Image classification model based on spark and CNN

Jiangfeng Xu, and Shenyue Ma*

School of Information Engineering, Zhengzhou University, Zhengzhou, China

Abstract. Convolution neural network is a commonly used image classification model, but when the network nodes of the training process are too many, it will have a great influence on the training complexity. At the same time, when the size of the image data is large, many problems will appear on the single node, such as convergence slowly, frequently disk reading and writing. In order to overcome the above problems, this paper proposes a distributed convolution neural network based on Spark (Distribution Convolution neural network, Dis-CNN) model. The model first improves the initialization mode of convolution kernel parameters, then eliminates the redundancy of feature maps, and finally optimizes the distributed gradient descent by reducing the synchronous traffic between master and slave, so as to improve the convergence speed and performance. The experimental results show that the model not only improves the accuracy and recall of image classification, but also performs excellent in parallelism.

1 Introduction

With the rapid increase of image data, how to classify large scale image data is a hot topic. Currently, the convolution neural network (CNN) is widely used in the field of image recognition. In 2012, Hinton and its students used the AlexNet [1] to win the champion in the ImageNet competition. Then, in 2013 [2] and 2014 [3], the champion of the event also used the deep CNN. For image classification, the classic CNN model has achieved better recognition effect, but due to its hidden layer, there are too many network nodes, which increase the complexity of training. In order to reduce network complexity and improve recognition, the network structures were improved in the literatures [4, 5], the network training parameters were optimized in the literature [6] and incentive function was improved in the literature [7]. For small-scale image data, using the above model can quickly complete the training. But when the scale of the image is expanded, training on a single machine leads to frequent IO and can't solve the problem of time consuming in the CNN training process. Therefore, some people have turned research into distributed platforms, such as Stefan GM [8]. Literature [9] did the distributed training by GPU, and the literature [10] used distributed asynchronous stochastic gradient descent to train the depth network, both of them have the effect of training acceleration.

* Corresponding author: shenyueliu@163.com

Although there have been many researches on the distributed implementation of deep network, the problems of model parameters updating and communication delay are still needed to be analyzed. After the research and analysis of several distributed technologies, the Spark platform was selected for CNN processing in this paper. This is because Spark is based on in-memory operations, which is good for iteration, and can be highly processed with massive amounts of data. In both iterative and interactive aspects, it exceeds the Hadoop. In order to improve the convergence speed and classification accuracy of CNN's large-scale image training, this paper mainly works from the following two aspects:

(1) Change the initialization mode of the convolution kernel parameter and reduce the redundancy of the feature graph in the forward propagation process;

(2) Optimize the master-slave communication conflict in the distributed gradient descent, and assigning different weights to the gradient data to improve the training speed and classification accuracy of the model.

2 Relevant knowledge

Spark is a popular big data calculation engine, whose core is Resilient Distributed Datasets (RDD) taken as the basic data structure to perform distributed computing on multiple machines, RDD has two kinds of operations: conversion operator and action operator. Spark is just a distributed computing framework, and its computing data is usually stored on a Hadoop Distributed File System (HDFS). The HDFS consists of the Namenode and the Datanode, where the Namenode is responsible for scheduling the Datanode node. And Spark has the advantages of DAG execution engine and memory based multi-wheel iteration calculation, which is more helpful to the iterative calculation of convolution neural network than MapReduce.

The basic structure of CNN is shown in figure 1. In terms of structure, it has input layer, convolution layer, pooling layer, full connection layer and output layer. A CNN extracts low-level features through the coiling layer, and then combines them into more abstract features through the sampling layer to form the feature description of the image. Finally, the image is identified and classified by activation function. The training of CNN model usually requires a large sample data set, and it is very sensitive to the initialization of convolution kernel parameters and the number of network nodes hidden. Literature [4] used dropout to discard hidden layer network nodes according to probability, Literature [11] used k-means to initialize the convolution kernel parameters. A few changes in these two factors usually have a significant impact on network performance.

Fig. 1. Basic structure of the convolution neural network.

In the research of stochastic gradient descent for distributed cluster system, Stefan GM [8] and others used MapReduce parallel model to realize the parallelism of stochastic gradient descent algorithm. This algorithm has achieved an accelerated effect to some extent, but there are still a lot of shortcomings. But MapReduce is based on a large number of IO to achieve data transmission in Map and Reduce stages, which will undoubtedly waste a lot of time for iteration. The way based on Hadoop has computational and theoretical imperfections, For example, to complete the transmission of parameters by the communication between MapReduce main program and Map task, Communication between the Reduce task and the MapReduce master program to complete the update of the model parameters. The convergence rate of the model is greatly depressed. Therefore, the

problems caused by frequent communication and synchronous operation are usually the main contents in the process of designing distributed gradient descent algorithm.

3 Dis-CNN model

In order to speed up the CNN training speed and prediction accuracy of image classification, this paper puts forward an improved neural network, namely distributed Convolution neural network (Dis-CNN) model, In the model forward propagation and back propagation was improved. In the forward propagation process, firstly, the initialization of the convolution kernel parameters is improved. Then redundant feature maps will be discarded in each convolution layer, so that we can reduce network nodes to improve training speed and improve accuracy. In the process of back propagation, the computation of distributed gradient descent is speeded up by reducing the traffic between master and slave node. To group the slaves and assign different weights for the error calculation result, and different groups provide different contributions, the experience shows that the algorithm recognition accuracy can be improved.

3.1 Combining random with k-means to initialize convolution kernel parameters and reducing feature maps

Convolution kernel parameter initialization is very important for the training of the network, common model is zero initialization, random initialization etc, but those will break the symmetry among the neurons, or have little impact on the training results. For the above problems, the literature [11] used k-means to train the kernel parameters of the first layer of the network. While using it on the model, recognition degree got higher, but k-means are more sensitive to noise and isolated points, a small amount of this kind of data will be a great influence on the result, lead to bad clustering results. Therefore, this paper integrates k-means and random way to initialize the convolution kernel parameters, and the initial cluster is determined using the arithmetic sequence. And unlike the literature [11], this method was used for all the convolution layers.

DropConnect [12] can increase the generalization ability of the network, which is the improvement of the Dropout, and DropConnect is aimed at the weight function, and the Dropout targets the output of the neurons. However, after the network uses DropConnect, there is still a probability event occurrence feature maps redundancy, which not only increases the number of training parameters but also reduces the generalization ability and accuracy of the network. Therefore, the number of network nodes can be optimized by calculating the difference among the feature maps. In convolution layer, to calculate the similarity parameter H between maps. The more similar the feature maps are, the smaller the value of the H is, we can set a threshold a , if H is less than a , then eliminate the feature maps compared. The number of network nodes decreases accordingly. The calculation formula of convolution layer is as follows:

$$x_j^l = f \left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l \right) \quad (1)$$

x_j^l is the j th feature graph of the l layer, and k_{ij}^l represents the j th convolution kernel of the l layer, b_j^l indicating the j th bias of the l layer, and $f(*)$ represents the active function.

Step1 Determine the unit size of the image block: the model integrates the random and the k-means method to construct the convolution kernel parameter. In the first convolution layer of the model, there are 8 kernels of $5*5$. The strategy is that the 14 parameters on the periphery are filled with random numbers, and the 9 in the middle are obtained using k-means. So the image should be divided by small pieces of $3 * 3$.

Step2 Determine the 8 initial cluster center block points. Because k-means is sensitive to isolated points, this paper calculates the image block without the case of extreme value.

The model is to average the pixels of a 3*3 small block $mean_k = \frac{1}{9} \sum_{i=1}^3 \sum_{j=1}^3 x_{ij}$. After we sort the mean of these pieces. Firstly, removing the biggest and smallest value among means, and then getting the corresponding small block by arithmetic sequence, take them as the initial cluster center block $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8)$.

Step3 Learn the convolution kernel parameters using the k-means model.

Step4 Repeat the steps above until you have the convolution kernel parameter of all convolution layers. It is proved by many experiments that this model improves the accuracy and speed.

Step5 Reduce the similar feature maps. Calculate maps using formula (2) in each convolution layer.

$$H = \frac{\left| \sum_{i=1}^n a_i b_i \right|}{\sqrt{\sum_{k=1}^n a_k^2 \sum_{k=1}^n b_k^2}} \quad (2)$$

H is the absolute value of the cosine similarity of the two feature maps. $A(a_{11}, a_{12}, a_{13}, \dots, a_{1n})$ and $B(b_{11}, b_{12}, b_{13}, \dots, b_{1n})$, Those two vectors are respectively carried out in the rows of any two feature graph matrices. For any two feature maps, firstly calculate the parameter H , and judge its value with the custom size a , if a is less than H , then the figure map compared is redundancy, which should be discarded, otherwise the figure map would go into the pooling layer for processing.

3.2 Changing of master-slaves communication conflict in distributed gradient descent

The CNN requires a large number of iterations to complete the training. Compared with the Hadoop framework, the Spark platform is very suitable for iterative computation because of the memory based feature. This paper mainly improves the conflict between master and slaves in distributed gradient descent. The common distributed gradient descent model can be divided into two categories: data parallel and model parallel. After the study of literature [9, 10], the data parallel model was used in this paper. The stochastic gradient descent model of classical data parallelism realized by synchronous communication between master and slave, but there is bound to be a lot of synchronization delay. A large number of synchronous operations make it impossible for the algorithm to achieve true parallelism. In order to reduce the amount of synchronous communication data among master and slaves, this paper proposes a model based on the threshold division of the communication data. Back propagation is the inverse process of forward propagation, and the corresponding parameters of the l layer are calculated by the $l+1$ layer. The calculation process involves the derivative calculate of the convolution layer and the pooling layer. Suppose the loss function we use is:

$$E^N = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^c (t_k^n - y_k^n)^2 \quad (3)$$

t_k^n is the k th dimension of the label of the n th sample. y_k^n is the k th output value of the n th sample.

Each slave opens a listener process before calculating the loss function error. This process is responsible for collecting the gradient data passed by other slaves, and then passing the sum to the master node. According to the formula of loss function (3), the error

is compared with the custom threshold b . If $E^N < b$, this indicates that the recognition accuracy is high and the distribution weight is 0.55, or the weight is 0.45, which is determined by the experiment. Because in distributed computing, not all results will play a key role, so the calculation results can be weighted differently. This process, if it is the first to compute the result of the loss function, is used as the group leader. The team leader slave process collects the results from other slaves in this group and summarizes them according to the formula (4).

$$sum = 0.55 * \sum_{i=1}^P x_i \quad \text{or} \quad sum = 0.45 * \sum_{i=1}^P x_i \tag{4}$$

The parameter sum is the sum of the gradient data of this group; P is the number of slaves in this group, x_i is the gradient error of the i th slave iteration, the number of 0.55 and 0.45 represent weights. The group leader summarized and reported to the master. The detail Procession is shown in figure 2:

Fig. 2. Improved distributed gradient descent model.

In each iteration, the master randomly select a certain percentage of the sample and distribute to the slave nodes, slave starts a listening process, if it is the first to complete the calculation for loss function, this process will send information to the master and be choose as a leader, such as above figure2, slave2 and slave3 are two team leaders. The process is responsible for recording the result of the graduate data of this iteration and collecting the data from other slaves in the same group. For example, slave1 sends the results to salve2, and slave4 sends the gradient results to slave3. A threshold b is set for the loss function, that slaves are divided into two groups according to b . If the error is less than b , which proves the recognition accuracy is high, the gradient calculation results of the group are gave weight 0.55, or shows the identification degree is low, conversely give weight 0.45, these two weight are obtained from the experiment. The two leaders aggregate the results and send them to the master node. The master node then accumulates the gradient of the slave samples received, calculates the average gradient of the sample, and finally updates the weight according to the latest gradient and the weight of the last iteration. The algorithm is described as follows.

Input:Dataset,batchSize,slaveNumber, epoch
Output: model parameter weight
Master:
1. submitJob()
2. weight ← initWeight()
3. spiltData(datas, N)
4. For i ← 1 to N parallel do
5. receiveErrorFromSlaves(error,slaveNumber)
6. sortSlavesAndChooseTwoLeaders()
7. sendLeadersToSlaves()
8. Δw ← ReceiveFromLeaderSlaves()
9. weight ← weight - α*Δw
10. sendWeightToSlaves()
11. End For

Slave :
12. P ← ListenProcess() 13. For j ← 1 to epoch do 14. For i ← 1 to N parallel do 15. weight ← getWeightFromMaster() 16. error ← forwardError() 17. P.sendToMaster(error, slaveNumber) 18. P.getLeadersFromMaster() 19. Δweight ← miniGradient(batch,weight) 20. If(P. isLeader()) 21. for j ← 1 to slaves - 2 do 22. n, Δw← P.receiveOtherSlave(num, Δw) 23. If(n == P.slaveNumber) 24. sum←sum+Δw 25. End if 26. End for 27. If(P. err > b) 28. sum ←-0.45 *(sum + Δweight) 29. Else 30. sum ←-0.55 *(sum + Δweight) 31. End if 32. P.sendToMaster(sum) 33. Else 34. P.sendToSlaverLeader(num, Δw) 35. End if 36. Δweight ← 0 37. End for 38. End for

4 Experimental results and analysis

Model in this paper is aimed at the large-scale image classification. In order to prove the validity of the model and the computing advantages in the distribution platform , In this paper, three experimental comparisons are made on the effect of the different ways to initial convolution kernel parameters, the threshold parameters of the slaves group, and the acceleration ratio of the model.

4.1 Experimental hardware environment and datasets

The experiment platform consists of 4 Lenovo computers. Each of the memory is 16G, the processor is Intel(R) Core dual-core processor, the hard disk is 500G, and the operating system adopts Centos7 and Spark version 1.6.0. And the data set used in this experiment is cifar-10, consisting of 60,000 RGB color images of 32* 32, with 10 categories. 50,000 pieces of training, 10,000 tests (cross validation)

4.2 Preprocessing

Because this method is a data parallel strategy for data set processing, it is necessary to preprocess the image before the improved model training. That is, the data set is traversed, and the *i*th image pixel data is converted to a vector, and the transformation format is *V_i*

(label, data). Write V_i as a row of Data to the Data file; The label is a tuple $(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10)$, parameter values to 0 or 1, 1 representative categories for the position. Upload the data file to the HDFS. Using Spark to read files from HDFS, the map operation of the file makes it possible to parallel the operation and convert the format to (label, Matrix).

4.3 The effect of the initialization method of convolution kernel parameters on the overall accuracy

In order to demonstrate the effectiveness of the improvement, the experiment was designed and compared with CNN using random and k-means to initialize convolution kernel parameters. In this paper, the optimal convolution neural network structure obtained through the experiment is 9 layers of network structure, in which the first layer is input layer. The second, fourth and sixth layers are the convolution layer, using a filter of 5×5 , which is 8 and 16 and 14 respectively. The third and fifth layers are the pooling layer. The seventh and eighth are full connected layers, and the softmax function is adopted. The ninth layer is the output layer. The experimental results are shown in table 1, and the Overall accuracy (OA) is the Overall evaluation of the classification performance of the classifier. Definition of OA as formula (5):

$$p = \sum_{i=1}^c \frac{n_{ii}}{M} \tag{5}$$

where c is the number of samples, $M = \sum_{i=1}^c \sum_{j=1}^c n_{ij}$ is the total number of samples tested. n_{ii} is the value of the diagonal of the confusion matrix.

Table 1. Effects of different parameters initialization methods on OA.

Model	Epoch	OA(%)
Rand-CNN	212	70.87
KMean-CNN	205	74.22
Dis-CNN	198	76.25

The number of epoch in Table 1 is the best epoch at the highest level of OA on the distributed platform. Rand-CNN represents the common CNN using random numbers to initialize the convolution kernel parameters, and KMean-CNN represents the K-means initialization convolution kernel parameters used in the literature [12]. It can be seen that the parameter initialization method Dis-CNN is improved in this paper, the OA is the highest and the number of epoch is the least. Analysis of experimental results on the table 1, which shows that using the Rand-CNN model, there will be falling into local extreme point; with the KMean-CNN model, there will be more sensitive to noise phenomenon, this Dis-CNN model used for compromise between the two, each convolution kernel is obtained by the random number and K-means ways, which make full use of the advantages of both, so epoch is fewer, the OA also increased.

4.4 The influence of master-slaves classification threshold on the recall rate of Dis-CNN

First to discard the similar feature maps in this paper, and then to reduce the traffic master and slaves, finally the weights of slaves classification threshold is given, so you also need to analyze different threshold effects on Dis-CNN model. The similarity threshold of each feature graph can correspond to different classification thresholds of slaves, thus obtaining different recall rates. Through a large number of experiments, several excellent thresholds are selected, and the results are shown in table 2. In the table, the letter of a is the similarity

threshold of the feature maps, b is the threshold for slaves classification, and the Recall in the table 2 is the average recall of all kinds of images. When the values of a and b are 0.25 and 0.5 respectively, Dis-CNN has the highest recall rate, so the recall rate is not increased with the increase of the threshold. When both thresholds are 0, the CNN model is converted, and the recall rate is worse than Dis-CNN. Because our model has an initial analysis of the convolution kernel parameter, the training parameter is avoided to a local extreme value. The redundancy of feature graph is processed to make more concentrated contribution. For distributed training, use weights to distinguish different training errors, so as to provide different contributions, so that the iterative error can be further reduced. Based on the above three points, we can demonstrate that the performance of this model is better in the classification of large scale images.

Table 2. Effects of threshold on recall rate.

a	b	Dis-CNN Recall(%)
0.15	0.5	73.5
	0.8	75.2
0.20	0.5	74.3
	0.8	78.5
0.25	0.5	78.8
	0.8	76.2
0	0	75.6

4.5 Acceleration ratio

From above we can see, the Dis-CNN model for image classification compared with the traditional method has higher precision and recall rate, but the model using the technology of distributed design, so you should consider its parallelism. The commonly used indicator is the acceleration ratio of the algorithm. Through the experiment analysis, the acceleration is shown in figure 3. Acceleration ratio is defined as formula (6)

$$Speedrate(n)=\frac{t1}{tn} \tag{6}$$

$t1$ is the time consuming for single node and tn is time consuming for n work nodes.

Fig. 3. Acceleration ratio of different nodes.

For the above data analysis, the acceleration ratio has a significant upward trend as the number of nodes increases. For small data volumes, the more nodes, the communication data between nodes also increases. In figure 3, when the image sample is 5000, the speedup is almost unchanged. For a large amount of data, when the image sample is 45000, speedup in proportion to rise, the communication data between nodes is much smaller than the training data, so as the training is accelerated. So it can be analyzed from the acceleration ratio that when the amount of the sample data is large, the more the number of nodes, the greater the acceleration ratio, which shows that the model has good parallel performance in the training of large scale image data.

5 Conclusion

This paper combines the advantages of CNN and spark platform, On the basis of classical convolution neural network model, the parameters initialization and hiding layer are improved, and the training complexity is reduced. In the process of CNN distributed gradient descent, to the problems of synchronous communication delays between master and slaves, this paper puts forward a distributed gradient descent optimization, to reduce the synchronous communication of master and slaves during the gradient propagation, and give different weights to the classified slaves. By experiment comparison, the improved model has high accuracy, generalization ability and good parallelism for large-scale image classification. However, the model has several limitations, researches of the parallel model is necessary in the future.

Acknowledgments

This work was supported by the National Key Technology Research and Development Program of China (2014BAH09F00), National Natural Science Foundation of China (61379079).

References

1. Krizhevsky A, Sutskever I and Hinton G E 2012 *Advances in Neural Information Processing Systems* 1097-105
2. Zeiler M D and Fergus R 2014 *13th European Conf. Computer Vision* vol **8689** (Zurich: Springer Int Publishing Ag) pp 818-33
3. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 *Conf. on Computer Vision and Pattern Recognition* (Boston: IEEE) pp 1-9
4. Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 *Journal of Machine Learning Research* 15 1929-58
5. Liang T, Xu X and Xiao P 2017 *Pattern Recognition Letters* 94 105-11
6. Fu M, Xu P, Li X, Li X, Liu Q, Ye M and Zhu C 2015 *Engineering Applications of Artificial Intelligence* 43 81-88
7. He K, Zhang X, Ren S and Sun J 2015 *Int. Conf. on Computer Vision* (Boston: IEEE) pp 1026-34
8. Mihaita D and Stefan G M 2017 *Romanian Journal of Information Science and Technology* 20 186-97
9. Beri T, Bansal S and Kumar S 2017 *IEEE Transactions on Parallel and Distributed Systems* 28 1518-34
10. Zhang S, Zhang C, You Z, Zheng R and Xu B 2013 *Int. Conf. on Acoustics Speech and Signal Processing* (Vancouver: IEEE) pp 6660-3
11. Coates A, Carpenter B, Case C, Satheesh S, Suresh B, Wang T, Wu DJ and Ng AY 2011 *Proc. Int. Conf. on Document Analysis and Recognition* (Beijing: IEEE COMPUTER SOC) pp 440-5
12. Tabik S, Peralta D, Herrera-Poyatos A and Herrera F 2017 *International Journal of Computational Intelligence Systems* 10 555-68