Towards Edge Computing Based Distributed Data Analytics Framework in Smart Grids

Chunhe Song[1,2(✉)], Tong Li[3], Xu Huang[4], Zhongfeng Wang[1,2], and Peng Zeng[1,2]

[1] Chinese Academy of Sciences, Shenyang Institute of automation, Shenyang 110016, People's Republic of China
songchunhe@sia.cn
[2] Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110016, China
[3] Liaoning Electric Power Research Institute, State Grid Liaoning Electric Power Co., Ltd., Shenyang 110000, People's Republic of China
[4] Shenyang Power Supply Company, State Grid Liaoning Electric Power Co., Ltd., Shenyang 110000, People's Republic of China

Abstract. Edge computing, as an emerging paradigm empower the network edge devices with intelligence, has become a prominent and promising future for Internet of things. Meanwhile, machine learning method, especially deep learning method has experience tremendous success recently in many application scenario. Recently, deep learning method applied in IoT scenario is also explored in many literatures. However, how to combine edge computing and deep learning method to advance the data analytics in smart grids has not been fully studied. To this end, in this paper, we propose ECNN (Edge-deployed Convolution Neural Network) in edge computing assisted smart grids to greatly enhance the ability in data aggregation and analytics. We also discuss how to train such network in edge computing distributively. Experiments shows the advantage of our paradigm.

1 Introduction

Although the prevalent of Internet of Things (IoT) is inevitable, it is also envisioned that IoT will be limited by the network bandwidth, and the IoT will become both provider and consumer of the data, which analyze, process, and store the data at the edge of Internet [1]. Thus, the conventional centralized cloud computing model has reveal its inherent problems. For example, the conventional paradigm could not process the multi-sources massive data at the edge of network in realtime; Both the delay and bandwidth has also come to a bottleneck to satisfy the requirements. Due to above reasons, the traditional cloud computing cannot efficiently support the IoT-based application services and thus, trigger to born of the new computing paradigm edge computing by moving the computation to the data producer side.

X. Sun et al. (Eds.): ICAIS 2019, LNCS 11632, pp. 283–292, 2019.
https://doi.org/10.1007/978-3-030-24274-9_25

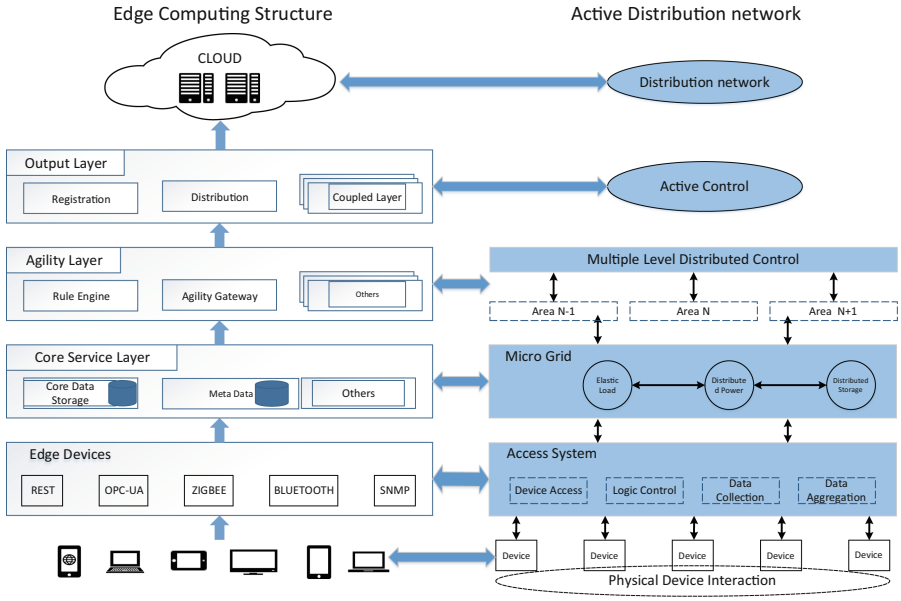

Fig. 1. Edge computing and smart grid system

Meanwhile, as the largest IoT system in real implementation, smart grids also come to the cross that how to improve the sensing ability with the power of artificial intelligence. Among also so called smart functions of smart grids, the data aggregation and analytics is of the most important and fundamental ones. However, the dumb end system has greatly limit the smart grids growing to real "Smart", as the AI algorithms require rather higher computation and sensing capacity of the end devices [2–4]. As such, we argue that, the edge computing paradigm could greatly solve such dilemma by deploying edge computing device close to the smart grids end system and also enable the high sensing and data aggregation functions. In fact, as shown in Fig. 1, edge computing paradigm is a very promising scheme to realized the so called Active Distribution Network of the smart grids, which proactively distributed and balance the power and actively collect the power from different kinds of power station, including not only the traditional ones but also the new power like wind, solar power and etc.

To realize such vision, in this paper, we design a new edge computing based distributed data analytics framework in Smart Grids. As is known to all that, the smart grids is a large scale distributed system with computing and data transmission ability. The framework of a large-scale distributed computing hierarchy provide new significance in the emerging era of IoT. We expected that most of data generated by the IoT devices must be processed locally at the devices or at the edge. Otherwise the amount of smart grid data would overwhelm the network bandwidth and lead to unacceptable processing delay. In comparison,

the distributed computing paradigm offers opportunities for system scalability, data security and privacy, as well as less processing delay [5–7].

On the other hand, deep learning and CNN has illustrate its potential and tremendous advantage in machine learning tasks, especially like image process and etc. Recently, it also shows the effectiveness in sensor analytics. Thus, we are motivated to combine the deep learning and the edge computing to enhance the ability of smart grids. In this paper, we show that Edge-deployed Convolutional Neural Networks (ECNN) can systematically exploit the inherent advantages of a distributed computing hierarchy for CNN applications and achieve similar benefits.

In this paper, we mainly make following contributions:

- We design and propose a new edge computing based framework for smart grids.
- We have utilized the edge-deployed CNN (ECNN) as the core computing technique for our framework.
- We have analysis the advantage of our framework both quantitatively and qualitatively.

2 Related Work

2.1 Deep Learning

Deep learning [8] is firstly proposed as an extension of neural network [9] and with the flourish of the computing paradigm and resources, thus make the traditionally unrealized deep models feasible. Recent research has been paid on how to explore different structure to make such realization more accurate [10,11]. Recent proposed BNN has been shown to achieve good accuracy in MNIST and CIFAR-10 [12] by using less memory and small computation resources inference [13]. These models are especially promising in end devices. The researchers also use deep learning approaches to perform reinforce learning. In ECNN, we inspired by the federate learning [14] techniques and applied them in both the end devices, edge cloud and the central cloud, so that the inference and the training of the model could be performed.

2.2 Distributed Deep Learning

Current research on distributing deep learning is focused on the structure and the training efficiency. DistBelief [15] distribute large NNs over thousands of CPU cores during the training in 2012. Recently, several methods have been proposed to scale up deep NNs training over GPUs [16,17]. In 2017, Surat et al. [18] proposed an distributed deep neural networks structure to fit to embedded devices. It proposes the training and inference deployed over a distributed computing hierarchy, rather than processed in parallel over CPUs or GPUs in the cloud. Most recent proposed concept in follows the federated learning paradigm, which was proposed by Google [14]. Federated Learning is aiming at train a high-quality centralized model while training data remains distributed over the edge, which have unreliable and relatively slow network connections.

3 Edge-Deployed Convolutional Neural Networks

In this section we give an overview of the proposed edge-deployed deep neural network (ECNN) architecture and describe how is the training and inference in ECNN performed.

3.1 ECNN Architecture

Basically, the ECNN is a distributed CNN, where the first several layer of the CNN, which composed with a series of convolution layers with filters (kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object. The edge cloud, and edge devices both own several layers of convolution layers and pooling layers. Meanwhile the local device could make the classification with the FC layers. Meanwhile, the cloud and edge also own their FC layers and the pooling layers.

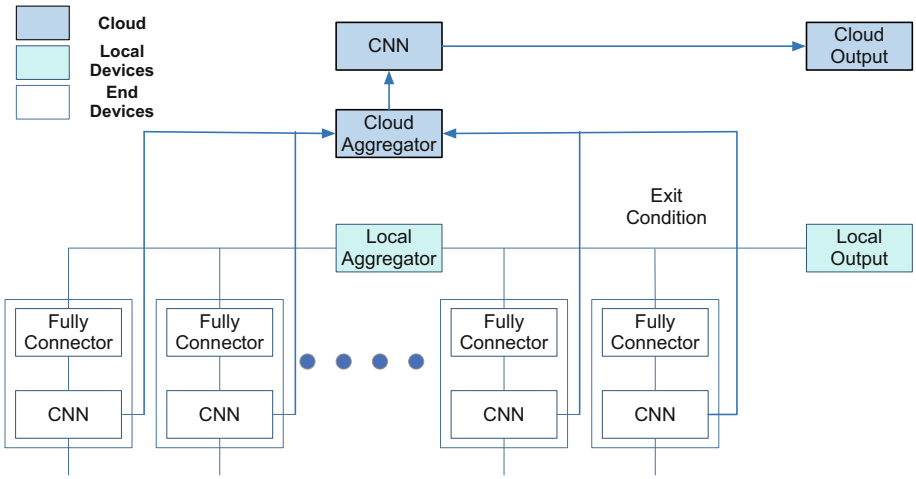

Fig. 2. ECNN structure for smart grid system

ECNN construct a CNN onto distributed smart-grid devices, edge-cloud, and the centralized cloud. Since ECNN relies on a combined CNN framework at all tiers in the neural network, the training and inference are greatly eased. Figure 2 is an overview of the ECNN architecture, which can be viewed as the standard CNN running in the edge and the cloud. In this case, sensor input captured on end devices will be sent to the cloud in form the features, the gradients or the original data.

This structure and the model could be performed in a single end device, by using first several layers of the CNN inference on the device. Then, using an exit condition, the local data could be used to perform local inference while the more

data-intensive inference in a broad area will require the edge cloud or the central could to perform. In this case, the intermediate CNN output, e.g. the features or even the gradients, is sent to the cloud, where further inference is performed using additional layers and a final classification. Note that the features can are much smaller than the sensor input, and therefore drastically reduce the network communication cost.

3.2 Data Aggregation in ECNN

The data aggregation is the essential feature of ECNN, which render our framework fitting into the distributed smart grids. This feature could be used to perform cross area data inference and decision making. Basically, in this subsection, we mainly answer following question, how could we efficiently use the ECNN to aggregate the output from the each end device with balanced computation and communication cost to perform classification? We mainly answer this question by proposing several different schemes for aggregation. We basically utilize different pooling skill in the method to aggregate different features, which are as follows:

- Max Pooling. This method mainly ensemble the input by taking the max of each row. Formally, max pooling can be expressed as

$$\hat{e}_{max} = \max_{1 \leq i,j \leq n} e_{ij}, \tag{1}$$

  where n is the number of inputs and e_{ij} is the element in i-th column and j-th row of the input matrix and $\hat{e}_{max}$ is the result of the max element.
- Average pooling (AP). AP aggregates the input vectors by taking the average of each component. This is written as

$$\hat{e}_{avg} = \sum_{i,j=1}^{n} \frac{e_{ij}}{n}, \tag{2}$$

  where n is the number of inputs and e_{ij} is the element in i-th column and j-th row of the input matrix and $\hat{e}_{max}$ is the result of the max element. Averaging may reduce noisy input presented in some end devices.

3.3 ECNN Training

Although our structure pose a well constructed structure for edge-assisted deep learning, how to train them with the distributed big data is still unsolved. Thus, in this section, we propose a primary method to solve this problem.

Basically, the ECNN system can be trained centralized in a powerful the cloud. But one question is that how to determine the multiple exit points as shown in Fig. 2. At training stage, the loss function and the gradients of each exit is combined so that the entire NN could be jointly trained, and each exit determine the accuracy relative to its depth. In this work, we inspired by the

work in [19,20] and proposed a federated learning [14] alike method. We now describe formally how we train ECNNs.

Let y be a label vector, x be a sample and $\mathcal{C}$ be the set of all possible labels. In every exit point, we design a specific softmax objective function which can be written as

$$L(\hat{y}, y; \theta) = ||\hat{y} - y||^2 \tag{3}$$

$$\hat{y} = \text{softmax}(z) = -\frac{\mathrm{e}^z}{\sum\limits_{c \in \mathcal{C}} \mathrm{e}_c^z}, \tag{4}$$

$$z = f_{\text{exit}_n}(x; \theta). \tag{5}$$

Here, f_{exit_n} is a function representing the computation of the neural network layers from an entry point to the n-th exit branch and θ represents the network parameters such as weights and biases of those layers.

Then the training could be performed wit the optimization problem as minimizing a weighted sum of the loss functions of each exit:

$$L(\hat{y}, y; \theta) = \sum_{n=1}^{N} \beta_n L(\hat{y}_{\text{exit}_n}, y; \theta)$$

where N is the total number of exit points and β_n is the associated weight of each exit. Usually, we define the weight of higher layer will be larger.

Note that each edge devices could jointly train the network with gradients exchange in the backward stage. The communication cost relies on the network size.

3.4 Inference of ECNN

Inference in ECNN is performed in several stages with exit thresholds T_i (where the T_i at each exit point i) which is a quantitative measure of how well is the prediction. Our basic idea is to construct the T_i by using a threshold as the confidence measure that determines whether to classify a sample at a exit point. This is enabled by searching the possible labeled set and while the max prediction value of the softmax is smaller the confidence level is also lower. The formal defined is as

$$\eta(\mathrm{x}) = max[softmax(c_i)], \forall c_i \in \mathcal{C} \tag{6}$$

where $\mathcal{C}$ is the set of all possible labels and c_i is the elements. Note that the softmax out put is the probability between 0 and 1. Thus, the η has values between 0 and 1, when η close to 1 means that the ECNN is confident about the prediction; Meanwhile, η close to 0 implies not suitable. At each exit point, η is compared against T_i in order to determine if the sample should exit at that point.

At a given exit point, if the predictor is not confident in the result (i.e., $\eta > T$), the inference task will be transferred to the higher layer along with the features extracted in the edge until the cloud layer.

4 Analysis

In this section, we evaluate our method both theoretically and experimentally. In the theoretical part, we mainly analyze the most important part of the communication cost of ECNN inference. While the experiment part evaluate our method in terms of the prediction performance.

4.1 Communication Cost of ECNN

The total communication cost for an end device with the edge cloud and central cloud is formalized as follows

$$c = s \times z \times \mathcal{L} + (1 - s)f \times t, \tag{7}$$

where s is the portion of samples exited locally, $\mathcal{L}$ is number of possible labels, f is the number of pooling and FC filters, and t is the output size of a single filter for the final NN layer on the end-device. The constant z corresponds to the size of the data represent the feature extracted. The first term implies the probability that the sample to be transmitted from the end device to the edge cloud belongs.

The second term is the communication cost between edge cloud and the central cloud.

4.2 Numerical Results

We evaluate the combined ECNN with recognition accuracy and the training cost in trained environment with the samples we have collected. The data set using to running our framework is the CSI data collected using to identify the behavior and activity consist with 3 data sets, named fixed, semi, open. Such network are using to recognize 8 kinds of activities. We just simplify such task into 8 kind classification task (Fig. 3).

4.3 Accuracy Performance

We take the 80% of samples in each class as the training set, the rest as the test set. For the training set, we use 10-fold cross validation. From Fig. 4, we get that the average accuracy of activity model is 89.14%.

4.4 Training Cost

As is illustrated in Fig. 4, our model converge in different data set no greater than 10000 round of iteration. In data set fixed our model perform the best with only 3000 iteration to converge and the Loss is smaller than 0.6.

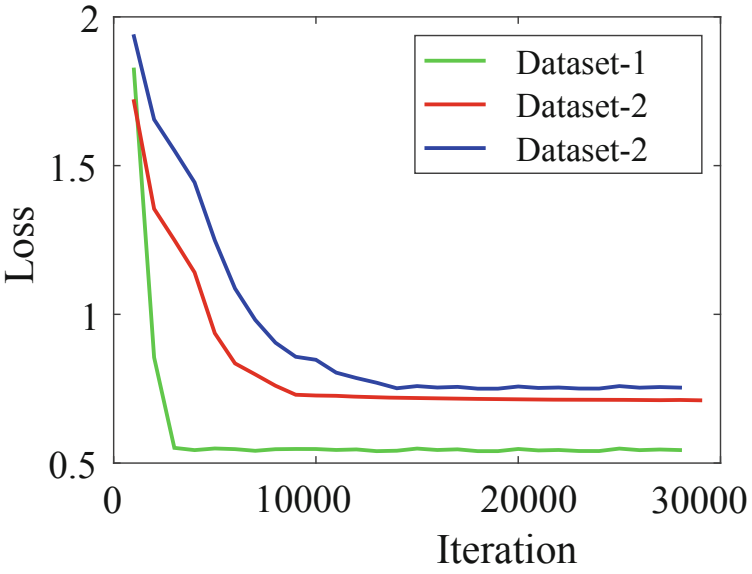

Fig. 3. The loss curve of ECNN

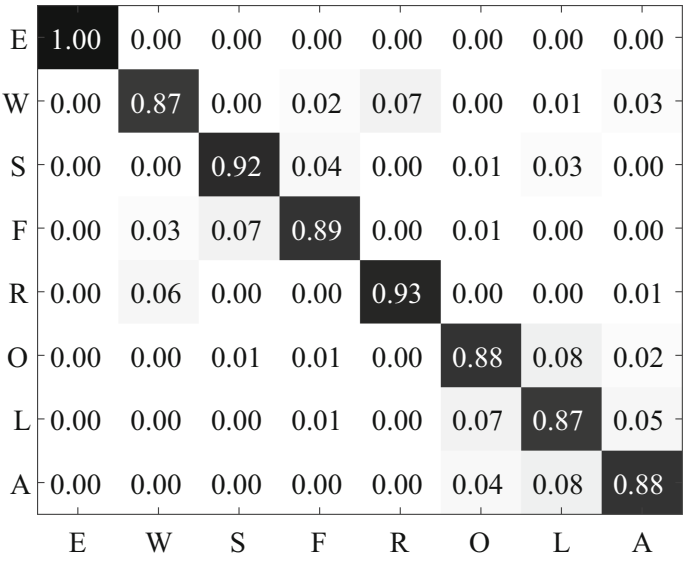

Fig. 4. The recognition accuracy of our method

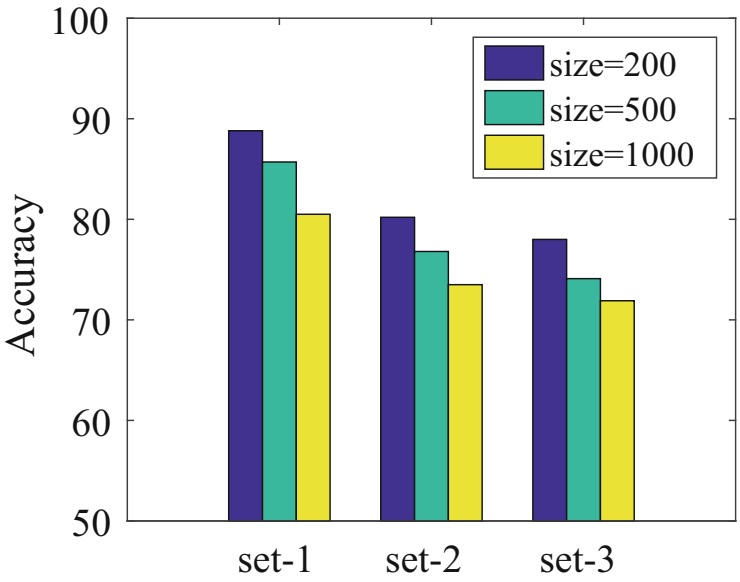

Fig. 5. The accuracy vs. the size of data set

4.5 Size of Data

According to the result illustrated in the Fig. 4 in different data set, we use different data size to train the model then use 10 folds to examine the accuracy. We find that, with only 1000 samples our model could achieve almost 90% accuracy. Plus, with no surprise, less data size lead to low prediction accuracy (Fig. 5).

5 Conclusions

In this paper we design and propose ECNN (edge-deployed Convolution Neural Network) in edge computing assisted smart grids to greatly enhance the ability in data aggregation and analytics. We also discuss how to train such network in edge computing distributively. Our method owning the advantage in communication cost and the prediction accuracy which fully utilized the distribution nature of the smart grid and the data compression and feature extraction ability of the CNN. We expect that our framework could greatly enhance the "smart" feature of the smart grids by providing low cost and high accuracy data analytics ability.

Acknowledgment. This work was supported by the State Grid Corporation Science and Technology Project (Contract No.: SG2NK00DWJS1800123).

References

1. Yi, S., Hao, Z., Qin, Z., Li, Q.: Fog computing: platform and applications. In: Hot Topics in Web Systems and Technologies, pp. 73–78 (2016)

2. Chang, X., Wang, J., Wang, J., Lu, K., Zhuang, Y.: On the optimal design of secure network coding against wiretapping attack. Comput. Netw. **99**(C), 82–98 (2016)
3. Chang, X., et al.: Accuracy-aware interference modeling and measurement in wireless sensor networks. IEEE Trans. Mob. Comput. **15**(2), 278–291 (2016)
4. Wan, M., Zhao, J., Yuan, W., Zeng, P., Cui, J., Shang, W.: Intrusion detection of industrial control based on semi-supervised clustering strategy. Inf. Control **46**(4), 462–468 (2017)
5. Skala, K., Davidovic, D., Afgan, E., SoviC, I.: Scalable distributed computing hierarchy: cloud, fog and dew computing. Open J. Cloud Comput. **2**(1), 16–24 (2015)
6. Song, C., Jing, W., Zeng, P., Rosenberg, C.: An analysis on the energy consumption of circulating pumps of residential swimming pools for peak load management. Appl. Energy **195**, 1–12 (2017)
7. Song, C., Wei, J., Peng, Z., Haibin, Y., Rosenberg, C.: Energy consumption analysis of residential swimming pools for peak load shaving. Appl. Energy **220**, 176–191 (2018)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
9. Song, C., Zeng, P., Wang, Z., Zhao, H., Yu, H.: Wearable continuous body temperature measurement using multiple artificial neural networks. IEEE Trans. Industr. Inf. **14**(10), 4395–4406 (2018)
10. Wang, J., Meng, R., Rice, S.G., Sun, X.: A fusion steganographic algorithm based on faster R-CNN. CMC Comput. Mater. Continua **55**(1), 001–016 (2018)
11. Li, F., Sherratt, R.S., Zeng, D., Dai, Y., Wang, J.: Adversarial learning for distant supervised relation extraction. CMC Comput. Mater. Continua **55**(1), 121–136 (2018)
12. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: imagenet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
13. Mcdanel, B., Teerapittayanon, S., Kung, H.T.: Embedded binarized neural networks (2017)
14. KoneČný, J., McMahan, H.B., Yu, F.X., Richtrik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency (2016)
15. Dean, J., et al.: Large scale distributed deep networks. In: International Conference on Neural Information Processing Systems, pp. 1223–1231 (2012)
16. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Keutzer, K.: Firecaffe: near-linear acceleration of deep neural network training on compute clusters, vol. 37, pp. 2592–2600 (2015)
17. Dean, J.: Large scale deep learning (2014)
18. Teerapittayanon, S., McDanel, B., Kung, H.T.: Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 328–339. IEEE (2017)
19. Szegedy, C., et al.: Going deeper with convolutions, pp. 1–9 (2014)
20. Teerapittayanon, S., Mcdanel, B., Kung, H.T.: BranchyNet: fast inference via early exiting from deep neural networks, pp. 2464–2469 (2017)