Evaluation of Distributed Processing of Caffe Framework Using Poor Performance Device

Ayae Ichinose
and Masato Oguchi
Ochanomizu University

Atsuko Takefusa
National Institute of Informatics

Hidemoto Nakada
National Institute
of Advanced Industrial Science
and Technology (AIST)

I. Introduction

The spread of various sensors and Cloud technologies has made it easy to acquire life-logs and accumulate data. As a result, many life-log analysis applications, which transfer data from sensors, especially cameras to a Cloud and analyze them in the Cloud, have been developed. Cameras with a server function called network cameras have become cheap and readily available for security services and the monitoring of pets and children from remote locations. In these services, raw data from sensors, including cameras, are generally transferred to a Cloud and processed there. However, it is difficult to transfer raw data from sensors to a Cloud because of the limitation of network bandwidth between sensors and a Cloud and privacy issues caused by sending raw sensor data to a Cloud.

In our study, we split a deep learning processing sequence of the Caffe framework [1] by defining new layers and performs distributed processing between a client side and a Cloud side in a pipeline manner. This approach makes it possible to protect privacy by sending not raw data but feature values, and reduce transferred data between a sensor and a Cloud. In the experiments, we investigate processing times of classification when the parameters of the network models of CIFAR-10 data sets [2] are changed using our method.

II. Deep Learning Framework Caffe

Deep learning is widely used for the recognition of images and sounds. Deep learning makes it possible to automatically perform feature extraction from data; so, it has attracted attention for improving the accuracy and speed. There have been several deep learning frameworks such as Caffe, TensorFlow [3], and Chainer [4]. Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC). Caffe comprises a combination of modules with specific functions such as convolution and pooling, and determines an operation of the whole system through the communication between the modules. This approach can be expanded to new data formats and network layers. The core part of Caffe is written in C++, so, it is possible to use user-friendly image classification tools, such as the Jupyter Notebook implemented in Python, using the Caffe C++ API. In addition, Caffe is capable of high-speed processing because it corresponds to the GPGPUs, and enabling easy execution of experiments using the trained network models provided in the Caffe package.

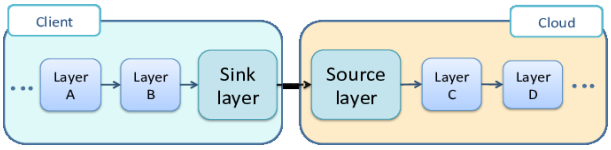

Fig. 1. Proposed pipeline-based distribution method for deep learning.

III. Distributed Deep Learning Framework

We propose a pipeline-based distribution method as shown in Figure 1. We modified Caffe so that we can split a convolution neural network into two portion, the client side and the Cloud side. The client side and the Cloud side are independent processes of Caffe. “Sink” is located at the end of the client side network, and terminates. “Source” is the starting point of the Cloud side network. Corresponding Sink and Source are connected by TCP/IP. Sink receives the data from the upstream layer, transfers it to the paired Source and waits for the ACK from the Source. Source receives data from the Sink, sends ACK and then, forward data to the downstream layers. Note that the client side prosess and the Cloud side process perform computation in parallel in a pipeline manner. Sink specifies the host name and port number of the Cloud side Source process. When one splits a network, more than one link could be cut. In such a case, one should set up a Sink-Source pair for each cut link. The pairs are identified by the port numbers.

This approach makes it possible to protect the privacy of users by not sending raw data but rather sending feature values, and reduce the amount of transferred data between a sensor and a Cloud for low-bandwidth environments.

IV. Experiments

To indicate the effectiveness of the proposed method, we compare the processing times of classification in three cases as follows: performing all processing on the client side, distributing the processing using the proposed method, and performing all processing on the Cloud side. We also investigate the learning accuracies when the parameters of the network model are changed to reduce the amount of data transferred to the Cloud. After we describe the network models and distribution

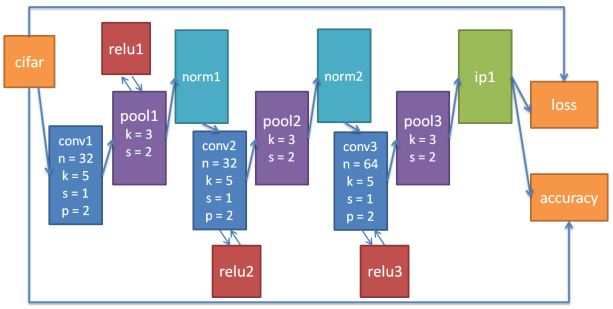

Fig. 2. Network model for CIFAR-10.

TABLE I
PARAMETERS DEFINED IN THE CIFAR-10 NETWORK MODEL.

n : num_output	number of filters
p : pad	width of padding
k : kernel_size	size of each filter
s : stride	interval to apply the filters
g : group	the number of division of the channels

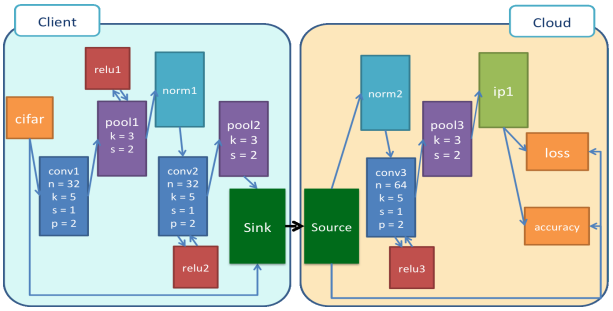

Fig. 3. Distribution method in the experiments.

methods for CIFAR-10 datasets, we show the experimental results.

A. Learning accuracies upon varing the number of filters

CIFAR-10 is a dataset in which images of 32 × 32 pixels are classified into 10 categories, and the network model of one of the datasets is provided by Caffe. The structure of the network model is shown in Figure 2. The parameters defined in each layer are shown in Table I. Caffe stores and communicates data in 4-dimensional arrays as follows: the batch size, the number of channels and the two-dimensional image size. The channel parameters accord with the number of filters in the convolution layer just before that. In the CIFAR-10 network model, the amount of data becomes smaller after the pool2 layer, so we split the network between the pool2 layer and the norm2 layer as shown in Figure 3.

We reduce the number of the filters of the conv2 layer to reduce the amount of transferred data. This may decrease the accuracy of recognition; so, we investigate the accuracies when we change the number of filters and thereby reduce the amount of transferred data. The default value of the number of filters is 32. The correspondence between the number of

TABLE II
CORRESPONDENCE BETWEEN THE NUMBER OF FILTERS, THE AMOUNT OF DATA (KB) AND THE ACCURACIES (%) IN THE EXPERIMENTS.

filters	4	8	12	16	20	24	28	32
amount of data	25.6	51.2	76.8	102.4	128.0	153.6	179.2	204.8
accuracy	73.35	76.56	77.16	77.13	77.71	77.63	77.99	78.11

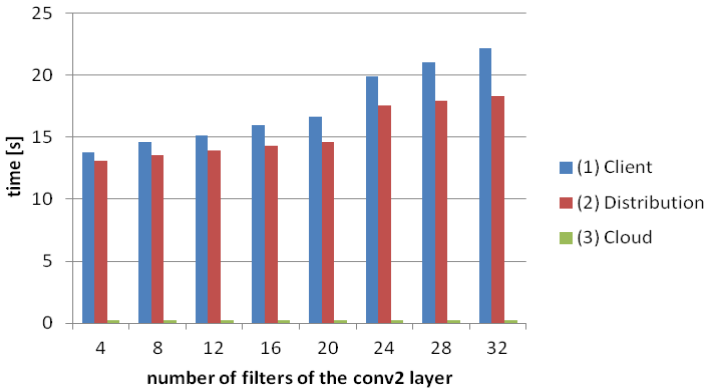

Fig. 4. Comparison of processing time.

filters, the amount of transferred data and the accuracies of the identification are shown in Table II. This shows that the accuracies converge when the numbers of filters are small. Even in the case of the number of filters being 4 at the conv2 layer, the accuracy is maintained at 73.35%, which is comparable to the result of the default state, 78.11%, while the amount of transferred data is one-twelfth of the raw data. Hence, we can see that high accuracy can be maintained even if we reduce the amount of transferred data by reducing the number of filters of the convolution layers.

B. Comparison of Processing Times

We show the effectiveness of the proposed method by measuring processing times of classification of 1 batch using two machines as a client side and a Cloud side, respectively. We use Raspberry Pi as a client side and GPGPU NVIDIA GeForce GTX 980 as a Cloud side. We set the network bandwidth between the two machines to 10Mbps in order to simulate an actual sensor and Cloud network environment.
We compare the three cases; (1) performing all processing in the client side, (2) distributing processing in the both client and Cloud sides using the proposed method and (3) performing all processing in the Cloud side. No data is transferred between the client and the Cloud in the case (1), while processed and filtered data are sent to the Cloud in the case (2), and the raw image data are sent to the Cloud in the case (3). In the cases (2) and (3), the client and the Cloud synchronously work, so we use the processing times measured in the client side, including connection times and waiting times.

Figure 4 shows that the results of the case (2) is faster than that of the case (1) because of the reduced processing on the client side. The case (3) are superior to the other cases because deep learning processing times are longer than the transferred times. However, in the case (3), the size of transferred data is larger than the other cases and privacy issues have not

been resolved. Therefore, the proposed distributed processing method is effective in an actual environment.

V. Conclusions

We propose a pipeline-based distributed processing for deep learning and implemented the distributed processing of the deep learning framework Caffe for the purpose of the sensor data analysis, considering privacy and the network bandwidth. We observed that it is possible to maintain a high accuracy even if we reduce the amount of transferred data from a sensor to a Cloud by reducing the number of filters of the convolution layers. When we take into account network bandwidth between general homes and a Cloud and privacy issues of sensor data, the effectiveness of the proposed method is proven.

Acknowledgment

This paper is partly supported by the New Energy and Industrial Technology Development Organization (NEDO) and JSPS KAKENHI Grant Number JP16K00177.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[2] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, http://download.tensorflow.org/paper/whitepaper2015.pdf. pp. 1-19. [Online]. Available: http://tensorflow.org/

[4] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015, 6 pages.