; Xiaoli, T. Energy harvesting technologies for achieving self-powered wireless sensor networks in machine condition monitoring: A review. *Sensors* **2018**, *18*, 4113. [[CrossRef](#)] [[PubMed](#)]
2. Wang, Z.L.; Wu, W. Nanotechnology-enabled energy harvesting for self-powered micro-/nanosystems. *Angew. Chem. Int. Ed.* **2012**, *51*, 11700–11721. [[CrossRef](#)] [[PubMed](#)]
3. Zhang, H.; Zheng, Z.; Xu, S.; Wei, D.; Qirong, H.; Xiaodan, L.; Zhiting, H.; Jinliang, W.; Pengtaol, X.; Xing, E.P. Poseidon: An efficient communication architecture for distributed deep learning on GPU clusters. In Proceedings of the 2017 USENIX Annual Technical Conference (USENIXATC 17), Santa Clara, CA, USA, 12–14 July 2017; pp. 181–193.
4. Dutta, A.; Bergou, E.H.; Abdelmoniem, A.M.; Yu, H.C.; Narayan, S.A.; Marco, C.; Panos, K. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3817–3824.

5. Yujun, L.; Yu, W.; Song, H.; William, J.D.; Huizi, M. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv* **2017**, arXiv:1712.01887.
6. Wang, L.; Wu, W.; Zhao, Y.; Zhang, J.; Liu, H.; Bosilca, G.; Fonseca, R. SuperNeurons: FFT-based Gradient Sparsification in the Distributed Training of Deep Neural Networks. *arXiv* **2018**, arXiv:1811.08596.
7. Chmiel, B.; Ben-Uri, L.; Shkolnik, M.; Elad, H.; Ron, B.; Daniel, S. Neural gradients are lognormally distributed: Understanding sparse and quantized training. *arXiv* **2020**, arXiv:2006.08173.
8. Wen, W.; Xu, C.; Yan, F.; Wu, C.; Wang, Y.; Chen, Y.; Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *arXiv* **2017**, arXiv:1705.07878.
9. Khirirat, S.; Feyzmahdavian, H.R.; Johansson, M. Distributed learning with compressed gradients. *arXiv* **2018**, arXiv:1806.06573.
10. Seide, F.; Fu, H.; Droppo, J.; Gang, L.; Dong, Y. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014.
11. Guo, J.; Liu, W.; Wang, W.; Jizhong, H.; Ruixuan, L.; Yijun, L.; Songlin, H. Accelerating Distributed Deep Learning By Adaptive Gradient Quantization. In Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; IEEE: New York, NY, USA, 2020; pp. 1603–1607.
12. Mishchenko, K.; Gorbunov, E.; Takáč, M. Distributed learning with compressed gradient differences. *arXiv* **2019**, arXiv:1901.09269.
13. Alistarh, D.; Grubic, D.; Li, J.; Ryota, T.; Milan, V. QSGD: Communication-efficient SGD via gradient quantization and encoding. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1709–1720.
14. Aji, A.F.; Heafield, K. Sparse communication for distributed gradient descent. *arXiv* **2017**, arXiv:1704.05021.
15. Shi, S.; Wang, Q.; Chu, X.; Bo, L.; Yang, Q.; Ruihao, L.; Xinxiao, Z. Communication-efficient distributed deep learning with merged gradient sparsification on gpus. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020.
16. Chen, C.Y.; Choi, J.; Daniel, B.; Ankur, A.; Wei, Z.; Kailash, G. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. *arXiv* **2017**, arXiv:1712.02679.
17. Strom, N. Scalable distributed DNN training using commodity GPU cloud computing. In Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association, Dresden, Germany, 6–10 September 2015.
18. Dryden, N.; Moon, T.; Jacobs, S.A.; Van Essen, B. Communication quantization for data-parallel training of deep neural networks. In Proceedings of the 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), Salt Lake City, UT, USA, 14 November 2016; IEEE: New York, NY, USA, 2016; pp. 1–8.
19. Cover, T.M.; Thomas, J.A. Elements of information theory. In *Wiley Series in Telecommunications and Signal Processing Hoboken*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
20. Abadi, M.; Isard, M.; Murray, D.G. A computational model for TensorFlow: An introduction. In Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, Barcelona, Spain, 18 June 2017; pp. 1–7.
21. Tsuzuku, Y.; Imachi, H.; Akiba, T. Variance-based gradient compression for efficient distributed deep learning. *arXiv* **2018**, arXiv:1802.06058.