A Blockchain-Driven IIoT Traffic Classification Service for Edge Computing

Heng Qi, *Member, IEEE*, Junxiao Wang, Wenxin Li, Yuxin Wang, and Tie Qiu, *Senior Member, IEEE*

***Abstract*—Nowadays, more and more sensors, devices and applications are connected in Industrial Internet of Things (IIoT), producing massive real-time flows which need to be scheduled for Quality-of-Service provision. To realize application-aware and adaptive flow scheduling, the problem of traffic classification must be addressed at first. When edge computing paradigm is introduced into IIoT, the traffic classification service can be deployed on edge node in the near-end. Recently, deep-learning-based IIoT traffic classification methods show better performance, but the computational cost of deep learning model is too high to be deployed on edge node. Moreover, increasingly unknown flows generated by new devices and emerging industrial APPs lead to frequent training of traffic classifiers. It is difficult to migrate the complex process of classifier training from cloud server to edge nodes with limited resources. To address these issues, we take the benefits of hash mechanism and consensus mechanism in blockchain to design a lightweight IIoT traffic classification service, which is more applicable for edge computing paradigm. First, inspired by the hash mechanism in blockchain and the learning to hash for big data, we propose a new learning-to-hash method named extension hashing. By this method, we can build the set of binary coding tress (BCT set), then generating hash table for more efficient k-nearest neighbor-based classification without complex classifier training. Then, we design a new voting-based consensus algorithm to synchronize the BCT sets and the hash tables across edge nodes, thereby providing the traffic classification service. Finally, we conduct data-driven simulations to evaluate the proposed service. By comparing traffic classification results on public data set, we can see that the proposed service achieves the highest classification accuracy with the minimal time cost and memory usage.**

***Index Terms*—Blockchain, consensus mechanism, edge computing, extension hashing, Industrial Internet of Things (IIoT), traffic classification.**

I. Introduction

IN THE era of Industry 4.0, Industrial Internet of Things (IIoT) draws more and more attentions from the academia and industry, which offers promising solutions for smart factory, data-driven manufacturing system, intelligent transportation system, and so on [1], [2]. With the rapid development of IIoT, more and more sensors and smart devices are connected while a large number of industrial APPs are arising. These devices and APPs produce a lot of real-time flows which have strict end-to-end delay bounds. It is a challenge to control and schedule these real-time flows to satisfy the Quality-of-Service (QoS) requirements in IIoT [3], [4]. To realize perfect real-time flow scheduling (e.g., application aware, adaptive, etc.), the problem of network traffic classification must be addressed at first [5].

Recently, edge computing paradigm has been applied in IIoT to offload services from cloud servers to edge nodes that are deployed close to end users, thereby reducing the data transmission between cloud servers and IoT devices [6], [7]. Therefore, we can deploy the traffic classification service on edge nodes for efficient flow identification. The state-of-the-art works of IIoT traffic classification show that the deep-learning-based methods are promising solutions, yielding high classification accuracy [8], [9]. However, the following two issues are ignored in existing work.

1) Although some deep learning models can be used in the edge computing paradigm [10]–[12], the performance of deep-learning-based traffic classification will inevitably be subject to constrains of limited resources of the edge node (e.g., CPU and memory). Therefore, we need a IIoT traffic classification solution with low resource-consuming, high efficiency and scalability, which is more applied for edge computing.
2) Increasingly emerging smart devices and industrial APPs generate more and more unknown flows, leading to frequent training of traffic classifiers. The classifier training with large-scale flow samples usually consumes a lot of resources and takes too long time. It is impossible to finish the training process on edge node. Therefore, we need an adaptive traffic classification solution with lowest cost of training, which can be run on the edge node.

To address the above two issues, it is necessary to design a lightweight IIoT traffic classification service in the scene of edge computing. In this article, we borrow the idea of blockchain to achieve this goal. As a underlying technology of decentralized public digital ledger system, blockchain is not limited to electronic payment system while being widely used in numerous distributed systems. Recently, there are a lot of works about the decentralized management and security of edge computing systems with blockchain [13], [14].

Manuscript received July 23, 2020; revised October 6, 2020; accepted October 20, 2020. Date of publication November 3, 2020; date of current version February 4, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102404; in part by NSFC under Grant 62072069, Grant 61772112, and Grant 61672379; and in part by the Science Innovation Foundation of Dalian under Grant 2019J12GX037. *(Corresponding author: Heng Qi.)*

Heng Qi, Junxiao Wang, and Yuxin Wang are with the School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China (e-mail: hengqi@dlut.edu.cn).

Wenxin Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

Tie Qiu is with the School of Computer Science and Technology, College of Intelligence and Computing, Tianjin University, Tianjin 300350, China.

Digital Object Identifier 10.1109/JIOT.2020.3035431

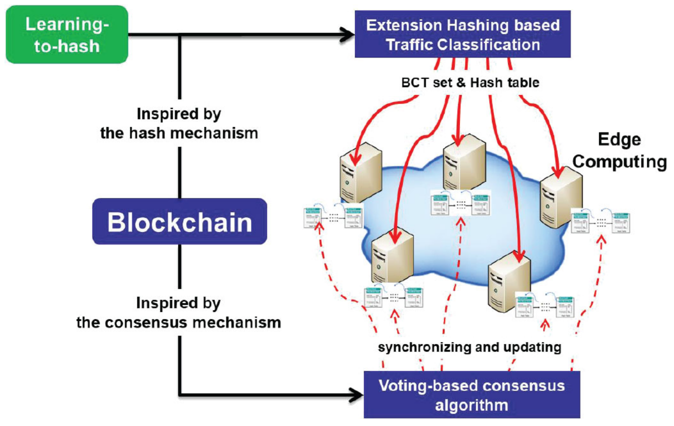
Learning-to-hash
Extension Hashing based Traffic Classification
BCT set & Hash table
Inspired by the hash mechanism
Edge Computing
Blockchain
Inspired by the consensus mechanism
synchronizing and updating
Voting-based consensus algorithm

Fig. 1. Overview of the proposed IIoT traffic classification service.

Inspired by the hash mechanism and the consensus mechanism in the blockchain, we propose an extension hashing-based IIoT traffic classification service which can be deployed across edge nodes. The overview of this service is shown in Fig. 1.

First, we propose an extension hashing-based traffic classification method. In general, the hash mechanism is designed for the security and the privacy of data in blockchain. In our view, the low computational complexity and less memory cost of hash codes are as important as security. For example, the computation of Hamming distance between two hash codes is extremely fast taking only two CPU instructions. Motivated by this, we introduce the learning to hash [15] to implement efficient k-nearest neighbors (kNNs)-based classification. The kNN is one kind of lazy classification algorithm without the process of classifier training. By learning-to-hash algorithms, the kNN-based classification can be mapped to the hash table searching whose execution time and memory cost are both acceptable. However, the learning to hash is limited to the computer vision field because the existing methods are only designed for high-dimensional data. Therefore, we propose a new learning-to-hash method named extension hashing which is more suitable for network flow features. In extension hashing method, we need to build the set of binary coding tress (BCT set) at first, which is used for flow feature encoding. Then, we can build the hash table consisting of hash codes and labels for kNN-based classification. Here, the hash table is not merely the concept in the blockchain, being inspired by the learning to hash for big data as well.

Second, based on the extension hashing, we design a lightweight distributed IIoT traffic classification service in the scene of edge computing. In this service, the BCT set and the hash table are stored on every edge node to implement encoding and kNN-based traffic classification, respectively. When emerging unknown flows coming, we can use existing BCT set to encode the flow features into hash codes, then inserting into hash table. Compared with classifier retraining, this process has advantage of low computational expense. To ensure the accuracy of classification, we also can update the BCT set and the hash table timely while synchronizing them across edge nodes by a proposed voting-based consensus algorithm.

In summary, the main contributions of this article are described as follows.

1) As we know it is the first work to integrate the learning to hash and the blockchain to implement IIoT traffic classification for edge computing. We not only find a feasible solution of traffic classification on edge node, but also widen application fields of the learning to hash and the blockchain.
2) We propose a new learning-to-hash method named extension hashing, which is more suitable for traffic classification with minimal resources. We also propose a voting-based consensus algorithm to implement an IIoT traffic classification service with extension hashing. Compared with existing work, the proposed method has strong adaptiveness, low resource-consuming and high efficiency, without sacrificing accuracy of traffic classification.
3) We conduct extensive data-driven simulations. On the public data set, the simulation results show that the extension hashing-based traffic classification can achieve the highest accuracy with the minimal time cost and memory cost.

The remainder of this article is organized as follows. In Section II, we give an overview of related work about the network traffic classification. In Section III, we propose a new learning-to-hash method named extension hashing, which is more suitable for traffic classification on edge node. In Section IV, we design a voting-based consensus algorithm to build an extension hashing-based IIoT traffic classification service for edge computing. In Section V, we conduct extensive simulations on a public data sets for performance evaluation. Finally, we give the conclusions in Section VI.

II. Related Work

In existing IIoT system, a large amount of data generated by sensors and devices is transferred in the networks. The network issues are the key to the success of IIoT system, such as time-sensitive routing [16], robust network [17], [18], reliable communication [19], and so on. For ensuring the network performance, it is necessary to realize efficient and effective real-time flow scheduling, especially, application-aware flow scheduling and adaptive flow scheduling [20]. To this end, we need to realize the traffic classification for IIoT system, thereby identifying the flows generated by which applications or devices for perfect scheduling.

As an indispensable pillar of network protocol and application identification, traffic classification plays an important role in a variety of network and security activities, such as traffic engineering, QoS, anomaly monitoring, intrusion detection and so on. In the early stage, the port-based methods [21] and the payload-based methods [22] are generally used, but these methods are limited by dynamical port and encrypted traffic. To overcome these limitations, more and more machine learning-based traffic classification methods are proposed, such as modular system combining linear binary classifiers [23], bag-of-flows-based classifier [24], self-learning intelligent classifier [25], robust statistical classifier combining supervised and unsupervised algorithms [26], deep neural networks [27], and so on.

Moreover, software-defined networking (SDN) provides a better way for flow feature sampling [28], thereby realizing machine learning-based traffic classification more easily. Many SDN-based traffic classification systems or frameworks are proposed, such as openflow-based ensemble learning classifiers [29], wildcard-based flow identification for SDN dataplane [30], FlowSeer system [31] and restricted Boltzmann machine-based multimedia flow identification with SDN [32]. With the development of artificial intelligence (AI), deep-learning-based traffic classification methods begin to emerge, such as a combination model of CNN and RNN for IoT traffic classification [8], a capsule network assisted classification model for end-to-end IoT traffic classification [9], a deep-learning-based autonomous identification framework [33], and so on.

Nowadays, the edge computing is utilized in IIoT system to offload the service from the cloud to the edge close to end users, thereby reducing data transmission and improving the scalability and the intelligence of the system [34]–[36]. To ensure the real-time flow scheduling, the traffic classification service need to be deployed on edge nodes. Although the deep-learning-based traffic classification seems to be promising, the computational overhead of these deep-learning-based methods is too high to deploy them on the edge nodes having limited resources. Moreover, the deep-learning-based methods usually used the end-to-end models processing flow traces directly instead of flow features, but the gateway or the SDN controller usually provides flow statistic features or sampling features in real applications. Therefore, we propose a lightweight IIoT traffic classification service, which can be deployed across edge nodes to process the flow features from the gateway or the SDN controller.

In this article, the proposed IIoT traffic classification service is inspired by the ideas of learning to hash and blockchain. The learning to hash is a promising solution for kNN search of large-scale and high-dimensional data in big data and computer vision fields [15], [37]. Based on the idea of learning to hash, we can implement more efficient classification by binary coding and hash table searching, whose execution time and memory cost are both acceptable. Recently, the blockchain are widely used in IoT for network security and data management [38], [39]. Based on the idea of consensus in blockchain, we can ensure the synchronization of binary coding trees (BCTs) and hash tables across edge nodes. As we know this is the first work to study the IIoT traffic classification with learning to hash and blockchain.

III. Extension Hashing-Based Traffic Classification

In this section, we first describe the idea of learning to hash and the motivation of our method. Then, we propose a new learning-to-hash method named extension hashing for traffic classification.

A. Preliminaries

Learning to hash has been viewed as one promising solution of approximate kNN search for big data, which aims to learn a data-dependent hash function from a given data set so that the kNN search results in the hash coding space are as close as possible to the results in the original space [37], [40]. Because of the less memory usage of hash codes and the high search efficiency in hash coding space, it has been widely used in large scale object retrieval, image classification, and other computer vision or machine learning applications.

To implement the traffic classification with learning-to-hash methods, we first learn a hash function from the set of collected flow samples, thereby mapping these samples to hash codes. Then, we can map a given flow sample to a hash code by the same hash function for kNN searching in the hash coding space to identify its category. Although existing learning-to-hash methods can be applied for traffic classification directly, these methods are designed for vision features. Because the vision features extracted from images are usually high-dimensional, the dimensionality reduction is a key step in existing methods for generating compact hash codes. However, the flow samples are usually represented by the feature vectors consisting of several uncorrelated dimensions. Therefore, the dimensionality reduction is not necessary for flow features. Furthermore, we should map the flow features to longer hash codes instead of compact hash codes for preserving more information of original space. To this end, we propose a new learning-to-hash method named extension hashing. The key notations are listed in Table I.

B. Extension Hashing Method

In this article, we design a new learning-to-hash method named extension hashing, which is more suitable for traffic classification. Compared with existing methods, the extension hashing includes the following three tricks to improve the performance of traffic classification.

C. Extension Hash Coding Versus Compact Hash Coding

In existing learning-to-hash methods, the high dimensional feature space is first transformed to a low dimensional feature space consisting of uncorrelated dimensions by dimension-reduction algorithms, e.g., principal component analysis (PCA). Then, each dimension of low dimensional vectors can be mapped to one bit, achieving the compact binary coding. However, the feature space of network flows is not high-dimensional, while each dimension of flow features can be viewed as an independent attribute. Taking into account the significant difference in representation of vision features and flow features, we propose extension binary coding instead of the compact binary coding. We present a simple and efficient algorithm to compute the mean values of one array in an iterative manner, thereby building one BCT. The process of BCT building is described by Algorithm 1, which is motivated by the traditional binary search algorithm.

When we want to map each dimension of traffic flow features into $\mathcal{L}$ bits, we can build one BCT whose height is $\mathcal{L}$ for each dimension by Algorithm 1, thereby d BCTs being built

TABLE I
KEY NOTATIONS IN EXTENSION HASHING METHOD

Notations	Definitions
X_n^d	A matrix including n feature vectors of flow samples, in which each feature vector is $d-$dimensional.
f^d	A $d-$dimensional feature vector of flow sample.
$f^d(j)$	The value on the $j-$ dimension of f^d.
f_u^d	A $d-$dimensional feature vector of unknown flow.
L_n	A set including n labels of flow samples.
l_u	The predict label of the unknown flow.
$\mathcal{L}$	The number of bits corresponding to every dimension of flow features.
h^d	A bit vector including $d * \mathcal{L}$ bits.
$h^d(j)$	The value on the $j-$ dimension of h^d.
H_n	A hash table consisting of n hash codes and n labels of flow samples.
$T_{\mathcal{L}}$	A binary coding tree whose height equals to $\mathcal{L}$.
S_{BCT}	A set of binary coding trees.
$T_{\mathcal{L}}^j$	The $j-$th binary coding tree in S_{BCT}, whose height equals to $\mathcal{L}$.
A_n	An array including n elements.
W^d	An array of d weight values for weighted hamming distance calculation.
$W^d(j)$	The $j-$th weight value in W^d.
$W_{orig}(j)$	The $j-$th original weight value before normalization.
$XSet$	A set of flow sample matrices.

Algorithm 1: The Construction of Binary Coding Tree

```
Input  : $A_n$: an array including $n$ elements;
         $\mathcal{L}$: the number of bits.
Output: $T_{\mathcal{L}}$: a binary coding tree whose height equals to
         $\mathcal{L}$.
1  if $\mathcal{L} == 1$ then
2      Compute the mean value of the array $A_n$:
       $v_m = \frac{\sum_{i=1}^{n} a_i}{n}, a_i \in A_n$.;
3      $T_{\mathcal{L}}.root \leftarrow v_m$;
4      $T_{\mathcal{L}}.lchild \leftarrow NULL$;
5      $T_{\mathcal{L}}.rchild \leftarrow NULL$;
6  else
7      Compute the mean value of the array $A_n$:
       $v_m = \frac{\sum_{i=1}^{n} a_i}{n}, a_i \in A_n$.;
8      for $a_i \in A_n$ do
9          if $a_i - v_m \geq 0$ then
10             Add $a_i$ into a new array $A_n^+$.;
11         else
12             Add $a_i$ into another new array $A_n^-$.;
13     $\mathcal{L} = \mathcal{L} - 1$;
14     $T_{\mathcal{L}}.root \leftarrow v_m$;
15     $T_{\mathcal{L}}.lchild \leftarrow$ Call Alg.1(itself) with the inputs $A_n^-$
       and $\mathcal{L}$;
16     $T_{\mathcal{L}}.rchild \leftarrow$ Call Alg.1(itself) with the inputs $A_n^+$
       and $\mathcal{L}$;
```

for the hash coding of d-dimensional flow features. Based on these d BCTs, we can implement the extension hash encoding, the process of which is shown in Algorithm 2. By Algorithm 2 a d-dimensional flow feature vector $f^d = (f^d(1), \ldots, f^d(d))$ is encoded into a hash code represented by a bit array h^d consisting of $d * \mathcal{L}$ bits. When $\mathcal{L} > 1$ the length of the hash code is greater than the dimensionality of the feature vector. By contrast, when applying the existing learning-to-hash methods, a feature vector can only be encoded into a short hash code

Algorithm 2: The Extension Hash Encoding based on Binary Coding Trees

```
Input  : $f^d$: a $d$-dimensional flow feature vector;
         $S_{BCT} = \{T_{\mathcal{L}}^1, \cdots, T_{\mathcal{L}}^d\}$: a BCT set including $d$
         BCTs whose heights are $\mathcal{L}$.
Output: $h^d$: a bit vector including $d * \mathcal{L}$ bits.
1  for $1 \leq j \leq d$ do
2      $T_{\mathcal{L}} \leftarrow T_{\mathcal{L}}^j$ ;
3      $i = (j-1) * \mathcal{L} + 1$ ;
4      while $T_{\mathcal{L}} \neq NULL$ do
5          if $f^d(j) - T_{\mathcal{L}}.root \geq 0$ then
6              $h^d(i) = 1$;
7              $T_{\mathcal{L}} \leftarrow T_{\mathcal{L}}.rchild$;
8          else
9              $h^d(i) = 0$;
10             $T_{\mathcal{L}} \leftarrow T_{\mathcal{L}}.lchild$;
11         $i = i + 1$;
```

whose length is smaller than its dimensionality. Therefore, we call the proposed method as extension hashing.

D. Weighted Hamming Distance Versus Hamming Distance

The Hamming distance calculation usually consists of two steps. The first step is to compute the result of bitwise XOR between hash codes. The second step is to count the nonzero bits of the result from the first step. Although the Hamming distance calculation is very fast, it usually leads to confusing ranking in kNN search, because it only represents the number of different bits between two hash codes without considering the differential information between bits. For example, the query point q and other two different sample points p_1 and p_2 are encoded to binary codes 01101, 10101, and 01110, respectively. The Hamming distance between q and p_1 is the same as which between q and p_2, leading to confusing ranking. To overcome this drawback, the weighted Hamming

Algorithm 3: The Class Label based Weight Calculation

```
Input  : $X_n^d$: a $n * d$ flow sample matrix consisting of $n$ $d$-dimensional feature vectors of flow samples;
         $L_n$: a class label set consisting of $n$ labels of flow samples.
Output: $W^d$: an array of $d$ weight values where the $j$-th weight corresponds to the $j$-th dimension.
1  Based on the class label set $L_n$, flow samples are divided into $m$ classes to get the index set $I_d$.;
2  $I_d(k)$ includes the indexes of flow samples belonging to the $k$ class.;
3  for $1 \leq j \leq d$ do
4      $N_{all}(j) \leftarrow$ counting the number of distinct values of $X_n^d(:, j)$.;
5      for $1 \leq k \leq m$ do
6          $N_{class}(k, j) \leftarrow$ counting the number of distinct values of $X_n^d(I_d(k), j)$.;
7      $W_{\text{orig}}(j) = \frac{N_{all}(j)}{\sum_{k=1}^{m} N_{class}(k,j)} - 1/m$;
8  Normalization:;
9  for $1 \geq j \geq d$ do
10     $W^d(j) = \frac{W_{\text{orig}}(j)}{\sum_{j=1}^{d} W_{\text{orig}}(j)}$;
```

distance is usually calculated instead of the Hamming distance [41].

In the weighted Hamming distance calculation, one weight value is assigned to each bit, which can represent the importance of this bit. Instead of counting the nonzero bits, the weighted summation is calculated based on the result of bitwise XOR and weight values of bits. The weighted Hamming distance between any two hash codes h_1^d and h_2^d is defined as

$$d_{WH}\left(h_1^d, h_2^d\right) = \sum_{m=1}^{k} W^d(m) * \left|h_1^d(m) - h_2^d(m)\right| \tag{1}$$

where $W^d(m)$ denotes the mth weight value in the weight value array.

To acquire more accurate distance measure, more suitable weight values should be given. In traffic classification task, we infer the class labels of unknown flows based on the labels of known flow samples. Therefore, we also can directly compute the weight values based on the labels and the feature values of known flow samples. To this end, we propose a class label-based weight calculation algorithm shown in Algorithm 3.

Weight values computed by Algorithm 3 represent the impact of the distribution of each dimensional feature values on traffic flow classes. If there is no same feature values on the ith dimension of flow samples from different classes, we have

$$\frac{N_{\text{all}}(j)}{\sum_{k=1}^{m} N_{\text{class}}(k, j)} = 1. \tag{2}$$

Under this condition, the maximum original weight value $W_{\text{orig}}(j) = 1 - 1/m$ is assigned to the ith dimension. In other words, there is a great possibility of distinguishing among different traffic flow classes by this dimension. If the ith dimensional

Algorithm 4: The Extension Hash Table Construction

```
Input  : $X_n^d$: a $n * d$ flow sample matrix consisting of $n$ $d$-dimensional feature vectors of flow samples;
         $L_n$: a class label set consisting of $n$ labels of flow samples;
         $\mathcal{L}$: the number of bits corresponding to each dimension.
Output: $S_{BCT} = \{T_{\mathcal{L}}^1, \cdots, T_{\mathcal{L}}^d\}$: a BCT set including $d$ BCTs whose heights are $\mathcal{L}$;
        $W^d$: an array of $d$ weight values;
        $H_n$: a hash table consisting of $n$ hash codes and $n$ labels of flow samples.
1  for $1 \leq j \leq d$ do
2      $T_{\mathcal{L}}^j \leftarrow$ Call Alg.1 with the inputs $X_n^d(:, j)$ and $\mathcal{L}$;
3      Insert $T_{\mathcal{L}}^j$ into $S_{BCT}$;
4  $W^d \leftarrow$ Call Alg.3 with the inputs $X_n^d$ and $L_n$;
5  for $1 \leq i \leq n$ do
6      $H_n(i).code \leftarrow$ Call Alg.2 with the inputs $X_n^d(i, :)$ and $S_{BCT}$;
7      $H_n(i).label \leftarrow L_n(i)$;
```

feature values of all flow samples are the same, we have

$$\frac{N_{\text{all}}(j)}{\sum_{k=1}^{m} N_{\text{class}}(k, j)} = 1/m. \tag{3}$$

In our view, the ith dimension is meaningless to the traffic classification. Thus, the minimal original weight value $W_{\text{orig}}(j) = 0$ is assigned to the ith dimension. Finally, the original weight values should be normalized to ensure that the summation of all weight values equal to 1.

E. Traffic Classification Using Extension Hashing

Based on the proposed extension hashing method, we can implement the traffic classification efficiently and effectively. The process of extension hashing-based traffic classification can be divided into two stages, namely, the extension hash table construction and the kNN classification with extension hash codes.

In the first stage, we build a CT set based on all d-dimensional feature vectors of flow samples at first. Then, each feature vector is encoded into a hash code based on d BCTs in set. Finally, we get a hash table which consists of all hash codes and labels of flow samples. Because the hash codes occupy less memory, the hash table can be directly loaded into memory for accelerating the kNN search. Algorithm 4 shows the process of extension hash table construction.

In the second stage, the feature vector of unknown flow is also encoded into a hash code based on the BCT set at first. Then, an exhaustive kNN search in hash table can be implemented very efficiently by computing the weighted Hamming distances. Finally, we can predict the label of unknown flow by a majority vote of its k nearest neighbors. Algorithm 5 shows the process of kNN classification with extension hash codes.

Algorithm 5: The Extension Hash Code based kNN Classification

```
Input : $f_u^d$: a $d$-dimensional feature vector of unknown flow;
        $S_{BCT}$: a BCT set including $d$ BCTs;
        $W^d$: an array of $d$ weight values;
        $H_n$: a hash table consisting of $n$ hash codes and $n$ labels of flow samples.
Output: $l_u$: the predict label of the unknown flow.
1 $h_u \leftarrow$ Call Alg.2 with the inputs $f_u$ and $S_{BCT}$;
2 for $1 \le i \le n$ do
3     $Dh(i) = \sum_{j=1}^{d} W^d(j) * \mid h_u^j - H_n(i).code^j \mid$;
4 Rank $Dh$ to find $k$ nearest neighbors;
5 $l_u \leftarrow$ Find the class label most common among the k nearest neighbors;
```

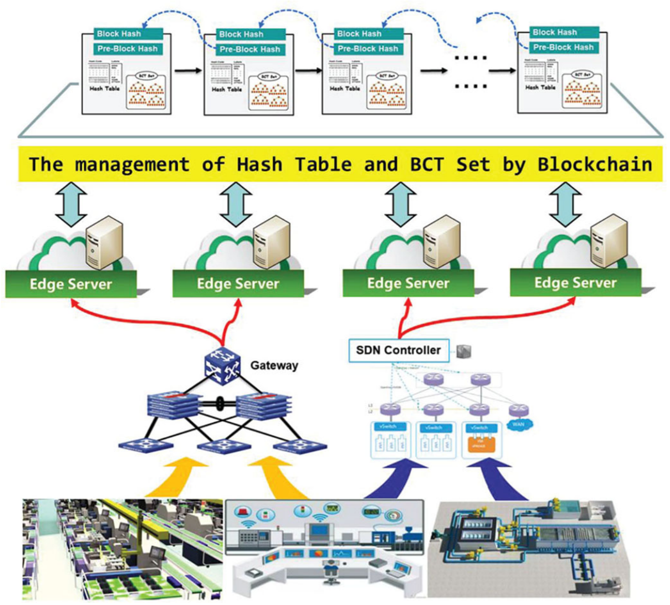

Fig. 2. Blockchain-driven IIoT traffic classification service.

IV. Framework of Blockchain-Driven IIoT Traffic Classification Service

To implement IIoT traffic classification across the edge nodes, we design a blockchain-driven service framework as shown in Fig. 2.

In our service framework, when the gateway or the SDN controller extracting the features of each flow to be classified, it invokes the traffic classification service from the edge server in the near-end, thereby the flow can be distinguished for effective application-aware scheduling or traffic engineering.

To implement extension hashing-based traffic classification service on edge nodes, we can deploy an improved blockchain system for management of BCT sets and hash tables across distributed edge nodes. In the initialization stage, a set of binary coding trees (BCT set) and a hash table are constructed by Algorithm 1 and Algorithm 4, respectively. Then they are both encapsulated into the first block in the blockchain.

Algorithm 6: The Voting-based Consensus for New Hash Table and BCT Set

```
Input : $XSet = \{X_{n_1}^d, \cdots, X_{n_m}^d\}$: $m$ flow sample matrices generated on $m$ edge nodes, respectively;
        $LSet = \{L_{n_1}, \cdots, L_{n_m}\}$: $m$ class label sets corresponding to $m$ flow sample matrices;
        $\mathcal{L}$: the number of bits corresponding to each dimension.
Output: $S_{BCT}^{new}$, $H_n^{new}$: a new BCT set and a new hash table.
1  Step 1: The process of local hash table construction;
2  for $1 \le j \le m$ do
3      $S_{BCT}^j, H_n^j \leftarrow$ Call Alg.4 with the inputs $X_{n_j}^d$, $L_{n_j}$ and $\mathcal{L}$;
4      Broadcasting $S_{BCT}^j$ and $H_n^j$ to other $m-1$ edge nodes;
5  Step 2: The process of voting-based consensus;
6  for $1 \le j \le m$ do
7      for $1 \le k \le m$ do
8          if $k \ne j$ then
9              Using $X_{n_j}^d$ and $L_{n_j}$ as unknown flow matrix and its label set;;
10             Predicting the label set $L_{n_j}^*$ of $X_{n_j}^d$ by Alg.5 with the inputs $S_{BCT}^k$ and $H_n^k$;;
11             Computing the prediction accuracy of $S_{BCT}^k$ and $H_n^k$ by comparing $L_{n_j}$ and $L_{n_j}^*$;
12     Voting for $S_{BCT}^k$ and $H_n^k$ with the highest prediction accuracy;;
13     Broadcasting the vote result to other $m-1$ edge nodes and receiving the vote results from other $m-1$ edge nodes;;
14     if $S_{BCT}^j$ and $H_n^j$ with the highest votes then
15         $S_{BCT}^{new} = S_{BCT}^j$ and $H_n^{new} = H_n^j$;
16         Encapsulating $S_{BCT}^{new}$ and $H_n^{new}$ into a new block.;
17         Broadcasting the new block to other $m-1$ edge nodes;;
18         Receiving the results of validate and Updating blockchain.;
19     else
20         if The new block information from the node $k$ then
21             $S_{BCT}^{new} = S_{BCT}^k$ and $H_n^{new} = H_n^k$;
22             Validating the new block with the voting results and sending back the result of validating;;
23             Waiting the synchronous update of blockchain.;
```

When the traffic classification service is running, the BCT set and the hash table are used for ecoding and kNN-based classification, respectively. In the updating stage, we design a voting-based consensus algorithm shown in Algorithm 6 to generate and validate a new block, including a new hash table and a new BCT set across these edge nodes, while appending the new block into the blockchain. Finally, we can implement traffic classification with the same BCT sets and hash tables across edge nodes. In this service, we also assume each edge node is a validated peer by remote cloud service, thereby constructing permissioned blockchain to manage the hash tables and BCT sets on them.

V. Performance Evaluation

A. Simulation Setup

To evaluate the performance of the proposed traffic classification service, we conduct a large number of data-driven simulations. As we know, there is no public data set of flow statistical features from real IIoT systems. Therefore, we select one public data set of labeled Internet traffic flows, namely, Andrew W. Moore's data set[1] for objective evaluation.

The Andrew W. Moore's data set is usually used to evaluate the feature-based traffic classification methods, which consists of 11 subsets. The first ten subsets are collected from a website by a high-performance network monitor, and each from a different period of the 24-h day. The 11th subset is collected at the same site 12 months later to evaluate how well the traffic classification methods perform when classifying new flow data. In this data set, each flow is characterized by 248 features, including server port, client port, total packets, packet inter arrival time, and so on. One application class label is assigned to each flow. There are 12 different application classes in this data set, namely, "WWW," "MAIL," "FTP-CONTROL," "FTP-PASV," "FTP-DATA," "ATTACK," "P2P," "DATABASE," "MULTIMEDIA," "SERVICES," "INTERACTIVE," and "GAMES." In some work the flows belonging to "FTP-CONTROL," "FTP-PASV" and "FTP-DATA" classes are labeled as "BULK" class, thereby the flows are classified into 10 different classes. In this article, we deal with 12 classes for more accurate classification.

In simulations we focus on the accuracy, the time cost and the memory cost of traffic classification methods. Because the edge node in IIoT system is usually a portable server or a small server with limited resources, we conduct simulations on a general Mac mini PC, which has a 3-GHz Intel Core i7 CPU and 16-GB DDR3 memory.

B. Simulations on the Andrew W. Moore's Data Set

To achieve the fairness of comparison, we adopt the same train and test data in [27] while the same evaluation metric is also used. The train set consists of the first ten subsets of flow samples in the Andrew W. Moore's data set, which includes 377 526 flows. The 11th subset is used as the unknown flow set including 19 626 flows for test. The performance of different traffic classification methods can be assessed by comparing the predict labels and the actual labels given in the data set. The accuracy is used as the evaluation metric, which is the number of unknown flows that were classified correctly divided by the total number of unknown flows.

[1]http://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/data/papers/sigmetrics/index.html

C. Comparison of Accuracy

We first compare the extension hashing-based methods with other three kinds of representative methods.

The first kind of methods is the traditional methods includes the Naïve Bayes (NB), the NB with the kernel density estimation (NB+Kernel), the NB with the feature selection method named fast correlation-based filter (NB+FCBF), the NB with Kernel and FCBF (NB+Kernel+FCBF), the nearest neighbor search (NN) and the NN with the feature selection (NN+FCBF). The NN is special case of kNN with $k = 1$. In [27] the results show that the feature selection provides a major improvement in traffic classification task, while selecting 11 important features from raw 248 features by FCBF. Thus, we also evaluate the methods with feature selection.

The second kind of methods is the newest work on the Andrew W. Moore's data set, which is based on neural network, including the Bayesian neural network (BNN) and the perceptron network (PN).

The third kind of methods is the representative learning-to-hash-based methods, including the iterative quantization hashing-based approximate nearest neighbor search (ITQ-ANN), and the ITQ-ANN with the feature selection (ITQ-ANN+FCBF). Because ITQ hashing only generate the compact hash code, the length of hash code cannot exceed the limit of dimensions of feature vectors. Thus, we evaluate it with 248, 128, and 64 b, respectively. When using ITQ with FCBF, the length of hash codes cannot be greater than 11. Because the proposed extension hashing can yield the longer hash code to improve the performance, we evaluate extension hashing with 248, 496, and 744 b, respectively. For 11-D new features generated by FCBF, we evaluate the extension hashing with 11, 22, and 33 b, respectively. Table II shows the accuracy of these different traffic classification methods.

From Table II we can see that the accuracy of EH-ANN+FCBF is the best when the feature vectors are encoded into 33-b hash codes (each dimension of feature vectors corresponding to 3 b). It demonstrates that it is feasible to introduce the idea of learning to hash into the traffic classification. Although the accuracy of ITQ-ANN is unchanged with different lengthes of hash codes, the accuracy of ITQ-ANN+FCBF is less with the decrease of the length of hash codes. It demonstrates that existing learning-to-hash methods cannot be directly used for traffic classification. The compact binary coding does not apply to the features of Internet traffic. We also can see that the accuracies of EH-ANN and EH-ANN+FCBF are both increasing as the length of hash codes increases. It is verified that more information of features can be preserved by extension binary coding.

From Table II we also find one interesting point. The accuracies of EH-ANN and EH-ANN+FCBF are higher than that of NN and NN+FCBF. In computer vision field, the learning-to-hash methods are usually used to find the approximate nearest neighbor instead of the exact nearest neighbor. These methods try to obtain less memory usage and high search efficiency by sacrificing the accuracy. However, in the Internet traffic classification, we can get higher accuracy. We think the reason is the differences of feature spaces. For the features of

TABLE II
ACCURACY OF DIFFERENT TRAFFIC CLASSIFICATION METHODS ON THE ANDREW W. MOORE'S DATA SET

Methods		Accuracy
Traditional Methods	NB	20.75%
	NB+Kernel	37.65%
	NB+FCBF	93.38%
	NB+Kernel+FCBF	93.73%
	NN	86.76%
	NN+FCBF	95.32%
Neural Network	BNN	95.3%
	PN	96.7%
Representative Learning-to-Hash	ITQ-ANN(248-bit hash code)	79.47%
	ITQ-ANN(128-bit hash code)	79.47%
	ITQ-ANN(64-bit hash code)	79.47%
	ITQ-ANN+FCBF(11-bit hash code)	82.96%
	ITQ-ANN+FCBF(8-bit hash code)	83.01%
	ITQ-ANN+FCBF(4-bit hash code)	81.68%
Extension Hashing	EH-ANN(248-bit hash code)	88.94%
	EH-ANN(496-bit hash code)	95.49%
	EH-ANN(744-bit hash code)	95.49%
	EH-ANN+FCBF(11-bit hash code)	86.59%
	EH-ANN+FCBF(22-bit hash code)	95.48%
	EH-ANN+FCBF(33-bit hash code)	**98.72%**

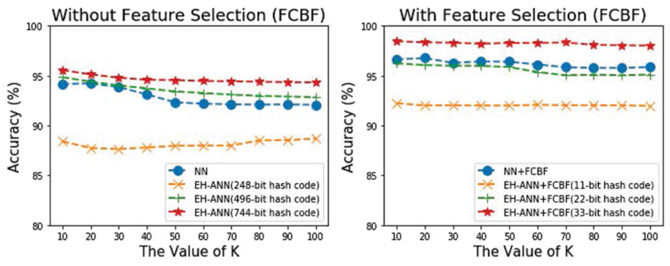

Fig. 3. Accuracies of different traffic classification methods with different k-values.

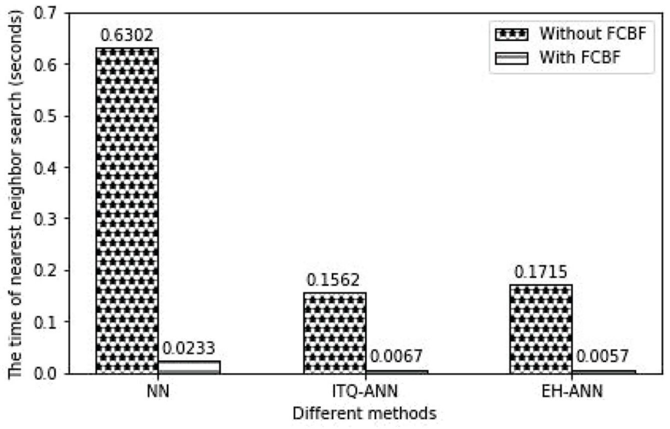

Fig. 4. Time costs of different traffic classification methods.

Internet traffic, we can get a more suitable feature space for classification by extension hashing.

Then, we evaluate the effect of the k-value on the performance of traffic classification. In simulations, we evaluate the accuracies of NN, EH-ANN, NN+FCBF and EH-ANN+FCBF with $k = 10, 20, \ldots, 100$, respectively. The results are shown in Fig. 3.

From Fig. 3, we can see that the effect of the k-value on the performance of traffic classification is little. When using the hash codes consisting of more bits, the accuracies of traffic classification are increased. It is evaluated that the extension hashing is reasonable. Moreover, we find that the increasing of accuracies cannot be sustainable as the length of hash codes increases. This point is definitely worth further investigation.

D. Evaluations of Time Cost

In this section, we evaluate the time cost of different traffic classification methods. In simulations, 19 626 unknown flows in the 11th subset are classified by each of methods, thereby computing the average time spent in classification as the time cost. Fig. 4 shows the time costs of NN, ITQ-ANN, EH-ANN, NN+FCBF, ITQ-ANN+FCBF and EH-ANN+FCBF. The time costs of NN and NN-FCBF are mainly for the nearest neighbor search, while the time costs of other four learning-to-hash-based methods are for the feature encoding and the nearest neighbor search. Although feature encoding brings extra time cost, it is trivial in comparison to the time cost of nearest neighbor search (see Table III).

From Fig. 4, we can see that the time costs of ITQ-ANN and EH-ANN are still significantly less than that of NN whether using feature selection method (FCBF) or not. It is verified that we can take advantage of learning to hash to decrease the time cost sharply. We also can see that the feature selection is very important for traffic classification, by which we can get much smaller and more useful features to reduce the time cost and the memory usage of traffic classification.

TABLE III
TIME COSTS OF HASH TABLE CONSTRUCTION, FEATURE ENCODING AND NEAREST NEIGHBOR SEARCH

Methods	Time Cost (seconds)		
	hash table construction	feature encoding	nearest neighbor search
ITQ-ANN(248-bit)	163.46	7.50E-6	0.1562
EH-ANN(248-bit)	6.12	1.02E-5	0.1715
ITQ-ANN+FCBF(11-bit)	4.12	**1.17E-7**	0.0067
EH-ANN+FCBF(11-bit)	**0.23**	1.01E-6	**0.0057**

From Table III, we can see that the time costs of hash table construction and feature encoding, which are acceptable for traffic classification. Moreover, by the proposed EH-based methods, we can reconstruct the hash table more quickly, thereby not affecting the on-line traffic classification. We also can see that the time cost is affected by the number of bits of hash codes. To further reduce the time cost, the feature selection can be used. Table III shows that the time cost of EH-ANN+FCBF (11 b) is only $0.0057 + 1.01E - 6$ seconds. Compared with classifier training, it is suitable for real-time traffic classification.

E. Evaluations of Memory Usage

In nearest neighbor search, all samples need to be loaded into memory to accelerate the process of search. Therefore, the memory usage is represented by the size of data loaded into memory for traffic classification in the experiments. By NN, 377 526 feature vectors of sample flows are loaded into memory. If learning-to-hash-based methods are used instead of NN, we only load 377 526 hash codes into memory. Obviously, the size of hash codes is significantly smaller than that of feature vectors. Fig. 5 shows that we can take advantage of learning to hash and feature selection to achieve the lowest memory usage.

From Fig. 5, we can see that the size of raw features of sample flows is above 700 MB, while the size of 248-b hash codes of sample flows is less than 90 MB. It is verified that it is more feasible to load hash codes into memory instead of raw feature vectors. The hash codes with the same lengthes have the same memory usage. The EH-ANN-based methods only generates longer hash codes, but the accuracies of traffic classification are more higher. Therefore, from two perspectives: 1) accuracy and 2) memory usage, the EH-ANN-based methods with feature selection are promising.

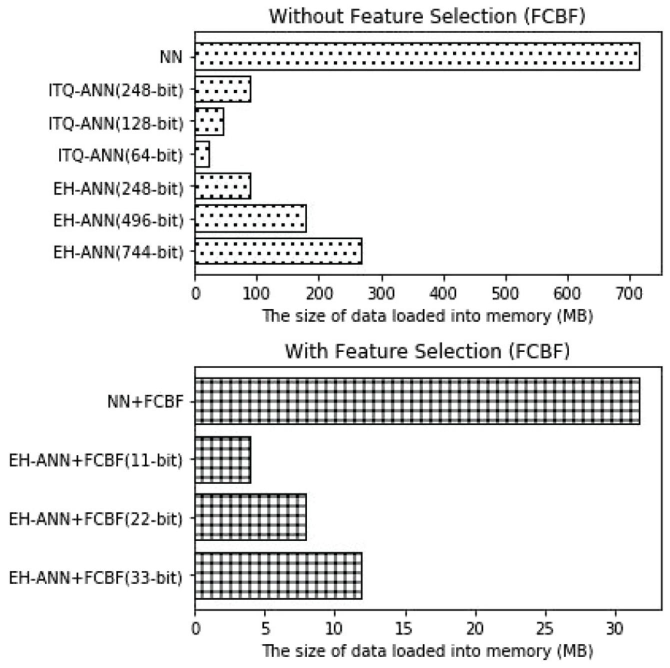

Fig. 5. Memory usage of different traffic classification methods.

F. Discussion About Simulation Results

From the above evaluations, we can see that the extension hashing-based traffic classification yields the highest accuracy with minimal time cost and memory usage. We also find the accuracy of classification can be improved by introducing fast correlation-based filter (FCBF). The FCBF is one kind of feature selection algorithm, by which we can select a little important features instead of all features to avoid the effect of irrelevant features on traffic classification, thereby improving the performance. Although we can obtain high accuracy with shorter hash code after feature selection, the extension hashing can still be used for mapping the selected features to longer hash codes, leading to higher accuracy. Therefore, we can get the highest accuracy by "EH-ANN+FCBF" method with 33-b hash code in the above evaluations.

Furthermore, we can keep increasing the length of hash codes to obtain higher accuracy. However, the longer hash codes yield higher time cost and memory usage. Table IV shows the performance of EH-ANN+FCBF with hash codes of different length. From Table IV, we can see that the improvement of accuracies is little when encoding each dimension of flow features into more than 3 b, while the growths of time cost and memory usage are fast. Therefore, we think 33 b is an appropriate length for EH-ANN+FCBF method on the Andrew W. Moore's data set.

To simulate the actual status of edge nodes, the above evaluations are implemented on general Mac mini PC. The extension hashing-based traffic classification can be deployed on edge nodes with limited resources. In distributed edge computing, the BCT sets and the hash tables are same on edge nodes. The performance of traffic classification on every edge node likes the above stand-alone simulation results. Many unknown flows in real-world scenario may affect the performance. In the future work, we will build real service prototype for evaluation.

TABLE IV
PERFORMANCE OF EH-ANN+FCBF WITH HASH CODES OF DIFFERENT LENGTH

Methods	Accuracy	Time Cost of kNN (seconds)	Memory Usage (MB)
EH-ANN+FCBF(11-bit hash code)	86.59%	0.0057	3.96
EH-ANN+FCBF(22-bit hash code)	95.48%	0.0134	7.92
EH-ANN+FCBF(33-bit hash code)	98.72%	0.0203	11.88
EH-ANN+FCBF(44-bit hash code)	98.78%	0.0239	15.84
EH-ANN+FCBF(55-bit hash code)	98.92%	0.0313	19.8

VI. CONCLUSION

To implement the IIoT traffic classification for edge computing paradigm, we propose a blockchain-driven service framework in this article. First, inspired by the hash mechanism, we borrow the idea of learning to hash to propose an extension hashing method for efficient and scalable traffic classification. Then, inspired by the consensus mechanism, we propose a voting-based consensus algorithm for synchronizing and updating BCT sets and hash tables needed for extension-hashing-based traffic classification across edge nodes. Finally, extensive data-driven simulations show that we can take full advantage of extension hashing to achieve the highest classification accuracy with the minimal time cost and memory usage. Therefore, we can implement a lightweight IIoT traffic classification service in the scene of edge computing, which has strong adaptiveness, low resource-consuming and high efficiency.

REFERENCES

[1] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[2] T. Qiu, B. Li, W. Qu, E. Ahmed, and X. Wang, "TOSG: A topology optimization scheme with global small world for industrial heterogeneous Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3174–3184, Jun. 2019.

[3] J. W. Guck, A. V. Bemten, and W. Kellerer, "DetServ: Network models for real-time QoS provisioning in SDN-based industrial environments," *IEEE Trans. Netw. Service Mang.*, vol. 14, no. 4, pp. 1003–1017, Dec. 2017.

[4] Y. Han, D. Guo, W. Cai, X. Wang, and V. C. M. Leung, "Virtual machine placement optimization in mobile cloud gaming through QoE-oriented resource competition," *IEEE Trans. Cloud Comput.*, early access, Jun. 12, 2020, doi: 10.1109/TCC.2020.3002023.

[5] G. S. Aujla, A. Singh, and N. Kumar, "AdaptFlow: Adaptive flow forwarding scheme for software-defined industrial networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5843–5851, Jul. 2020, doi: 10.1109/JIOT.2019.2951235.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[7] S. Shen, Y. Han, X. Wang, and Y. Wang, "Computation offloading with multiple agents in edge-computing–supported IoT," *ACM Trans. Sens. Netw.*, vol. 16, no. 1, pp. 1–27, 2020.

[8] M. Lopez-Martion, B. Carro, A. Sanchez, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[9] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted IoT traffic classification mechanism for smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7515–7525, Oct. 2019.

[10] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[11] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[12] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May/Jun. 2019.

[13] R. Yang, F. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[14] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud-edge-end in IoT: A blockchain-assisted collective *Q*-learning approach," *IEEE Internet Things J.*, early access, Jul. 7, 2020, doi: 10.1109/JIOT.2020.3007650.

[15] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data—A survey," *Proc. IEEE*, vol. 104, no. 1, pp. 34–57, Jan. 2016.

[16] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, May 2018.

[17] N. Chen, T. Qiu, X. Zhou, K. Li, and M. Atiquzzaman, "An intelligent robust networking mechanism for the Internet of Things," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 91–95, Nov. 2019.

[18] T. Qiu, J. Liu, W. Si, and D. O. Wu, "Robustness optimization scheme with multi-population co-evolution for scale-free wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1028–1042, Jun. 2019.

[19] B. Zhou, Q. Yang, Y. Wang, and C. Wu, "Reliable communication in transmission grids based on nondisjoint path aggregation using software-defined networking," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 842–855, Feb. 2019.

[20] L. Gu, D. Zeng, W. Li, S. Guo, A. Y. Zomaya, and H. Jin, "Intelligent VNF orchestration and flow scheduling via model-assisted deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 279–291, Feb. 2020.

[21] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th Int. Conf. World Wide Web (WWW)*, New York, NY, USA, May 2004, pp. 512–521.

[22] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Aug. 2005, pp. 229–240.

[23] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Trans. Knowl. Discovery. Data*, vol. 6, no. 1, pp. 1–34, 2012.

[24] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 5–15, 2013.

[25] D. Divakaran, L. Su, Y. Liau, and V. Thing, "SLIC: Self-learning intelligent classifier for network traffic," *Comput. Netw.*, vol. 91, pp. 283–297, Nov. 2015.

[26] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.

[27] A. Michael, E. Valla, N. Neggatu, and A. W. Moore, "Network traffic classification via neural networks," Dept. Comput. Lab., Univ. Cambridge, Cambridge, U.K., Rep. UCAM-CL-TR-912, 2017.

[28] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: Spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 999–1013, Jun. 2020.

[29] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *Proc. 24th IEEE Int. Conf. Netw. Protocols (ICNP)*, Singapore, Nov. 2016, pp. 1–5.

[30] H. Xu, Z. Yu, C. Qian, X.-Y. Li, Z. Liu, and L. Huang, "Minimizing flow statistics collection cost using wildcard-based requests in SDNs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3587–3601, Dec. 2017.

[31] S. Chao, K. Lin, and M. Chen, "Flow classification for software-defined data centers using stream mining," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 105–116, Jan./Feb. 2019.

[32] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 566–578, Mar. 2019.

[33] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for DL-based traffic classifier update," 2020. [Online]. Available: arXiv:2002.06359.

[34] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[35] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020, doi: 10.1109/JIOT.2020.2986803.

[36] D. Zeng, L. Gu, and H. Yao, "Towards energy efficient service composition in green energy powered cyber–physical fog systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 757–765, Apr. 2020.

[37] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[38] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.

[39] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.

[40] Y. Cao, H. Qi, W. Zhou, J. Kato, K. Li, X. Liu, and J. Gui, "Binary hashing for approximate nearest neighbor search on big data: A survey," *IEEE Access*, vol. 6, pp. 2039–2054, 2017.

[41] Y. Cao, H. Qi, J. Kato, and K. Li, "Hash ranking with weighted asymmetric distance for image search," *IEEE Trans. Comput. Imag.*, vol. 3, no. 4, pp. 1008–1019, Dec. 2017.

Heng Qi (Member, IEEE) received the B.S. degree from Hunan University, Changsha, China, in 2004, and the M.E. and Ph.D. degrees from the Dalian University of Technology, Dalian, China, in 2006 and 2012, respectively.

He was a JSPS Oversea Research Fellow with the Graduate School of Information Science, Nagoya University, Nagoya, Japan, from 2016 to 2017. He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology. He has published over 100 technical papers, such as the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON MULTIMEDIA, and INFOCOM. His research interests include computer network and multimedia computing.

Junxiao Wang received the B.S. degree from Dalian Maritime University, Dalian, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Dalian University of Technology, Dalian.

His current research interests include software defined network, network function virtualization, and cloud computing.

Wenxin Li received the B.E. and Ph.D. degrees from the School of Computer Science and Technology, Dalian University of Technology, Dalian, China, in 2012 and 2018, respectively.

He is currently a Postdoctoral Researcher with the Hong Kong University of Science and Technology, Hong Kong. From May 2014 to May 2015, he was a Research Assistant with the National University of Defense Technology, Changsha, China. From October 2016 to September 2017, he was a visiting student with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. His research interests include data center networks and cloud computing.

Yuxin Wang received the M.S. degree in computer application and the Ph.D. degree in computer science in from Dalian University of Technology, Dalian, P.R. China, in 2000 and 2012, respectively.

He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology. His research interests include distributed computing, big data analysis, deep learning, and computer vision.

Tie Qiu (Senior Member, IEEE) received the Ph.D. degree in computer science from Dalian University of Technology, Dalian, China, in 2012.

He is currently a Full Professor with the School of Computer Science and Technology, Tianjin University, Tianjin, China. Prior to this position, he held an Assistant Professor in 2008 and an Associate Professor in 2013 with the School of Software, Dalian University of Technology. He was a Visiting Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA, from 2014 to 2015. He has authored/coauthored 9 books, over 150 scientific papers in international journals and conference proceedings, such as IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, *IEEE Communications Magazine*, INFOCOM, and GLOBECOM. There are 10 papers listed as ESI highly cited papers. He has contributed to the development of 4 copyrighted software systems and invented 16 patents.

Prof. Qiu serves as an Associate Editor for IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, an Area Editor of *Ad Hoc Networks* (Elsevier), an Associate Editor of *Computers and Electrical Engineering* (Elsevier), *Human-Centric Computing and Information Sciences* (Springer), and a Guest Editor of *Future Generation Computer Systems*. He serves as the general chair, the program chair, the workshop chair, the publicity chair, the publication chair, or a TPC Member of a number of international conferences. He is a Senior Member of China Computer Federation and a Senior Member of ACM.