Clustering-Based Edge Compression Method with Application to Electromagnetic Object Recognition

Ziyan Yan[1], Qibin Zheng[2], Yun Lin[3], Qingjiang Shi[4]
{1831577@tongji.edu.cn[1], zhengqb_2018@tongji.edu.cn[2], linyun@hrbeu.edu.cn[3]}

School of Software Engineering, Tongji University, Shanghai, 201804, China[1,2,4]
School of Information and Communication Engineering, Harbin Engineering University, Harbin, 150001, China[3]

Abstract. Collaborative intelligence has attracted more and more attention. By properly portioning deep neural networks (DNN) and distributing the DNN calculation to the edge and cloud, we could reduce the prediction delay and power consumption to meet the actual application requirements. Toward this direction, this paper proposes an edge compression method based on clustering to address the issue of high data communication cost and time delay between the edge and cloud. Specifically, by using K-means clustering algorithm, this method compresses the output layer of the edge DNN, reducing the amount of transmission data and thus the delay and energy consumption. Based on the compression-based edge-cloud collaboration paradigm, we propose a distributed inference scheme for electromagnetic object recognition. The simulation results show that the proposed method can greatly reduce communication cost while maintaining the prediction performance.

Keywords: deep learning, object detection, edge compression, clustering.

1 Introduction

Deep learning technology has been widely used in the field of image / video object detection and recognition, and achieved good prediction results [1][2] since 2012. Traditional DNN-based prediction methods are generally performed in two ways: transmitting the original data to the cloud for prediction, or directly do the prediction on the edge device. However, the former brings great pressure on the communication bandwidth, which will result in serious delay, while in the latter the prediction capability can be constrained by the device performance and power consumption.

1.1 Collaborative intelligence

Considering the problems and limitations of the above two methods, a "collaborative intelligence" method [3] was proposed in related research works, which can be used to optimize the delay and energy consumption of prediction tasks. This work first studies the significant differences in computing time and output data size of each network layer of AlexNet [4], as shown in **Figure 1**. It can be seen that in computing time, fc6 and fc7 of full connection layer are significantly more than those of other layers, while pool1~pool5 of pooling layer can significantly reduce the data size, and the size after pool5 layer is smaller

EAI MOBIMEDIA 2020, August 27-28, Harbin, People's Republic of China

DOI 10.4108/eai.27-8-2020.2294473

than the original input. In addition, this work also compares the total delay when a prediction task adopts pure cloud mode (only in the cloud center) and pure edge device mode (only in the edge device). The results show that the communication time accounts for more than 94.1% of the total time in pure cloud mode, and the computing time in pure edge device mode is much longer than that in pure cloud mode.

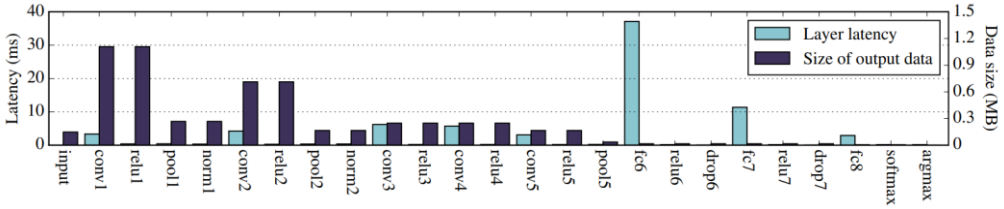

Fig. 1. Calculation time and output data size of each layer of AlexNet [3].

Based on the above results, as shown in **Figure 2**, collaborative intelligence divides the neural network into two parts, so as to share the computing to the edge device and cloud center. This method has two main advantages: 1) the edge device only needs to upload the output data of the hidden layer of the neural network, so that when the output of the hidden layer is less than the original data, it can greatly reduce the traffic; 2) selectively put the heavy calculation in the cloud center, reducing the calculation of the edge device. Through the idea of neural network partition, collaborative intelligence integrates the advantages of pure cloud mode and pure edge device mode, so as to achieve more outstanding performance indicators.

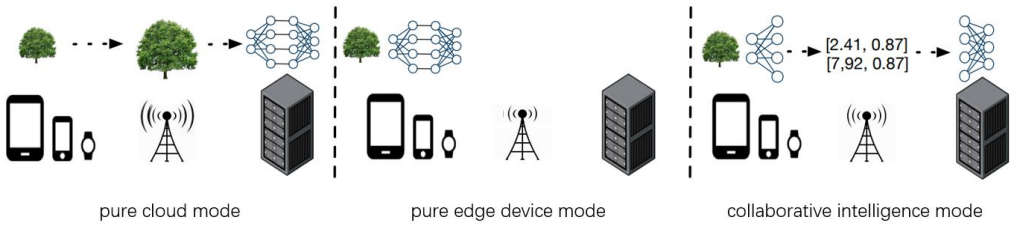

Fig. 2. Three ways to perform prediction tasks [3].

However, the cooperative intelligence [3] does not consider the compression operation of the edge transmission data, which can result in an increase in the network communication delay (also energy consumption) caused by the output of the hidden layer of the neural network when the output of the hidden layer is larger than the original data. Therefore, it can be expected that DNN-based collaborative intelligence calls for some methods to achieve good accuracy but with less overhead.

1.2 Edge compression

Edge compression refers to the use of data compression method to compress the output of edge partition neural network on the basis of collaborative intelligence, so as to effectively reduce the traffic, communication delay and energy consumption.

The authors in [5] researched the output data characteristics of CNN's first convolution layer, chose to use 8 bits to quantize the floating-point number of each unit of the output data, effectively reducing the number of bits of the transmitted data, and then use the lossless image coding technology PNG to compress the quantized data. The experimental results show that the average compression rate of the method is 28.57%, and the performance improvement in delay and energy consumption is 4.9 times and 4.6 times respectively. The work in [6] studies the impact of lossless compression / lossy compression on the hidden layer, and conducts experiments based on the object detection task of the yolo9000 model [7]. Experimental results show that the influence of quantization combined with lossless compression on object detection accuracy can be ignored, while lossy compression can get higher compression rate, but it also affects detection accuracy. The detection accuracy can be effectively improved by using compression enhanced training method. The authors in [8] investigate the difference between deep feature data and natural image data, and propose a simple and effective near lossless deep feature compressor. Compared with HEVCIntra, the bit rate of this method is reduced by 5%, which is much lower than other commonly used image codecs. In [9], a computing architecture based on collaborative intelligence is designed. The architecture introduces a unit, which is composed of a separate convolution layer. It can reduce the dimension of hidden layer output of CNN dividing points, so as to achieve the compression effect. Based on resnet-50 model [10], the experimental results show that the performance of this method in terms of delay and energy consumption is improved by 53 times and 68 times respectively.

1.3 Contribution of this paper

Based on the above research, this paper studies the edge compression of CNN model, and proposes a cluster-based edge compression method (CECM). This method can use clustering algorithm to compress the traffic, and then reduce the delay and energy consumption during the transmission process. In addition, we customize this method with a particular application to the electromagnetic object recognition, and the experiment shows that the method can effectively reduce the transmission traffic and energy consumption on the premise that the accuracy of the original neural network is almost not significantly deteriorated.

In the following content of this paper, Chapter 2 describes the detailed steps of edge compression method based on clustering. Chapter 3 describes the application scenario of the method in electromagnetic object recognition. Chapter 4 evaluates and verifies the performance of the method through simulation experiments. Finally, Chapter 5 concludes this paper.

2 Edge compression method based on clustering

As previously mentioned, the neural network can be divided into two parts, which are deployed separately on the cloud and edge devices, named cloud DNN and edge DNN for easy of exposition. The forward calculation of the DNN is accomplished by relaying the output of the edge DNN to the cloud followed by the rest forward calculation of the cloud NN. This method can take advantage of the cloud and the edge device. However, it may incur large communication overhead if the output of the edge DNN has high dimension. Hence, we should properly design the interface between the edge and cloud and special attention should

be paid to how to downsize the amount of data transmission. Obviously, if the data transmitted by the edge is considered to be compressed, even if the output data of the hidden layer of the neural network is larger than the original data, it can effectively reduce the traffic and the delay. Thus, our method is proposed to reduce the amount of data transmission in the communication process by edge compression based on clustering. We perform K-means clustering on the output of the edge DNN, and let the edge send the clustering results to the cloud, so that the amount of data during the communication process can be greatly reduced. The detailed steps of Cluster-based Edge Compression Method (CECM) are shown as follows:

Step1. Firstly, a high-precision deep neural network model is trained, and divided into two sub neural networks N_{e} and N_{c} according to the network layer, where we deploy N_{e} in the edge device and N_{c} in the cloud center.

Step2. After the edge device processes the input data through N_{e}, it generates the m-dimensional output vector $\boldsymbol{v}_{\mathrm{e}} = [x_1, x_2, \dots, x_m]^T$, where $x_i (i = 1,2, \dots, m)$ are floating-point number. We apply K-means clustering [11] to the set $\{x_{\mathrm{i}} | i = 1,2, \dots, m\}$ which all elements $x_i (i = 1,2, \dots, m)$ from the output vector $\boldsymbol{v}_{\mathrm{e}}$ compose the collection, generate k clusters $S = \{S_1, S_2, \dots, S_k\} (1 \le k \le m)$ and obtain the k-dimensional vector $\boldsymbol{v}_1 = [\mu_1, \mu_2, \dots, \mu_k]^T$ composed of the center points $\mu_j (j = 1,2, \dots, k)$ from each cluster, where

$$\mu_j = \frac{1}{|S_j|} \sum_{y \in S_j} y, \quad j = 1,2, \dots, k . \tag{1}$$

Where $|S_j|$ is the number of elements in the cluster S_j, y is the element of cluster S_j. Then, each cluster is labeled with a different label $1 \sim k$, so that the output vector of edge device $\boldsymbol{v}_{\mathrm{e}}$ can be mapped to m-dimension label vector $\boldsymbol{v}_2 = [l_1, l_2, \dots, l_m]^T$, Where

$$l_i = \underset{j}{\operatorname{argmin}} \| x_i - \mu_j \|, \quad i = 1,2, \dots, m, \ \ j = 1,2, \dots, k . \tag{2}$$

It can be seen from (2) that $l_i \in \{1,2, \dots, k\}$, we encode $1 \sim k$ with binary, thus $\lceil log_2 k \rceil$ bits are needed at most to represent l_i. Then the transmission capacity can be reduced compared with the floating-point form of direct transmission.

Step3. The edge device transmits the vector $\boldsymbol{v}_1$ composed of cluster center points and m-dimension label vector $\boldsymbol{v}_2$ to the cloud center.

Step4. After the cloud center receives the vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, replace the label in the vector $\boldsymbol{v}_2$ with the center point coordinate in the vector $\boldsymbol{v}_1$ to get the m-dimension vector $\boldsymbol{v}_{\mathrm{c}} = [\mu_1', \mu_2', \dots, \mu_m']^T$, which is the input of the cloud center sub neural network, Where

$$\mu_i' = \mu_{l_i}, \quad i = 1,2, \dots, m . \tag{3}$$

The cloud center will process according to the input, and finally get the corresponding recognition results.

Step5. Repeat step 4 for p times to get the average performance (recognition speed and power consumption) of the current overall depth neural network model, $p \in [3, 20]$. Then adjust the partition point of the neural network and k, and repeat steps 1 to 4 t times ($t \in [50, 150]$), finally select the deep neural network model with the best performance.

Figure 3 shows the network structure schematic diagram of the distributed neural network. In **Figure 3**, the content of the red box indicates that the output vector of the specified layer of neural network is clustered (the number of clusters k is 3) to obtain the vector $\boldsymbol{v}_1 = (0.3, 1.5, 2.7)$ which is composed of three cluster center points, then we use the center point to approximate the original output vector as the input of the next layer, so as to reduce the amount of data in the communication process.

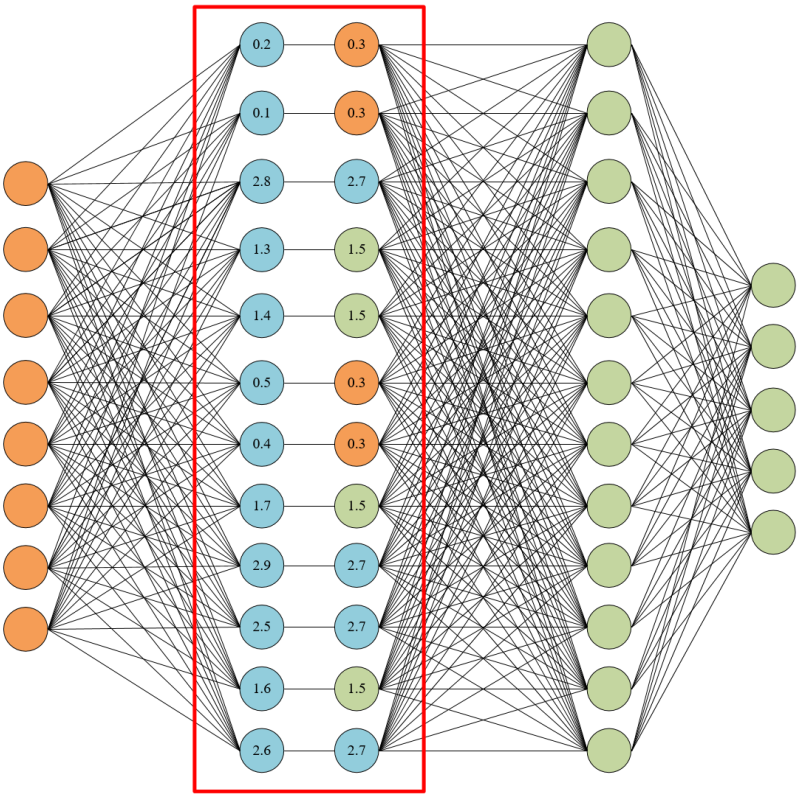

Fig. 3. Clustering and approximate representation of hidden layer output.

3 Application in electromagnetic object recognition

In the era of information-based war, with the development of new electronic and information technology, the electromagnetic object (such as airplane, warship, etc.) often carries many kinds of electronic equipment. However, due to their characteristics of stealth and high speed, these kinds of electromagnetic objects make the recognition system unable to easily and quickly recognize them and react in time. Such situation greatly increases the risk of attack. Therefore, it is required that the recognition of electromagnetic object not only needs to distinguish accurately but also needs to distinguish quickly, so as to make timely defense and attack the target object effectively.

As shown in **Figure 4**, we give an example of the workflow of using this CECM method to identify electromagnetic objects in the hypothetical battlefield. Firstly, the reconnaissance equipment (edge equipment) is used as the acquisition node to collect the signal from the electromagnetic object. After collection, the data is processed preliminarily and the intermediate results are output through the neural network deployed on the edge device. Then the edge devices compress the intermediate results by K-meaning clustering, and foward the compressed data to the cloud center through a communication link. After receiving the data from the edge node, the cloud center decompresses it, and takes the decompressed content as the input of the remaining neural network. Then the final recognition result can be obtained after some processing.

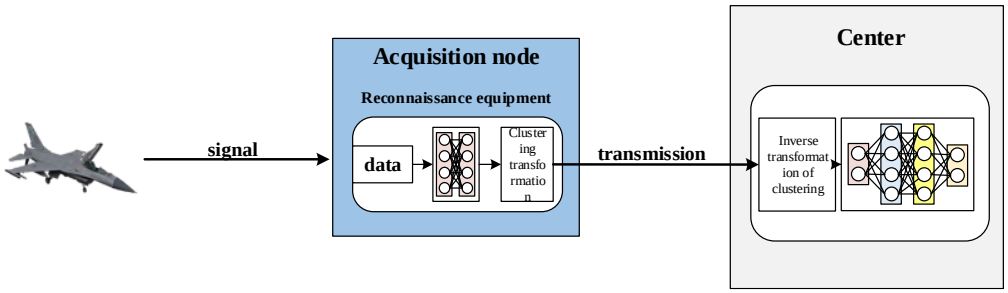

Fig. 4. Schematic diagram of distributed identification of electromagnetic object.

4 Experiment section

In this section, some simulations are carried out to validate our proposed method by focusing on a communication modulation recognition task. Note that modulation classification can be viewed as a part of object recognition task. The used dataset and the simulation setup is firstly introduced, followed by some simulation results.

4.1 Dataset and Simulation Setup

RadioML2016.10a [12] is a synthetic dataset generated by GNU radio software, which consists of 11 modulation modes with different SNR (8 digital modulation and 3 analog modulation). The dataset can be used for modulation pattern recognition. We divide the dataset into two parts, 80% of which is the training set and 20% of which is the test set. The specific information is shown in Table 1:

Table 1. Dataset Description

Parameter	Value
Total number of samples	220000
Number of training set samples	176000
Number of test set samples	4400
Feature dimension	(2,128)

Category (modulation mode)	11
Modulation mode value	8PSK, AM-DSB, AM-SSB, BPSK, CPFSK, GFSK, PAM4, QAM16, QAM64, QPSK, WBFM
Number of Signal to Noise Ratio (SNR)	20
SNR Value	-20, -18, -16, -14, -12, -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

We use convolutional neural network VT-CNN2 [13] as a performance baseline. At the same time, on the basis of VT-CNN2, the proposed method CECM is obtained by clustering compression. Next, we compare and verify the performance of the two methods by experiments.

4.2 Prediction Accuracy

As shown in **Figure 5**, we compare VT-CNN2 with CECM, and draw the prediction accuracy curve of each cluster when k is 2~10. Among them, dotted curve and solid curve represent the prediction accuracy of VT-CNN2 and CECM under different Signal to Noise Ratio (SNR) respectively. It can be seen from the **Figure 5** that when SNR increases, the accuracy of VT-CNN2 gradually increases, and the accuracy changes insignificantly after SNR ≥ 0, maintaining at 70%~75%; in general, CECM keeps the similar change trend with VT-CNN2; with the increase of k, the two curves gradually coincide, and the accuracy loss between CECM and VT-CNN2 decreases. When k is greater than 2, the change of recognition accuracy is almost stable.

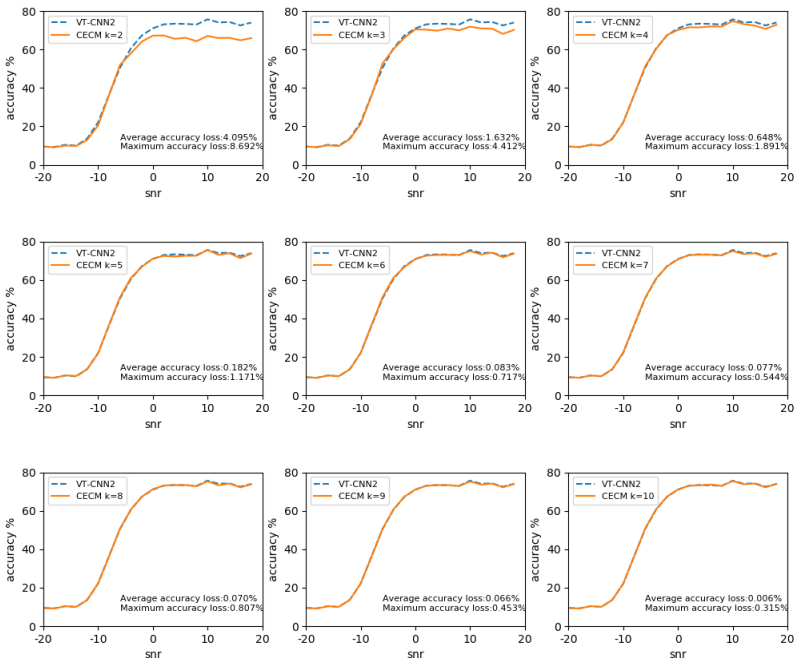

Fig. 5. Prediction accuracy comparison between CECM and VT-CNN2.

Figure 6 compares the confusion matrix of VT-CNN2, CECM (k = 2) and CECM (k = 10). In diagram, the vertical coordinate in the **Figure 6** represents the real category of test samples, and the horizontal coordinate represents the category of predicted results of test samples. The element in row i and column j of confusion matrix indicates the proportion of samples of class i predicted as class j in all samples of class i. The larger k is, the higher similarity the two methods have.

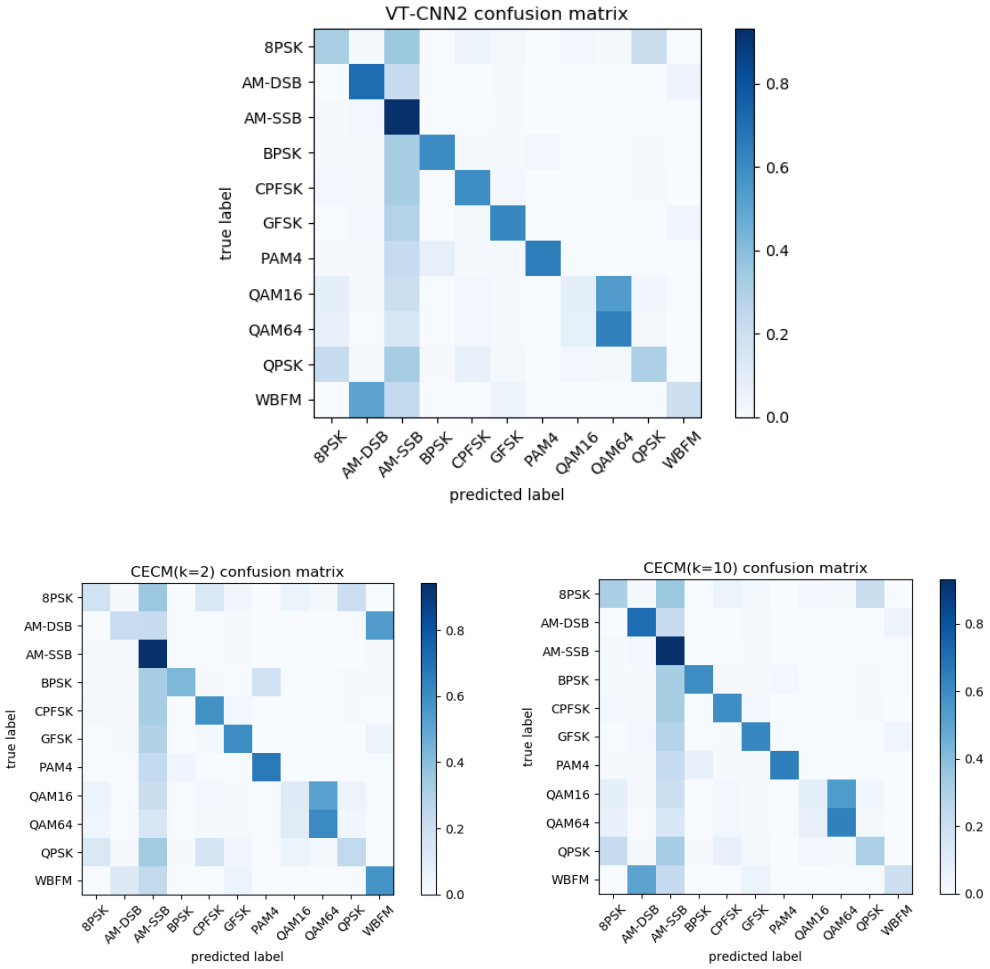

Fig. 6. Confusion matrix of VT-CNN2, CECM (k = 2) and CECM (k = 10).

It can be seen from the above experiments that, when the total number of classes k increases, CECM has recognition performance close to VT-CNN2. Particularly, when k is greater than 2, their performance is almost identical.

4.3 Compression ratio

Now let us look at the compression ratio of these two methods. From the specific implementation steps in Chapter 2, it can be seen that the dimensions of the output vector $\boldsymbol{v}_{\mathrm{e}}$ of the edge device and the label vector $\boldsymbol{v_2}$ after clustering conversion are both m. The dimensions of the center vector $\boldsymbol{v_1}$ after clustering conversion are k. It can be concluded that without cluster conversion, the data transmission between the edge device and the cloud center in the original model is: $m \cdot B$, where B denotes the number of data bits occupied by a floating nuber. While after clustering transformation in our scheme, the amount of data transmission is: $k \cdot B + m\lceil log_2 k \rceil$. Therefore, the compression ratio of CECM method is:

$$\frac{\text{data transmission in our scheme}}{\text{data transmission without compression}} = \frac{k \cdot B + m\lceil \log_2 k \rceil}{m \cdot B}$$

To be specific, let us consider the case when B =32 and m =256, and outline the compression ratio for k=2~10 in **Table 2**.

Table 2. B=32, m=256, the data compression ratio when k is 2~10.

Cluster number k	Compression ratio
2	3.91%
3	7.42%
4	7.81%
5	11.33%
6	11.72%
7	12.11%
8	12.50%
9	16.02%
10	16.41%

4.4 Tradeoff between accuracy and compression

As shown in **Figure 7**, we can see that, with the increase of cluster number k, the recognition accuracy increases, but the compression ratio will increase accordingly. Particularly, when k = 4, we obtain the best tradeoff between accuracy and compression.

5 Conclusion

Referring to the idea of collaborative intelligence and edge compression, this paper proposes an edge compression method based on clustering, which can reduce the amount of transmission data. The experimental results show that this method can greatly reduce the communication overhead while maintaining the prediction performance. In the future, we will consider a possible extension of this method to the case of multiple edge nodes.

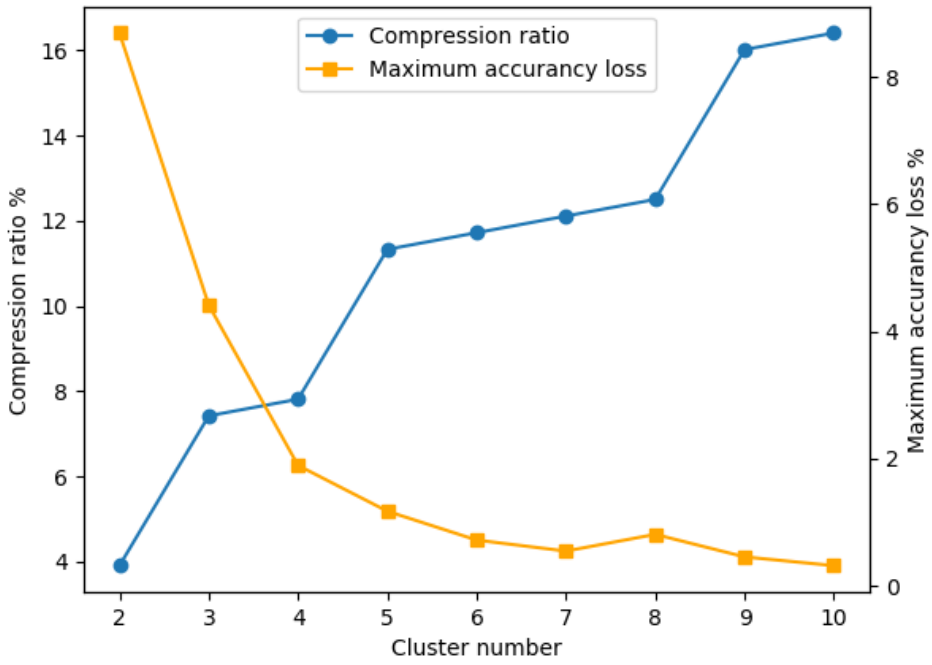

Fig. 7. The maximum precision loss and compression ratio changes with number of cluster k.

Acknowledgments. This work was supported in part by the National Key Research and Development Project under grant 2017YFE0119300, the NSFC under Grants 61671411, and by the Fundamental Research Funds for the Central Universities under grant 22120180113.

References

[1] X, Zhou. D, Wang. and P, Krahenb.: Objects as points. arXiv preprint arXiv:1904.07850(2019)

[2] S, Kanimozhi, G, Gayathri, T, Mala.: Multiple Real-time object identification using Single shot Multi-Box detection. International Conference on Computational Intelligence in Data Science (ICCIDS) (2019)

[3] Kang, Y, Hauswald, J. Gao, C. et al.: Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. Acm Sigplan Notices. 52(4), pp. 615-629 (2017)

[4] Krizhevsky, A. Sutskever, I. Hinton, G, E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. pp. 1097-1105 (2012)

[5] Eshratifar, A, E. Abrishami, M, S. Pedram, M.: JointDNN: an efficient training and inference engine for intelligent mobile cloud computing services. arXiv preprint arXiv:1801.08618 (2018)

[6] Choi, H. Bajić, I, V.: Deep feature compression for collaborative object detection. 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE. pp. 3743-3747 (2018)
[7] Redmon, J. Farhadi, A.: YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263-7271 (2017)
[8] Choi, H. Bajić, I, V.: Near-lossless deep feature compression for collaborative intelligence. 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP). IEEE. pp. 1-6 (2018)
[9] Eshratifar, A, E. Esmaili, A. Pedram, M.: Towards collaborative intelligence friendly architectures for deep learning. 20th International Symposium on Quality Electronic Design (ISQED). IEEE. pp. 14-19 (2019)
[10] He, K. Zhang, X. Ren, S. et al.: Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770-778 (2016)
[11] k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.
[12] O'shea, T, J.: West N. Radio machine learning dataset generation with gnu radio. Proceedings of the GNU Radio Conference. 1(1) (2016)
[13] O'Shea, T, J. Corgan, J. Clancy, T, C.: Convolutional radio modulation recognition networks. Inte-rnational conference on engineering applications of neural networks. Springer, Cham. pp. 213-226 (2016)