Edge Computing Based Traffic Analysis System Using Broad Learning

Xiting Peng, Kaoru Ota, and Mianxiong Dong(✉)

Department of Information and Electronic Engineering,
Muroran Institute of Technology, Muroran, Japan
{17096505,ota,mxdong}@mmm.muroran-it.ac.jp

Abstract. Current traffic analysis methods are executed on the cloud, which need to upload the traffic data and consume precious bandwidth resources. Edge computing is a more promising way to save the bandwidth resources and improve users' privacy by offloading these tasks to the edge node. However, traffic analysis methods based on traditional machine learning need to retrain all traffic data when updating the trained model, which are not suitable for edge computing due to the poor computing power and low storage capacity of edge nodes. In this paper, we propose a novel edge computing based traffic analysis system using broad learning. For one thing, edge computing can provide a distributed architecture for saving the bandwidth resources and protecting users' privacy. For another, we use broad learning to incrementally train the traffic data, which is more suitable for edge computing because it can support incremental updates of models on the edge nodes without retraining all data. We implement our system on the Raspberry Pi, and experimental results show that we have 98% probability to accurately identify these traffic data. Moreover, our method has the faster training speed compared with Convolutional Neural Network (CNN).

Keywords: Traffic analysis · Edge computing · Broad learning system

1 Introduction

With the rapid development of Internet services, traffic types become diverse. In the complex network environment, traffic analysis is regarded as an important approach to ensure the security of network. It can effectively deal with many security issues, such as intrusion detection [3]. Moreover, network traffic analysis also plays an important role in modern network management systems, such as quality of service (QoS) [26]. Accurate identification of network traffic types is the basis for providing better services. By executing accurate traffic analysis, service providers can monitor the operation of the entire network. By analyzing users' traffic, service providers can realize their behaviors, which can provide the personalized services. Therefore, how to provide accurate traffic classification method is crucial.

S. Han et al. (Eds.): AICON 2019, LNICST 287, pp. 238–251, 2019.
https://doi.org/10.1007/978-3-030-22971-9_20

Current classification methods for traffic types are executed on the cloud, which need to upload all users' traffic to the cloud [19]. However, many applications, such as video applications, can generate large amounts of traffic in a short amount of time, and it will consume a lot of precious bandwidth resources if all traffic data is uploaded to the server [5]. Therefore, current cloud-based traffic analysis system does not apply to traffic-intensive applications. Moreover, service providers only need the classification results for monitoring the network operation and providing better services, rather than all traffic data. Therefore, it is better to propose a novel distributed traffic analysis architecture, which only needs to upload the classification results to the cloud.

Edge computing is a promising way to achieve the distributed traffic analysis architecture by offloading this task to the edge node close to user devices [2,9,17]. We can deploy the traffic classification models on the edge node, such as WiFi access points. In one hand, this method only needs to send the classification results to the cloud, rather than uploading all traffic data, which can analyze users' traffic in real time and save the precious bandwidth resources. In the other hand, users' traffic may contain many private information, such as location, gender, or education background, which can be inferred by the malicious service providers [22]. Compared with the cloud based traffic analysis architecture, edge computing can better protect the privacy of users.

Current traffic classification models are mainly based on traditional machine learning, including SVM, random forest and decision tree. However, these traditional methods cannot be directly used at edge computing based traffic analysis architecture due to the low computing and storage capabilities of edge nodes. In order to maintain the classification accuracy, it is better to regularly update the trained model. Traditional methods require retraining all data when performing model updates, which consume large amounts of computing power. Usually, the edge node has these characteristics: poor computing power and low storage capacity [20], which cannot support frequent retraining of all data. To address the above challenges, we use a novel and lightweight neural network structure, broad learning system (BLS) [4], which has the faster training speed due to its flat network structure. More importantly, BLS can use incremental learning to constantly update the trained model when new data enters, and no retraining process is needed.

In this paper, we propose a novel edge computing based traffic analysis system using broad learning. The proposed system is composed of two major modules: basic training on the cloud and incremental training (model updating) on the edge node. Firstly, we use some traffic data to train a basic model and send the basic model to the edge node. Secondly, when the accuracy of the trained model is not enough to provide better services, model updating will be executed on the edge node by the incremental way. We implement the edge computing based traffic analysis system on the Raspberry Pi, and the experimental results show that our method has the faster training speed compared with Convolutional Neural Network (CNN).

The main contributions of this paper are summarized as follows:

- We propose a distributed traffic analysis architecture, edge computing based traffic analysis system, which can save the bandwidth resources and protect users' privacy from being inferred by malicious service providers.
- In order to solve the problem of poor computing power and low storage capacities of edge nodes, we use a novel and lightweight neural network structure (broad learning system) to analyze traffic data, which has fast training speed and can support incremental learning.
- We implement our system on the Raspberry Pi and perform comprehensive experiments using real-world dataset to validate its performance.

The rest of this paper is organized as follows. Related work about the traffic analysis and the edge computing is introduced in Sect. 2. We briefly introduce the broad learning system in Sect. 3. Section 4 presents the proposed edge computing based traffic analysis system. We discuss the experimental evaluation in Sect. 5. We conclude this paper in Sect. 6.

2 Related Work

In this section, we will introduce the existing works with respect to the traffic analysis and explain their disadvantages under the environment with diverse traffic-intensive applications in detail.

2.1 Traffic Analysis

Traffic analysis plays an important role in providing better services by service providers. Nowadays, traffic analysis has received wide attention from both the academia and the industry [10,15]. Researchers have proposed many methods for traffic classification, which can be classified as follows: port-based identification, deep packet inspection, and machine learning. With the rapid development of Internet services, the first two technologies have strong limitations, which has the low accuracy when identifying varied traffic types. Therefore, machine learning based methods have been widely used by the academia and the industry.

Machine learning based traffic classification methods [14] have a wide range of applications in different fields. In [11], the authors proposed a traffic classification method based on semi-supervised support vector machine (S-SVM) to accurately identify a variety of network traffic. In [24], random forest was used to identify smartphone apps by fingerprinting the network traffic. Besides, many machine learning based methods are available for traffic classification. For instance, in [1], the authors used six common algorithms, such as Linear Regression, Decision Tree, and Multi-layer Perceptron, for malware traffic classification and compared their performances when confronted with real network data.

However, these methods are executed on the cloud, which need to upload all traffic to the server and consume large amounts of precious bandwidth resources.

In addition to wasting resources, uploading traffic to the cloud will also cause the leakage of users' privacy. In [7], the authors pointed out that users' privacy can be inferred by executing traffic analysis even in the case of encryption. Therefore, in this paper, we propose a novel distributed architecture with the help of edge computing to ensure that traffic data is analyzed at the edge node which is trustworthy to users. Furthermore, edge node always has the poor computing power and low storage capacity, which cannot directly deploy traditional machine learning based methods because these methods need to retrain all data when model updating. Deep learning based traffic analysis methods [23] can save and update the trained model in an incremental way. However, due to the complexity of the deep structure, it costs more time. Therefore, we use a lightweight neural network structure, broad learning system, to train the traffic data on the edge node in this paper.

2.2 Edge Computing

As users have higher requirements for lower latency, lower bandwidth consumption, and higher security and privacy, cloud services cannot meet their requirements. Therefore, edge computing is proposed as a promising way to assist cloud to achieve these requirements [13,20,25]. Generally speaking, edge computing means offloading some tasks that are previously carried out on the cloud to the devices that are close to the user side [21]. In some classic edge computing scenarios, such as Vehicular Network, Smart Home, Body Things, and other IoT scenarios, the edge nodes refer to the Road Side Unit (RSU), Hub (Gateway), smartphone, and WiFi access point, which always have little computing and storage capacities. By combining edge computing and cloud computing, service providers not only can meet the requirements of large-scale data processing, but also meet the needs of real-time [12].

The classic edge computing based architecture is comprised of the following three layers [8]: Cloud layer, Edge layer, and User layer. In this architecture, the users send the data to the edge node, rather than the cloud. And the edge node performs data processing tasks, which can provide the real-time services and reduce the burden of the server. After completing the data processing, the edge node sends the results to the cloud.

3 Preliminary Knowledge

In order to meet the characteristics of the poor computing power and low storage capacity of the edge node, we utilize broad learning system [4], a novel and lightweight neural network structure, which has fast training speed and does not need to retrain all data when regularly updating the trained model.

Broad learning system (BLS) is developed with fewer parameters and the training speed can be faster due to its flat network structure, which is composed of input layer, feature nodes/enhancement nodes, and output layer. The standard structure of BLS is shown in Fig. 1: Firstly, the mapped features are

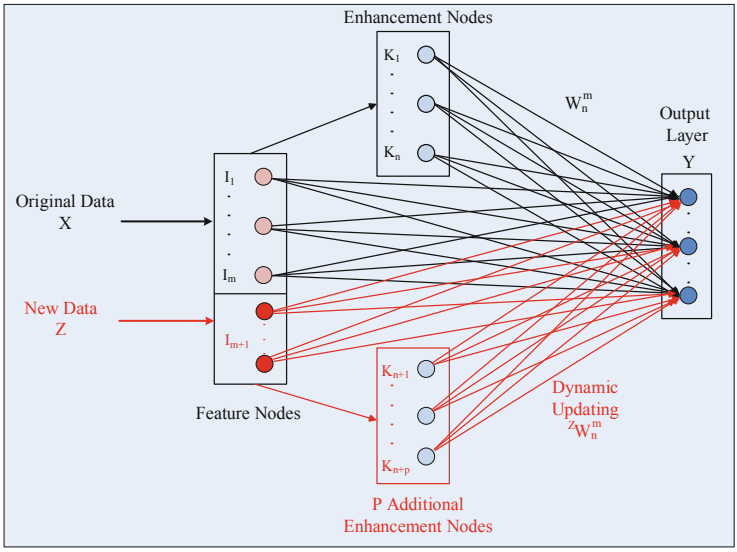


Fig. 1. The architecture of the broad learning system

extracted from the original data to generate the feature nodes. Then through the activation functions which can be either nonlinear or linear, the feature nodes are enhanced into enhancement nodes. The connections of all the feature nodes and the enhancement nodes are used to generate the output through the pseudoinverse.

We set I_X^m and K_X^n as the m groups of feature mapping nodes and n groups of enhancement nodes of the initial network respectively, where X represents the initial input. These nodes can be generated by the following way:

$$I_X^m = \left[\varphi(XA_{e1} + \theta_{e1}), ..., \varphi(XA_{em} + \theta_{em})\right] \tag{1}$$

$$K_X^n = \left[\chi(I_X^m A_{h1} + \theta_{h1}), ..., \chi(I_X^m A_{hn} + \theta_{hn})\right] \tag{2}$$

where $A_{ei}, \theta_{ei}, \varphi$ are the random weights, bias and activation function of feature nodes, and $A_{hi}, \theta_{hi}, \chi$ are corresponding parameters of enhancement nodes. According to these generated nodes, we can compute the weights $W_n^m = (I_X^m | K_X^n)^+ Y$, where $(I_X^m | K_X^n)^+$ is the pseudoinverse of the matrix $(I_X^m | K_X^n)$ and Y is the label of the input data.

In traffic analysis scenarios, we need to update traffic classification model frequently. BLS with incremental learning can meet this requirement in a resource-saving way. Introducing incremental learning into BLS is a good option which can avoid retaining all traffic data. When new traffic data comes into the trained model, the output-layer weights of the network can be updated without retraining the network model. What's more, how to update the network model is illustrated as follows.

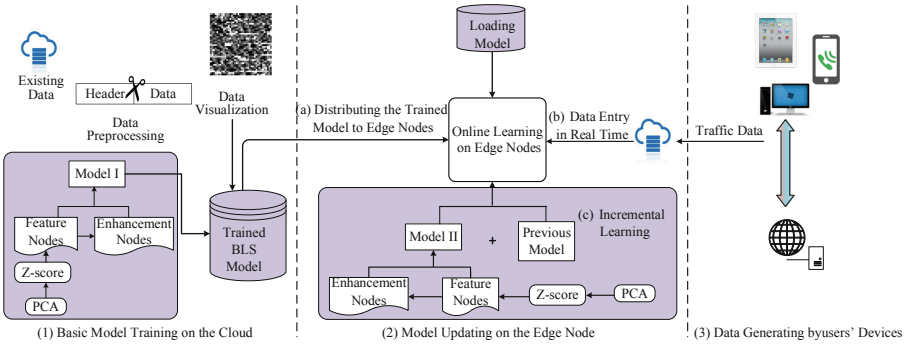


Fig. 2. Edge computing based traffic analysis system

When new data Z inputs, the corresponding feature nodes I_Z^m and enhancement nodes K_Z^n can be calculated using the same way as the Eqs. (1) and (2). The new weights of the BLS model can be updated by the dynamic stepwise updating algorithm based on the previous weights W_n^m:

$$^{Z}W_n^m = \left[W_n^m + B(Y_z - (I_Z^m|K_Z^n)W_n^m)\right] \tag{3}$$

where Y_z is the label of the new data Z. Note that,

$$B = \begin{cases} (C^T)^+, & \text{if } C \neq 0 \\ (I_X^m|K_X^n)^+ D(I + D^T D)^{-1}, & \text{if } C = 0 \end{cases} \tag{4}$$

where $C^T = (I_Z^m|K_Z^n) - D^T(I_X^m|K_X^n)$, and $D^T = (I_Z^m|K_Z^n)(I_X^m|K_X^n)^+$.

4 System Design and Implementation

In this section, we will introduce the proposed edge computing based traffic analysis system from the perspective of its architecture and the concrete training process.

4.1 System Design

To thwart the problems that cannot be solved by the cloud computing (saving bandwidth resources) and protect users' data privacy, we propose a distributed traffic analysis system with the help of edge computing and use a novel and lightweight network structure to train the traffic data, which can better fit the edge computing based architecture. As shown in Fig. 2, the proposed edge computing based traffic analysis system is mainly composed of two stages: basic classification model training on the cloud and the model updating on the edge node, which are designed to reduce the bandwidth resources and protect users' privacy from being inferred by malicious service providers without affecting the classification results.

Algorithm 1. Basic Model Training

Input: Traffic Data X, Label $Y_{a,c}$, maptimes t_1, enhencetimes t_2, Batchsize g
Output: Trained Basic Model $\mathcal{M}$

$X^{'} \leftarrow$ PCA (X); $X^{''}_{a,b} \leftarrow$ Z-score $(X^{'})$;
// Feature Nodes I
for *each* $i \in [1, t_1]$ **do**
  $A_{b,g}, \theta \leftarrow$ Random ();
  $I^{i}_{a,g} \leftarrow \varphi(X^{''}_{a,b} A_{b,g} + \theta)$; //$\varphi$ is activation function
  $I_{a,m} \leftarrow$ Add $I^{i}_{a,g}$; // $m = t_1 * g$
end
// Enhancement Nodes K
for *each* $i \in [1, t_2]$ **do**
  $A^{'}_{m,g}, \theta^{'} \leftarrow$ Random ();
  $K^{i}_{a,g} \leftarrow \chi(I_{a,m} A^{'}_{m,g} + \theta^{'})$; //$\chi$ is activation function
  $K_{a,n} \leftarrow$ Add $K^{i}_{a,g}$; // $n = t_2 * g$
end
// Updating weight W
$P_{m+n,a} \leftarrow$ Pseuedoinverse $(I_{a,m}, K_{a,n})$;
$W_{m+n,c} \leftarrow P_{m+n,a} Y_{a,c}$;
$\mathcal{M} \leftarrow$ Save the Model ($W_{m+n,c}$, $P_{m+n,a}$, $I_{a,m}$, $K_{a,n}$)
return $\mathcal{M}$

Basic Model Training on the Cloud. Considering that the edge node usually has poor computing power and low storage capacity, it is impossible to process large amounts of traffic data for basic model training. Therefore, we use the cloud to perform this task, which is the same as the current architecture. After the basic model training, the cloud sends the basic model to the edge node, and the edge node updates the trained model using the incremental learning, which does not need to interact with the cloud.

In order to make broad learning system more suitable for the proposed architecture, we modify the basic broad learning algorithm. In our system, the cloud performs the following two steps as shown in Algorithm 1: (1) Analyzing the existing traffic data and generating the basic model. Our approach is not to completely abandon the previous architecture, but to further optimize on the basis of the existing architecture. Since the cloud has collected many traffic data, we can reuse these data for basic model training instead of recollecting the data, which can further save the bandwidth resources. Firstly, we use data reduction method, principal component analysis (PCA), to process these traffic data and extract the important feature data (input data). Then feature nodes and enhancement nodes can be generated based on the input data, random weights and bias, and the activation function. By computing the pseudoinverse of the feature nodes and enhancement nodes, we can obtain the final weights, which constitute the trained model. (2) Saving the trained model and distributing it to the edge nodes. In order to obtain the trained model which has a certain precision, the cloud can set the checkpoint. When the training accuracy reaches a

threshold (the checkpoint), the training process is over. Then this model will be distributed to the edge nodes. Note that, the training process on the cloud only needs to be done once in an offline way when the traffic types do not change. If the traffic types increase, the basic model needs to be retrained on the cloud. However, the traffic types will not change for a long time and the retraining process can be completed in an offline way, which has a negligible impact on the system's performance.

Algorithm 2. Incremental Learning on the Edge Node

Input: Basic Model $\mathcal{M}$, New Traffic Z, Label $Y'_{a',c}$
Output: The Updated Model $\mathcal{M}'$

$W_{m+n,c}$, $P_{m+n,a}$, $I_{a,m}$, $K_{a,n} \leftarrow$ Load Basic Model $\mathcal{M}$
$Z' \leftarrow$ PCA (Z); $Z''_{a',b} \leftarrow$ Z-score (Z');
// Feature Nodes I'
$I'_{a',m} \leftarrow$ Use the same method as Algorithm 1
// Enhancement Nodes K'
$K'_{a',n} \leftarrow$ Use the same method as Algorithm 1
// Calculate $B_{m+n,a'}$
$D^T_{a',a} \leftarrow (I'_{a',m}|K'_{a',n})P_{m+n,a}$;
$C^T_{a',m+n} \leftarrow (I'_{a',m}|K'_{a',n}) - D^T_{a',a}(I_{a,m}|K_{a,n})$;
if $C_{m+n,a'}$ *!= 0* **then**
  $B_{m+n,a'} \leftarrow$ Pseuedoinverse $(C^T_{a',m+n})$;
end
else
  $B_{m+n,a'} \leftarrow P_{m+n,a}D_{a,a'}((D^T_{a',a}D_{a,a'} + diag(a')).I)$
end
// Updating weight W
$W_{m+n,c} \leftarrow W_{m+n,c} + B_{m+n,a'}(Y'_{a',c} - (I'_{a',m}|K'_{a',n})W_{m+n,c})$;
$P_{m+n,a+a'} \leftarrow ((P_{m+n,a} - B_{m+n,a'}D^T_{a',a})|B_{m+n,a'})$;
return $\mathcal{M}'$

Model Updating on the Edge Node. To save the precious bandwidth resources and protect users' data privacy, we change the current system and adopt the edge computing based traffic analysis system. In our system, the edge nodes refer to the devices between the user side and the cloud, which are close to the users' device and can be controlled by the users instead of the service providers. In our scenario, the common edge nodes refer to the WiFi access point, router, hub, and switch devices. Usually, the edge nodes have poor computing power and lower storage capacity. Therefore, it is reasonable to deploy the lightweight machine learning on the edge node.

After receiving the trained model from the cloud, the edge node loads this model and continues to train the model by the way of incremental learning. The concrete algorithm is shown in Algorithm 2. The edge node does not need to process previous data, and it only needs to generate additional feature nodes

and enhancement nodes for the new input data. Data preprocessing is completed by principal component analysis and the generated method of the additional feature nodes and enhancement nodes is the same as the Algorithm 1. These newly generated nodes will be used to update the BLS model by the dynamic stepwise updating algorithm. Through incremental learning, we can constantly update the model and maintain the accuracy of the trained model.

Moreover, we use a specific scenario to show the operational process: The network operator wants to block some types of traffic (e.g., video traffic) going through their networks. Firstly, they construct the classification model based on the broad learning algorithm on the server (Step 1: Basic model training on the cloud). Then this model will be distributed to the edge nodes (such as gateway in this scenario), which can be used to continue training on the edge node based on the real-time incoming data (Step 2: Model updating on the edge node). This model can be deployed on the edge nodes, and special types of data will be blocked when they go through these edge nodes based on the classification results.

4.2 Principal Component Analysis for Data Reduction

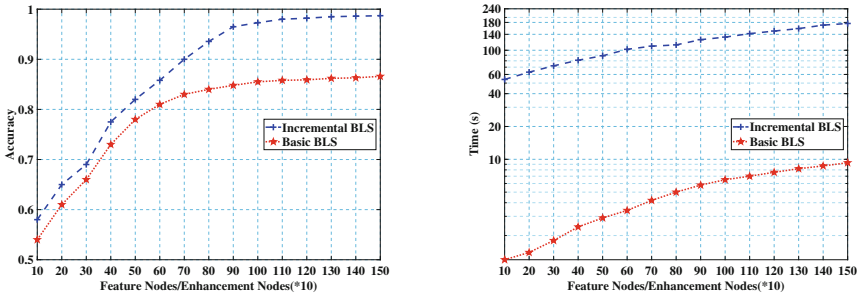
Although broad learning system can efficiently reduce the training time due to its simple network structure, it is difficult to directly deploy the broad learning algorithm on the devices with the limited performance. Fortunately, it can be realized with the following step: Since the input data always has higher dimension, it will inevitably take up a lot of training time if we directly input high-dimensional data into the training algorithm. Therefore, instead of inputting all traffic data into the training algorithm, we use data reduction method to process these traffic data and extract the important features. Principal component analysis is a common method for data reduction. It uses an orthogonal transformation to convert the high-dimensional data, which can convert the correlated variables into a set of linearly uncorrelated variables. We can set the suitable parameters for principal component analysis, which will have a negligible influence to the classification accuracy.

5 Evaluation

To evaluate the performance of the edge computing based traffic analysis system, we implement our system on the classic devices, Raspberry Pi, and compare its performance with other machine learning based method. In this paper, we use a similar metric to other works on traffic analysis to evaluate the performance of our system, which selects several classic traffic types [11]. We use massive data packets which mainly contain the following traffic: Twitter, Youtube, Gmail, Http, Instagram, Samba, and so on. The traffic data used in this paper is collected by Wireshark, a classic network packet capture, which is saved as the packet capture (pcap) file and can be analyzed by the standard libpcap library. The pcap file is consist of the pcap header, a series of the packet header and the

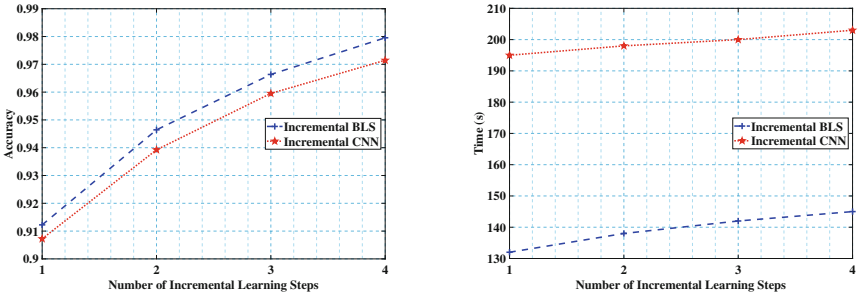
corresponding packet data. Especially, the packet header contains some metadata about the packet, including the packet length, timestamp, and a 32-bit link layer type field, which needs to be removed when preprocessing data [10]. Note that we only use the packet data for traffic classification.

In our experiment, we use 40% of traffic data for training the basic model on the cloud. After the basic training, the trained model will be distributed to the edge node for further training. Moreover, 40% of traffic data is used for incremental learning on the edge node. In the process of incremental training, we will assign these data to four training steps to evaluate the effectiveness of the incremental learning. The model will be saved when each training step is completed. Finally, the remaining 20% of the data is the test set.



(a) Classification Accuracy (b) Training Time

Fig. 3. Accuracy and time under different number of nodes



(a) Classification Accuracy (b) Training Time

Fig. 4. Accuracy and efficiency comparison with CNN

5.1 Prototype

For evaluating the effectiveness/efficiency of our proposed scheme, we have deployed our scheme on the Raspberry Pi 3 Model B with a 64-bit quad-core ARM-v8 CPU at 1.2 GHz and 1 GB of ARM memory. As a classic device, Raspberry Pi has the similar computing and storage capabilities with the edge node

that is defined in the paper [6,16,18]. Therefore, it is reasonable to evaluate the proposed edge computing based traffic analysis system using this device.

The performance metrics in the experiments include training accuracy and training time, which can be affected by the number of feature/enhancement nodes. In our experiment, we evaluate the effect of the number of feature nodes and the enhancement nodes by varying their numbers from 100 to 1500. For comparison with other methods, we fix the number of feature nodes and the enhancement nodes at 1000. The activation function is the commonly used functions in machine learning, including sigmoid, tanh, relu, and orth.

5.2 Classification Performance

We deploy the proposed broad learning system model on the edge nodes, Raspberry Pi 3 Model B, to evaluate its performance. Firstly, we will use the classification accuracy as the evaluation parameter. The number of the feature nodes and enhancement nodes are the important factors affecting the classification accuracy, which can be adjusted for obtaining accurate results. Note that, the training time will grow with the increase of the number of these nodes. Therefore, we should find a tradeoff between the training time and the accuracy by choosing suitable number of nodes. Figure 3(a) shows the classification accuracy of the basic model and the incremental model under different number of nodes. From this figure, we can see that the accuracy of the trained model is significantly improved on the basis of the basic model by incremental training of new data.

In our experiment, we evaluate the effect of the number of feature nodes and the enhancement nodes by varying their numbers from 100 to 1500. When we set the value of both the feature nodes and the enhancement nodes to 1000, the classification accuracy of the basic model and the incremental model are 85.5% and 98%. We can see that with the increase of the number of nodes, the classification accuracy of the basic model and the incremental model will significantly grow when these nodes are small. However, in the later stage, even if the nodes continue to increase, the grow of the classification accuracy is limited. That is, when the classification accuracy reaches a threshold, the increase of the nodes does not work. Moreover, the increase of the nodes means the training time will grow, as shown in Fig. 3(b). As shown in this figure, although basic model training takes a short time, the training time of the incremental learning will significantly increase with the grow of the number of feature nodes/the enhancement nodes. Therefore, we need set a reasonable parameter (such as 1000 or 1100) instead of increasing these nodes as many as possible.

5.3 Performance Comparison with Deep Learning

We compare the broad learning system based traffic classification model with the deep learning based model, which is the most popular machine learning method. Note that, in this paper we use the classic deep structure, Convolutional Neural Network, which includes multiple convolutional layers and full-connected

layers. Like the Broad learning system, CNN can also save the trained model and continue to train the model by the way of incremental learning. Therefore, we can use the same dataset and set the same parameters as the broad learning system, which can provide a consistent environment for performance comparison.

In our setting, we construct a CNN model which contains four convolutional layers and two full-connected layers. Data pre-processing uses the same steps as the broad learning system, including the data visualization (38×38). When the accuracy of the basic CNN model is consistent with the basic model of the broad learning system, the basic CNN model is distributed to the same Raspberry Pi for incremental learning. The training results are shown in Fig. 4(a), where $i = 1, 2, 3, 4$ represents each incremental stage. This figure shows that these two models have the similar training accuracy in each incremental stage. Then we evaluate the training time of each incremental stage, and the results are shown in Fig. 4(b). From this figure we can see that our scheme only needs about 70% of the training time of the CNN model, which can show the superiority of the proposed model.

6 Conclusion

We propose a novel edge computing based traffic analysis system using broad learning. In one hand, edge computing can provide a distributed architecture, which can save the precious bandwidth resources and provide the safer service environment. In the other hand, we use broad learning system to incrementally train the traffic data, which is more suitable for the edge computing because it can support incremental updates of models on the edge nodes without retraining all data. We implement the edge computing based traffic analysis system on the Raspberry Pi, and the experimental results show that our method has the faster training speed compared with other neural network architecture.

Acknowledgments. This work is partially supported by JSPS KAKENHI Grant Number JP16K00117, JP19K20250, KDDI Foundation and the China Scholarship Council (201808050016). Mianxiong Dong is the corresponding author.

References

1. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD 2017, pp. 1723–1732. ACM, New York (2017). https://doi.org/10.1145/3097983.3098163
2. Baktir, A.C., Ozgovde, A., Ersoy, C.: How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. IEEE Commun. Surv. Tutorials **19**(4), 2359–2391 (2017). https://doi.org/10.1109/COMST.2017.2717482

3. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutorials **18**(2), 1153–1176 (2016). https://doi.org/10.1109/COMST.2015.2494502
4. Chen, C.L.P., Liu, Z.: Broad learning system: an effective and efficient incremental learning system without the need for deep architecture. IEEE Trans. Neural Networks Learn. Syst. **29**(1), 10–24 (2018). https://doi.org/10.1109/TNNLS.2017.2716952
5. Han, S., Xu, S., Meng, W., Li, C.: Dense-device-enabled cooperative networks for efficient and secure transmission. IEEE Network **32**(2), 100–106 (2018). https://doi.org/10.1109/MNET.2018.1700292
6. Jalali, F., Hinton, K., Ayre, R., Alpcan, T., Tucker, R.S.: Fog computing may help to save energy in cloud computing. IEEE J. Sel. Areas Commun. **34**(5), 1728–1739 (2016). https://doi.org/10.1109/JSAC.2016.2545559
7. Korczyński, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pp. 781–789, April 2014. https://doi.org/10.1109/INFOCOM.2014.6848005
8. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the internet of things with edge computing. IEEE Network **32**(1), 96–101 (2018). https://doi.org/10.1109/MNET.2018.1700202
9. Li, H., Ota, K., Dong, M.: ECCN: orchestration of edge-centric computing and content-centric networking in the 5G radio access network. IEEE Wirel. Commun. **25**(3), 88–93 (2018). https://doi.org/10.1109/MWC.2018.1700315
10. Li, L., Ota, K., Dong, M.: DeepNFV: a light-weight framework for intelligent edge network functions virtualization. IEEE Network **99**, 1–6 (2018). https://doi.org/10.1109/MNET.2018.1700394
11. Li, X., Qi, F., Xu, D., Qiu, X.: An internet traffic classification method based on semi-supervised support vector machine. In: 2011 IEEE International Conference on Communications (ICC), pp. 1–5, June 2011. https://doi.org/10.1109/icc.2011.5962736
12. Liu, J., Wan, J., Zeng, B., Wang, Q., Song, H., Qiu, M.: A scalable and quick-response software defined vehicular network assisted by mobile edge computing. IEEE Commun. Mag. **55**(7), 94–100 (2017). https://doi.org/10.1109/MCOM.2017.1601150
13. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutorials **19**(4), 2322–2358 (2017). https://doi.org/10.1109/COMST.2017.2745201
14. Meidan, Y., et al.: Profiliot: a machine learning approach for IoT device identification based on network traffic analysis. In: Proceedings of the Symposium on Applied Computing SAC 2017, pp. 506–509. ACM, New York (2017). https://doi.org/10.1145/3019612.3019878
15. Naboulsi, D., Fiore, M., Ribot, S., Stanica, R.: Large-scale mobile traffic analysis: a survey. IEEE Commun. Surv. Tutorials **18**(1), 124–161 (2016). https://doi.org/10.1109/COMST.2015.2491361
16. Patel, P., Intizar Ali, M., Sheth, A.: On using the intelligent edge for IoT analytics. IEEE Intell. Syst. **32**(5), 64–69 (2017). https://doi.org/10.1109/MIS.2017.3711653
17. Peng, M., Yan, S., Zhang, K., Wang, C.: Fog-computing-based radio access networks: issues and challenges. IEEE Network **30**(4), 46–53 (2016). https://doi.org/10.1109/MNET.2016.7513863
18. Roman, R., Lopez, J., Mambo, M.: Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. Future Gener. Comput. Syst. **78**, 680–698 (2018). https://doi.org/10.1016/j.future.2016.11.009

19. Sankari, S., Varalakshmi, P., Divya, B.: Network traffic analysis of cloud data centre. In: 2015 International Conference on Computing and Communications Technologies (ICCCT), pp. 408–413, February 2015. https://doi.org/10.1109/ICCCT2.2015.7292785
20. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016). https://doi.org/10.1109/JIOT.2016.2579198
21. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016). https://doi.org/10.1109/MC.2016.145
22. Sung, K.Y., Biswas, J., Learned-Miller, E.G., Levine, B.N., Liberatore, M.: Server-side traffic analysis reveals mobile location information over the internet. IEEE Trans. Mobile Comput. 1 (2018). https://doi.org/10.1109/TMC.2018.2857777
23. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263, October 2016. https://doi.org/10.1109/WINCOM.2016.7777224
24. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. IEEE Trans. Inf. Forensics Secur. **13**(1), 63–78 (2018). https://doi.org/10.1109/TIFS.2017.2737970
25. Xu, J., Ota, K., Dong, M.: Saving energy on the edge: in-memory caching for multi-tier heterogeneous networks. IEEE Commun. Mag. **56**(5), 102–107 (2018). https://doi.org/10.1109/MCOM.2018.1700909
26. Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y.: Internet traffic classification by aggregating correlated naive bayes predictions. IEEE Trans. Inf. Forensics Secur. **8**(1), 5–15 (2013). https://doi.org/10.1109/TIFS.2012.2223675