Hindawi
Complexity
Volume 2019, Article ID 5498606, 12 pages
https://doi.org/10.1155/2019/5498606

Research Article

Deep Learning Based Proactive Caching for Effective WSN-Enabled Vision Applications

Fangyuan Lei,[1,2] **Jun Cai**,[1] **Qingyun Dai**,[1,2] **and Huimin Zhao**[3]

[1]*School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510640, China*
[2]*School of Information Engineering, Guangdong University of Technology, Guangzhou, China*
[3]*School of Computer Sciences, Guangdong Polytechnic Normal University, Guangzhou 510640, China*

Correspondence should be addressed to Jun Cai; 597069873@qq.com and Huimin Zhao; zhaohuimin@gpnu.edu.cn

Received 15 November 2018; Revised 28 February 2019; Accepted 14 March 2019; Published 2 May 2019

Guest Editor: Jungong Han

Wireless Sensor Networks (WSNs) have a wide range of applications scenarios in computer vision, from pedestrian detection to robotic visual navigation. In response to the growing visual data services in WSNs, we propose a proactive caching strategy based on Stacked Sparse Autoencoder (SSAE) to predict content popularity (PCDS2AW). Firstly, based on Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies, a distributed deep learning network SSAE is constructed in the sink nodes and control nodes of the WSN network. Then, the SSAE network structure parameters and network model parameters are optimized through training. The proactive cache strategy implementation procedure is divided into four steps. (1) The SDN controller is responsible for dynamically collecting user request data package information in the WSNs network. (2) The SSAEs predicts the packet popularity based on the SDN controller obtaining user request data. (3) The SDN controller generates a corresponding proactive cache strategy according to the popularity prediction result. (4) Implement the proactive caching strategy at the WSNs cache node. In the simulation, we compare the influence of spatiotemporal data on the SSAE network structure. Compared with the classic caching strategy Hash + LRU, Betw + LRU, and classic prediction algorithms SVM and BPNN, the proposed PCDS2AW proactive caching strategy can significantly improve WSN performance.

1. Introduction

According to the latest Cisco release of the Visual Network Index (VNI) [1] forecast, the number of devices connected to the Internet of Things (IoT) is projected to expand to somewhere between 20 and 46 billion by 2021. Thanks to the simple deployment and various practical applications, the potential benefits of wireless sensor networks (WSNs) concern a wide range of application scenarios, from pedestrian detection to robot vision navigation, from industrial systems to home appliances. While billions of new devices connect to the network in a short period of time in WSNs, there will have huge data traffic. Therefore, congestion control and data share should be considered. As Figure 1 shows, the WSN network is deployed for traffic monitor, which composed of different network protocols that cannot be directly connected, and it is difficult to achieve fast sharing of sensing data to meet the performance requirements of the system.

For specific applications, once the distributed WSN is deployed for traffic monitor, the sensor node handles not only the sensing tasks but also maintenance of the route status. With the solidified resource management mode, when the upper-layer application requirements change, it is very difficult to apply flexible changes according to the new requirements. Therefore, it is seriously wasting resources without realizing dynamic perception.

Software-defined network (SDN) [2] as a new type of network architecture attracting attention in recent years is applied to future networks. The core idea of SDN is the separation of control plane and forwarding plane. The control plane is aware of network status and network resources, and then the central controller flexibly and dynamically configures the logic control functions and high-level policies of the network. On the data plane, this configuration can be performed without affecting the normal network traffic. Network Function Virtualization (NFV) [3] applies the goal

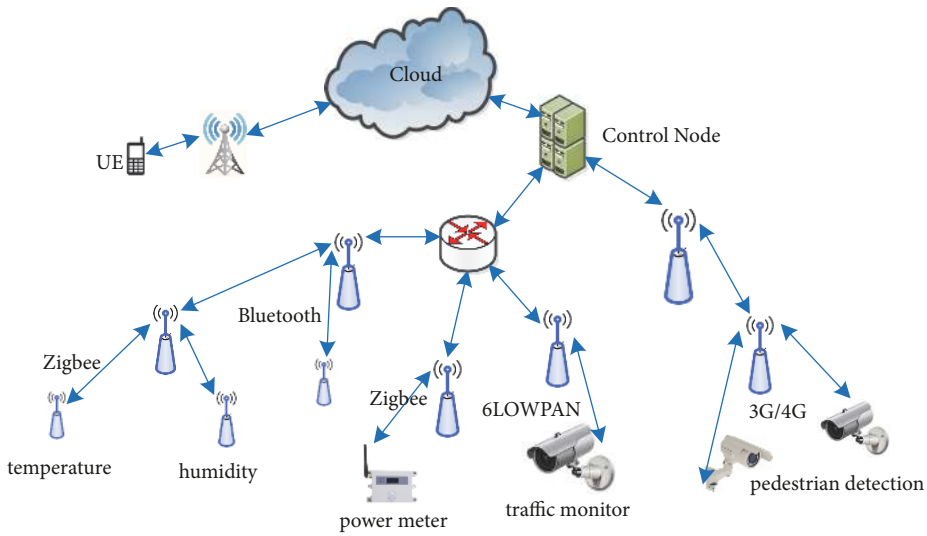

Figure 1: WSNs application in traffic monitor.

of automated management and distributed deployment by introducing virtualization technology. SDN and NFV are introduced to improve the performance in WSNs [4].

To address the congestion issue, proactive caching [5, 6], which wildly used in ICN, is introduced into the WSNs [7]. Data package caching will improve the performance of WSNs, including reducing packet latency, network traffic, and BER of transmission. Recently, proactive caching based on content popularity prediction of Deep Learning (DL), has attracted widespread attention in academia and industry [8–13], which will significantly benefit cache efficiency. Content popularity prediction based on DL can automatically find the rule from the data and use this rule to predict the unknown data. Therefore, in WSN data, caching would significantly improve resource utilization and network performance.

Therefore, we propose an efficient proactive cache strategy based on distributed stacked sparse autoencoder in WSNs (PCDS2AW) for data package content popularity prediction. Firstly, NFV functions are used to virtual part of hardware resources in the control node and sink nodes and routers of general hardware. Then, based on those virtual hardware resources, a distributed deep learning network, SSAEs, is constructed. Next, the SDN controller works in conjunction with global cache resources and SSAEs. At the WSNs network level, the SDN controller can dynamically sense the cache utilization information and periodically collect user data packet request information. These historical requests and real-time requested packet information are input into the distributed SSAE for content popularity prediction. Finally, based on the data package, popularity prediction the SDN controller generates the proactive cache strategy and synchronizes it to the WSNs cache nodes through the SDN flow table to implement content caching and replacement.

As a virtual technology, NFV can provide support for reference machine learning and can achieve seamless cooperation with SDN [4]. SDN uses stream-based data forwarding, with a focus on data exchange and forwarding, while WSN is constrained by the specific type of sensor deployed, which is essentially data-oriented. Therefore, based on NFV, machine learning algorithms are used to predict network traffic, and SDN realizes fast forwarding and complements WSN's efficient perception.

In order to implement cache strategy based on the global data package content popularity prediction, the SDN controller should know the popularity of all requested data package information in WSNs. Therefore, the SDN/NFV technology is used in upper-layer include sink nodes, control nodes, and router to construct some virtual content request and statistics servers. These statistics servers are configured to collect the request data package information from the WSN nodes. The content request information is aggregated to a virtual global content request statistics server. Therefore, the control node collects a large amount of global data package request information and uses the deep learning algorithm to build a prediction content popularity model and then uses the prediction model to guide the WSNs for efficient caching to reduce traffic.

The main contributions of this paper are as follows: (1) in the WSNs, we propose a method for constructing a distributed stack sparse autoencoder deep learning network; (2) SSAEs use the spatiotemporal information of user request data packets to predict the data packet popularity; (3) under the cooperation of SDN controller, cache nodes implement the proactive cache and replacement of the data packet content of the whole network, which makes the utilization of cache resources more reasonable; (4) simulation results show that, compared with Betw [14], Hash [15], and Opportunistic [16], the proposed proactive caching strategy could improve the performance of WSNs.

The remainder of this article is structured as follows. In Section 2, related works are described. In Section 3, the system model is proposed. In Section 4, some evaluation metrics and the numerical simulation results are discussed. Finally, we conclude in Section 5.

2. Related Works

In order to meet the needs of the interconnection of all kinds of objects, the distributed WSN for application design is beginning to transform to support heterogeneous interconnection, which is one of the root causes of SDN introduction of WSN.

In 2012, Luo [17]and Mahmud [18] discussed the integration of SDN and WSN almost simultaneously and made important contributions to the birth of SDWSN. Zeng et al. [19] proposed a software-defined sensor networks (SDSN) architecture that supports "sensing as a service" combined with cloud computing, in which architecture and the controllers are wirelessly used for different sensing tasks. In [20], the author proposed Elon system which implements the function of dynamically modifying the sensing node code by means of replaceable components. Subsequently, the author's team further implemented the transmission on the SDSN. In [21], based on a new event-triggered strategy, the author proposed an event-triggered distributed multisensor data fusion algorithm for wireless sensor networks (WSNs). In [22], the authors deployed multiple controllers to build a more flexible control channel, which greatly reduced the average network control delay. The dynamic redefinition function of the node function [23] could effectively extend the network lifetime. In [24], the authors proposed a general architecture for implementing the controller on the WSN base station to enhance the intelligent management of the network. In [25], the authors defined the cluster head as a switch in the wired network, and the only controller is deployed in the WSN base station to save the sensor node energy. 0The author [26] proposed UbiFlow, which was a software-defined IoT system for ubiquitous flow control and mobility management in a large number of heterogeneous WSNs.

SDN and NFV are not combined standard, and they provide different aspects of network-based services. Innovative network paradigms SDN and NFV have attracted the interest of IoT network operators and service providers. Some researchers have introduced SDN into the IOT network architecture, such as SDN-WISE [27], SDWN [28], TinySDN [22], and UbiFlow [26]. However, many of these architectures lacked virtualization methods. In [4], in order to quickly respond to the challenges brought by flexible deployment of the Internet of Things, the author combined NFV technology with SDN technology and proposed a general SDN-IoT architecture. In [29], to verify and test the 6LoWPAN testbed, a customized Software Defined–Network Functioning Virtualisation (SD–NFV) is proposed.

As a kind of technology to improve network performance, cache technology is widely used in future Internet and communication. Recently, caching has also been a concern in the Internet of things. In [30], the author introduced the in-network caching to the IOT. Firstly, the author defined the lifetime of IOT transmission data, which was determined by application time and location tags, then a trade-off model between multi-hop cost and data freshness was proposed and applied to the content router of the Internet of things. In [31], to adapt highly dynamic environments with a coarse knowledge of car trajectories, the author proposed a Mobility-Aware Probabilistic (MAP) edge caching strategy. In [32], to enhance the service in mobile ad hoc networks (MANNET), the authors proposed a comprehensive strategy which includes cache placement, cache discovery, cache consistency, and cache replacement algorithms. In [33], the author developed a dynamic source rate control algorithm based on cache awareness. Depending on the network traffic, the rate control algorithm used a cache management policy (such as a cache elimination policy and a cache size allocation) to move the transmission window so that packet loss may be mitigated during high speed. In [34], the caching mechanism which was applied to the transport protocol would effectively reduce the number of end-to-end retransmissions, node power consumption, and packet delay to improve network performance. In [35], the author had comprehensive evaluation of cache utilization characteristics with the cache replacement techniques. When caching techniques were applied in the WSNs, the simulation results showed that the performance of packet loss rate, power consumption, and packet transmission delay improved.

Content popularity prediction is one of the key points for caching in which content should be storages; there is no literature to mention in WSN fields. The author [36] used the Markov model to predict sensor operation and proposed a smart cache model based on sensor power to improve cache hit ratios in the ICN-based IoT sensor networks. In [15], based on content popularity, the author proposed a caching decision policy, which allowed a single ICN router to cache content more or less according to the popularity characteristics of the content. In [37], the author proposed a regression method that supported vector regression and Gaussian radial basis functions to predict the popularity of YouTube and Facebook online videos. Mao [38] proposed a multitasking learning (MTL) module and a relational network (RN). The module's general prediction model used to predict TV drama views. In [39], a bi-directional long-term short-term memory neural network (BiLSTM) was proposed to predict the prevalence of online content. Studied in video and news texts, data sets have shown that deep network performance is greatly improved. In [40], the authors proposed that content popularity predictions could translate as classification problems and then made the end-to-end multimodal prediction based on the deep neural networks. In the experiment, text information and visual information were used to verify the validity of the models. In [41], to reduce congestion of backhaul network traffic, the authors proposed to predict user request based on file popularity and user and file patterns, during off-peak request the system proactive cached files. Through the acquisition of mobile subscriber service data of multiple base stations within a few hours interval, analysis was conducted on the big data platform to study the extent of evaluation of the content popularity, and proactive caching was performed [42]. The result showed that the level of satisfaction with user requests could be increased, while the backhaul network traffic could be reduced.

Deep learning has raised concerns in WSN and IOT. In SDN-IoT network [43], to forecast the congestion and traffic load, the author proposed a deep learning algorithm

to predict the future traffic load. In [44], in order to predict the evolution of traffic in the global network, the author proposed a hybrid convolution long-term memory neural network (CRS-ConvLSTM NN) model based on critical path.

Our research shows an effort to be combined with SDN, NFV, DL, and WSN. The proposed proactive cache strategy provides a reliable and efficient method to share the network resource of WSN cache devices. We propose a simple structure of SDN / NFV combined with SSAEs to solve the challenges of WSN, especially in traffic load and congestion management issues. This will improve the network efficiency and flexibility of WSN applications.

3. Methods

3.1. Distribute Deep Learning Networks Architecture on the WSN. The distributed deep learning network system architecture is shown in Figure 2. In WSN, the upper-layer transmission hardware consists of sink nodes, control nodes, and routers. All of the elements are SDN-enabled architecture, which realizes the separation of the control plane and the forwarding plane. With calculation and caching function, these network elements cooperate through the SDN controller. Therefore, the NFV/SDN technology is utilized to construct virtual distributed deep learning network architecture. In the WSN, those devices which have cache resources, such as sink node, router, sub-control node, and main control node, will share partial hardware by NFV to construct the deep learning network. The sink node/router resource will be the input part, and the main control node is the major calculation resource. Then the SDN controller is constructed on the main controller node. With SDN enabled in the WSN, SDN community is with sink node and router through the flow table. Therefore, the main control node may be dynamically aware of the status of the WSN.

On the distributed deep learning network, we construct the SSAEs. The hidden layers and output layer are on the main controller. In our model, SSAEs will predict the data package popularity in the future through historical user data. Therefore, a statistics server will be a virtual construct on the main control node. Those sink nodes will collect the local data package request information and send it to the statistics server. During the training stage, the statistics server will provide the historical data to the input layers. And the prediction stage, the statistics service summarizes the several timeslot request data for the input layers. Then the SSAEs make a prediction for data package in the future. The SDN controller generates a cache strategy based on the prediction. And the cache strategy is synchronized to those cache nodes to implement proactive caching in the period.

3.2. Deep Learning Network

3.2.1. Sparse Auto-Encoder. As an unsupervised feature learning method is widely studied in the field of deep learning, Sparse Auto-Encoder (SAE) has the capability to find a concise and efficient representation of complex data.

The SAE network has three different layers: input layer, hidden layer, and output layer. In order to obtain the optimal hidden layer parameters, that is, to minimize the SAE reconstruction error, the SAE network requires that the output layer be equal to the input layer. The data procedure of SAE is divided into encoding and decoding.

Encoding: The encoding process is to obtain feature y of the input layer value x. The original user requirement data is denoted as vector $x = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]^T$; parameter n means the total number of the input nodes.

$$y = f(x) = \delta(Wx + b) \tag{1}$$

where vector $y = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]^T$ indicated the feature expression of the hidden layer, and parameter m denotes the nodes of the hidden layers. Parameter b denotes the bias vectors, and W represents the weight matrix from input layer to hidden layer. $\delta(x)$ represents the activation function.

Decoding: The decoding process is to obtain the reconstructed vector z of the output layer from the hidden layer value y.

$$z = g(y) = \delta\left(W^T y + b'\right) \tag{2}$$

where $z = [z^{(1)}, z^{(2)}, \ldots, z^{(n)}]^T$, y denotes the feature expression of output layer, and b' denotes the bias vector.

In the SAE feature learning process, in order to minimize the loss caused by coding, the basic requirement is that the reconstructed output is close to the input. In the process of constructing the loss function, in order to learn more optimized sparse features, the sparse penalty term is added to the objective function of the encoder. The feature learned in this way is not simply repeated input. In fact, the role of the sparse penalty term in the hidden layer is to control the number of activated neurons. When the output of the neuron is 0, the neuron is the inactive state. While its output is 1, the neuron is active. Another goal in the feature learning process is to reconstruct the input with fewer active nerves. The reconstruction loss function of SAE is as follows:

$$\begin{aligned} L = {} & \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left(x^{(i)} - z^{(i)}\right)^2\right] + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}^2 \\ & + \beta\sum_{j=1}^{m}\left[\rho\log\frac{\rho}{\rho_j} + (1-\rho)\log\frac{1-\rho}{1-\rho_j}\right] \end{aligned} \tag{3}$$

The loss function consists of three parts; the first parts denotes mean square error between the reconstructed output and the input. And the second part is $L2$ regularization which used to control the over-fitting issue, and w_{ij} denotes the weight. The last part is the sparsity regularization, where ρ denotes the desired target value, and $\rho_j = (1/m)\sum_{i=1}^{m}[y_j(x(i))]$, which mean the average activation output. In the loss function, parameter λ is to prevent overfitting, and β is to control the sparsity penalty.

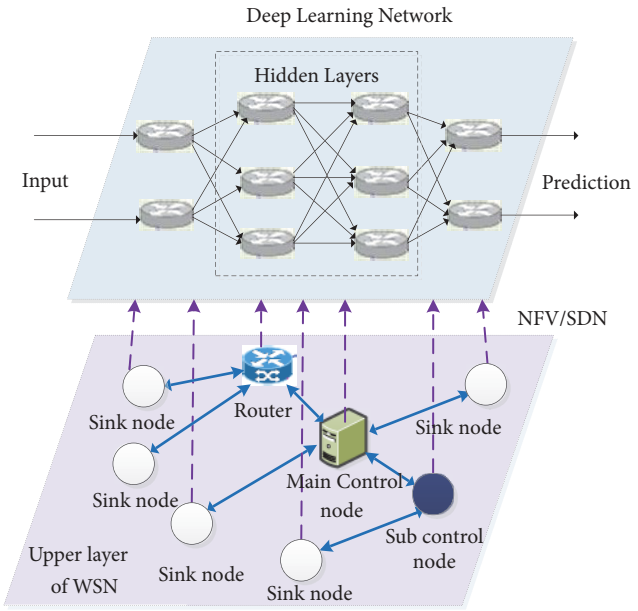

Figure 2: Deep learning network structure in WSNs.

With the greedy layer-wise training and backpropagation algorithm being used to obtain the parameters W and b, the detail iteration process is as follows:

$$w_{ij} = w_{ij} - \varepsilon \frac{\partial}{\partial w_{ij}} L \tag{4}$$

$$b_i = b_i - \varepsilon \frac{\partial}{\partial b_i} L \tag{5}$$

The hyperparameter learning rate ε is used to control the decreasing function of time. After training, SAE would learn the effective sparse feature representations.

3.2.2. Stacked Sparse Auto-Encoder Plus Softmax Classifier. In the WSN system, we will use the SSAE (Stacked Sparse Auto-Encoder) deep learning network due to the limitation of resources. The SSAEs are a deep neural network which consists of multiple layers of basic SAE. The outputs of each SSAEs layer are connected to the inputs of the successive layer. As shown in Figure 3, the SSAEs are composed of two layers SAE, where the first SAE is treated as an encoder, and the second SAE is a decoder. The output y_1 of the first SAE will be the input value of the input layer of the second SAE. When all the SSAE training finished, the model got the features parameter. And all the parameters of SSAEs are utilized to initialize the Softmax classifier to classify the objects.

3.3. Proactive Cache Strategy

3.3.1. Network Parameters of SSAEs. When we construct the SSAEs, some system parameter should be decided in advance, such as the dimension of the input layer, the number of the layers of the hidden layer, the number of neurons in each hidden layer, and the activation function of each layer.

In the SSAEs model, we could use some data groups to make a prediction. It is assumed that there has t width window slide along the time axis. The basic unit widow is one timeslot. In unit windows, all the sink nodes will report corresponding measure data, and those data contain spatial distribute of the WSN sink nodes. If few sink nodes' data missing, we consider the corresponding data is zero. Therefore, if there is more than one timeslot, the measure data contain the spatial-temporal information. If the timeslot window is t, there are $p \times t$ measure data which will provide for the input layers of SSAEs. And the $p \times t$ measure will concatenate as a vector.

Especially, when the window $t > 1$, the prediction of SSAEs will contain the spatial-temporal correlation of data package popularity. The measured data could present as $X^{i-1}, X^{i-2}, \ldots, X^{i-t}$, where i denotes the timeslot and t means the past time. Therefore, we could use the past measure to predict future data package popularity. The input data of the SSAE contain time dependence of content popularity. And the $p \times t$ measure data represent the traffic information of sink nodes in the WNSs.

In this article, it assumes that the content popularity has q levels; therefore the output of the Softmax classifier is also q levels. The output is present as the one-hot vector, and the corresponding level is the level of data package popularity.

Based on the previous description, we do not discuss the changes in the number of sensors in the WSN network. In the SSAEs network model construction, information collection is implemented at the sink nodes rather than directly on the sensor because the sensor nodes may change. In the first case, the sensor node is reduced, which is manifested by the fact that the sensor node loses connection with the WSN sink node for various reasons. This situation is similar to the fact

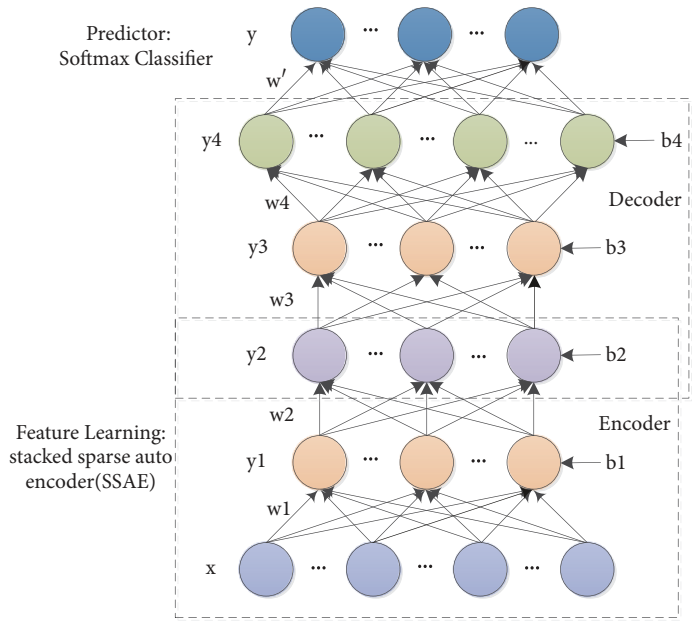

Figure 3: SSAEs structure.

that the data obtained by the sink node from the sensor node is zero, which does not affect the structure of the SSAE network and does not affect the prediction results. In the second case, new sensor nodes are added to the WSN network due to the need to acquire new data. In this case, we can consider a certain amount of node redundancy when initially building the network model. The measured data of these redundant nodes may take the average of the measured values of the WSNs sensors in the corresponding time slots as their values.

3.3.2. WSNs Data Package Popularity Prediction. Briefly, the popularity prediction procedure of SSAEs consist of initialization, training, and running stages.

(i) Initialization Stage. In this stage, the greedy layer-wise training and backpropagation algorithm is used to train the SSAEs model. Therefore, the initialization stage should get the labeling data, including the user request data vector and the corresponding classified information. As mentioned earlier, the user data package of the WSN network is collected by the SDN controller. And the output value is a multi-dimension one-hot vector which denotes the data package popularity in the appointed timeslot in the future.

(ii) Training Stage. In this stage, the pre-training obtains the general parameters, and fine-tuning will optimize those parameters. The greedy layer-wise unsupervised learning algorithm is executed from down to top. Then with the gradient-based optimization technique the BP algorithm is executed from top to down to finetune the SSAEs parameters.

When training is finished, the optimized parameters of SSAEs are fixed.

(iii) Running Stage. In this stage, the SDN controller will dynamically input the user request data information; then the SSAEs will make a prediction. And the data package popularity prediction will send to the SDN controller to generate cache strategy for the WSNs cache nodes.

3.3.3. Proactive Cache Strategy in WSNs. Based on the data package popularity prediction and the WSNs status, the SDN controller will generate the cache strategy.

In the SDN-enabled WSN, SDN controller reserves the topology information of all the WSN nodes and the routing information and is dynamically aware of the cache status of the cache node. The cache status information contains how many cache spaces are available and what is the priority level of the cache of each node. The basic principle of cache strategy is to reserve the high-level content and replace the low-level one. And those contents will be popular in the future time period and not be replaced. The SDN controller will periodically update the strategy when the SSAEs dynamically adjust the data package popularity prediction.

SDN controller generates the cache policies, and cache nodes are only responsible for the execution of the caching strategy. Due to the SDN controller taking the whole topology of WSN network and the cache status of all cache nodes into consideration, the cache strategy will be efficient to avoid the waste of caching resources and reduce the communication overhead between those cached nodes.

In the SDN-enabled WSN, SDN controller reserves the topology information of all the WSN node and the routing information and is dynamically aware of the cache status of the cache node. The cache status information contains how many cache spaces are available and what is priority level of the cache of each node. The basic principle of cache strategy is to reserve the high-level SDN controller

and is not only responsible for the routine work but also responsible for the deployment and maintenance PCDS2AW. When the cache strategy is updated, the SDN controller will update the routing forwarding table for content in the WSNs node and synchronize it to WSN node. When WSN node receives the routing forwarding table, that routing information will actively be inserted into the normal switch routing table before the data packet arrives. The data package will be cached to the assigned space when the high-level data packet arrives. Meanwhile, the low-level data packet will be replaced if the cache space is not enough.

WSN cache node addressing can work normally even if the routing and forwarding tables are missing or incomplete. The input information of the neuron in the PCDS2AW network is incomplete or some nodes are input timeout or the data noise is large. The PCDS2AW can still work properly and can predict the data package popularity of the input user request information. Therefore, PCDS2AW could show good performance of robustness.

Algorithm 1 (the implementation of PCDS2AW for proactive caching for WSN).

Initial: parameters of PCDS2AW

L = 3: Number of hidden layers

Ni = 1200: Number of input dimension

Nh = [300,200,100]: Number of hidden layer dimension

Ts=6: Number of timeslots

Sp = 0.1: Number of sparse parameters

α *= [1.2, 1.5]*: the Zipf law parameter

X_i: generate the data package request data based on $P(X_i = i) = i^{-\alpha} / \sum_{j=1}^{M} j^{-\alpha}$.

W_{li}: generate the weight parameter for each layer of SSAE.

b_i: generate the bias parameter for each layer of SSAE.

Pn = 40000: the number of pre-training epochs;

Fn = 10000; the number of fine-training epochs;

Training:

P = [Pn, Fn]

Repeat:

For Do *L*=1:3

(1) Input X_i,

(2) Calculate y_i based on Formula (1); $X_i = y_i$;

(3) Decode z_i based on Formula (2);

(4) Calculate cost function J based on Formula (3)

(5) Update $W_{li}$and b_i based on Formula (4) and (5).

Pi = Pi -1;

Until (*Pi=0*)

Proactive Cache:

Input X^t: the used request data information form slot *t-i* to *t*.

Calculate the data package popularity *Cp* based on Formula (1) ~ (3).

Sort $C_m = \min[C_{ij}]$, for all cache node N_k;

Replace the $C_m$with *Cp*.

4. Results

4.1. Experimental Environments. In our simulation, we use the TensorFlow framework to construct the SSAEs deep learning network. And the SDN is built on the OpenFlow protocol which separates the control plane of the WSN nodes from the data plane. Our simulation platform runs on Ubuntu 14.04 with 4 GPU cards, NVIDIA GeForce TitanX 12G GDDR5, and 64G RAM memories.

To simulate packet behavior in a WSN network, we assume that the set of packet content throughout the network is $F = f_1, f_2, \ldots, f_R$. These packets are generated by the content server in the WSN network and they are represented by the set $S = s_1, s_2, \ldots, s_p$. For ease of research, we assume that each content packet is randomly generated by only one content server, and each content server is only connected to one network node. At the same time, assuming that all content servers have the same size content unit, the cache space of each cache node of the WSN is the same size, and the cache slot in the cache memory can only accommodate one content unit. In addition, it is assumed that the timing of user packet content requests in the WSN system is subject to the Poisson distribution process. Assuming the user sends a packet request from a fixed virtual node, the overall user packet request conforms to Zipf's law. The frequency at which the user requests the content popularity i $(1 \le i \le M)$ of the data packet is as follows:

$$P(x = i) = \frac{(i^{-\alpha})}{C}, \qquad C = \sum_{j=1}^{M} j^{-\alpha} \tag{6}$$

where M is the total category of the data package content.

In the experiment, the user content request parameters in the WSN network collected by the SDN controller are first constructed as a one-dimensional sequence and then subjected to $[0, 1]$ normalization processing, and, finally, these measurement parameters are input to the input layer of the SSAEs network.

In the active cache emulation, when the user requests the content of the packet, the content matching is first performed in the cache in the corresponding node of the WSN. If the content is found, it indicates a cache hit; otherwise, the cache is not hit. When the user requests that the data packet is missed in the cache, the requested data packet content is traversed throughout the content distribution path of the content server. When the packet requested by the user is grouped, the SSAEs predict the content popularity level based on the measured values in the set slot segment. At the same time, the content popularity prediction result is sent to the SDN controller to generate a new cache policy. The SDN controller synchronizes the cache policy with the WSN cache node data plane through the flow table.

4.2. Evaluation Metrics. In this article, the research goal is full use of the cache resources in the WSN network by reducing the WSN cache node and increasing the storage

Table 1: Architecture structure for SSAEs. Table 1 is reproduced from F. Lei et al. (2018) (under the Creative Commons Attribution License/public domain).

Timeslot	Dimension of input layer	Hidden Layers	Hidden Layers units	MAPE (%)	MAE	RMSE
2	400	3	[300 200 100]	25.73	13.92	24.79
4	800	3	[300 200 100]	23.68	12.89	21.91
6	1200	3	[300 200 100]	22.11	13.95	20.24
8	1600	3	[300 200 100]	24.64	15.94	23.56
10	2000	4	[300 300 200 100]	26.32	16.28	25.73

content difference rate. Therefore, the evaluation criteria of the proactive cache are the cache hit rate, cache route hop count reduction rate. In addition, we also consider the impact of the data packet content popularity Zipf parameters on these evaluation metrics.

The results of the SSAEs prediction model will directly affect the generation of the caching strategy. Therefore, it is necessary to evaluate the SSAEs model performance indicators. We use three performance metrics: root mean square error (RMSE), mean absolute error (MAE), and mean absolute error (MAPE). The specific calculation formula for these indicators is as follows:

$$REMS = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(o_i - p_i)^2} \tag{7}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|o_i - p_i| \tag{8}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}\frac{|o_i - p_i|}{o_i} \tag{9}$$

where o_i is the observed number of data package being requested, p_i is the predicted number of data package content being requested, and N denotes the total number of evaluation samples. The RMSE measures the extremum effect and the error range of the predicted values, and the MAE measures the specificity of the average predicted value. Both of them evaluate the absolute error. MAPE reflects the relative error. When the MAPE is minimized, we consider the model as the optimal structure.

The proactive caching performance metrics is as follows.

CHR (Cache Hit Rate) is a traditional measure of caching performance. The CHR is defined as the ratio of the number of hits requested by the user in the intermediate node to the total number of packets requested by the user.

$$CHR = \frac{CacheHit}{TotalRequest} \tag{10}$$

where *TotalRequest* is the total data package content requests in all WSN nodes and *CacheHit* is total number of data package content request hit the cache nodes. It means the higher the *CHR*, the higher the cache efficiency.

HRR (Hop Reduction Ratio) is the ratio of the number of hops when requesting a data packet hit the cache node to the number of hops from the request data packet to the data source node.

$$HRR(t) = \frac{\sum_{r=1}^{R} h_r(t)}{\sum_{r=1}^{R} H_r(t)} \tag{11}$$

where $H_r(t)$ and $h_r(t)$ mean the hops that user get the request data package from source nodes and cache node in timeslot $[t, t+k]$, respectively. And if there are not cache resources in WSNs, $h_r(t) = 0$, then $HRR = 0$.

4.3. SSAEs Architecture Structure. Before performing content prediction, it is necessary to determine the appropriate SSAE architecture structure parameters, including the input layer dimension, the number of hidden layers, and the number of neurons in each hidden layer. We assume that cache nodes (including sink nodes, router, and control node) are 100 in the WSN network, and sink node is connected to 200 sensors, so 200 measurement data can be collected in each time slot. Therefore, the dimension of the appropriate input data can be determined by studying the number of slots. In the WSNs network, the SDN controller collects the WSN network node to input the user request data to the SSAE network every k time slot. As reported in [6, 10], we set the SSAEs deep learning with 3 hidden layers. We compare the time slot k from the set $\{2, 4, 6, 8, 10\}$, which means that the input dimensions range from 400 to 2000. When the MAPE is minimum, we obtain the optimal SSAEs structure for our prediction system model.

The SSAE network architecture test results are shown in Table 1. As the time slot increases from 2 to 10, the dimension of the input parameters is gradually increased. When the time slot is increased from 2 to 4, that is, the dimension of the input parameter is increased from 400 to 800, the MAPE reduction is significant. When the time slot is 6, the input dimension is 1200, and the MAPE reaches the minimum value, which is about 22.11%. Subsequently, as the time slot increases, that is, the input dimension increases, the MAPE increases. In subsequent experiments, the input dimension of the SSAEs network was set to 1200 and the three-layer hidden layer unit was [300 200 100]. In this process, the number of time slots represents the time-dependent characteristics of the number of inputs. If the number of time slots is too small, the correlation of the data cannot be reflected. If the time slot is too large, additional potential irrelevant inputs will be introduced, making it more difficult for the network architecture to learn a good representation.

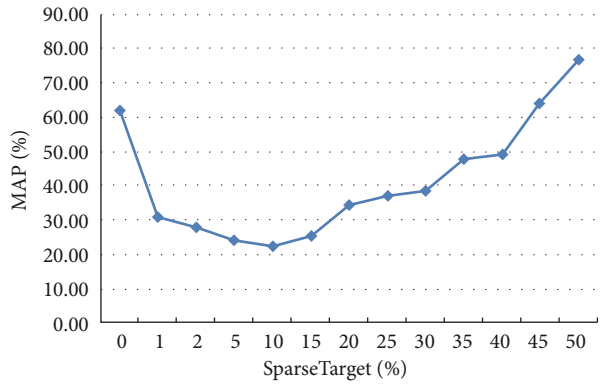

Figure 4: Sparse parameter of SSAEs.

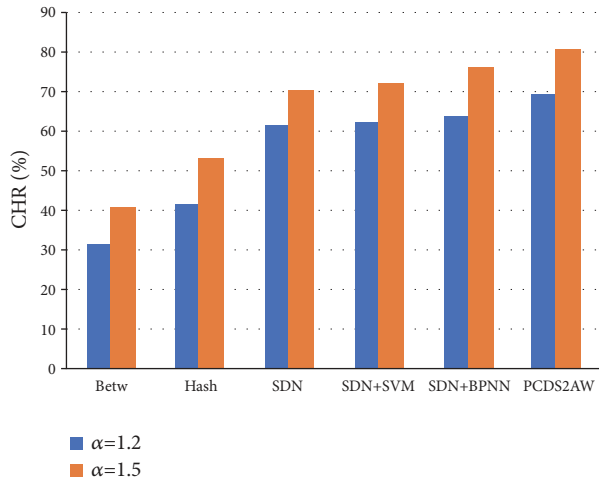

Figure 5: The impact of content popularity on caching.

The effect of sparse parameter of the SSAEs is compared, as shown in Figure 4. When the spare target is 0, which means no sparse target for the SSAEs, the MAP is about 62.18%. With the sparse target increase from 0 to 10%, the MAP is the minimum, 22.11%. With the sparse target increase from 10% to 50%, the MAP also increases to 78.86%. In the SSAEs, the sparse keep as 10%.

4.4. Experimental Results. In the simulation, we also assume that there are 100 cache nodes in the WSN network, 200 measurement data can be collected in each time slot. The other design parameters are as follows: the quantity of users' data package request is set to 120000, the average data package content size is the 1k bit, and the unit size of node cache is the 1k bit. All the following results are the average of 10 rounds simulation.

We study changes in network performance due to PCDS2AW prediction and generalization capabilities. Firstly, we compare PCDS2AW with traditional classic caching strategies that do not support SDN, such as Betw + LRU, Hash + LRU, and Opportunistic. At the same time, we also compare PCDS2AW with caching strategies that support SDN, such as SDN + BPNN (Back Propagation Network), SDN + SVM (Support Vector Machine). BPNN is a classical neural network that learns features through hidden layers. SVM is a widely used classic prediction model. In these comparative experiments, we used the same training set and test set as PCDS2AW to train and test these models.

Figure 5 shows the CHR (cache hit ratio) comparison of Betw, Hash, SDN, SDN+SVM, SDN+BPNN and PCDS2AW schemes when alpha =1.2 and alpha =1.5 are used. As the alpha value increases, the data distribution requested by the user is more concentrated, and the probability of the data packet hitting the cache is also improved, which shows that the CHR in all schemes is correspondingly improved. For Hash-LRU, all the nodes in the domain collaborate and realize the cooperative caching of the response content. Although the content cannot be optimized, the hit rate of the cache content is enhanced, and the cache hit rate increases by 27.5%. As for the Betw-LRU, the cache hit rate increases by 29.1% with the user's content request more centralized. In the SDN, because SDN's network perception function further makes full use of the storage space of all caching nodes, increasing the probability and utilization of caching content, CHR reaches 61.4% and 70.3%, respectively. And with the SVM to predict the popularity, the CHR of SDN+SVM is higher 1.1% and 2.1%

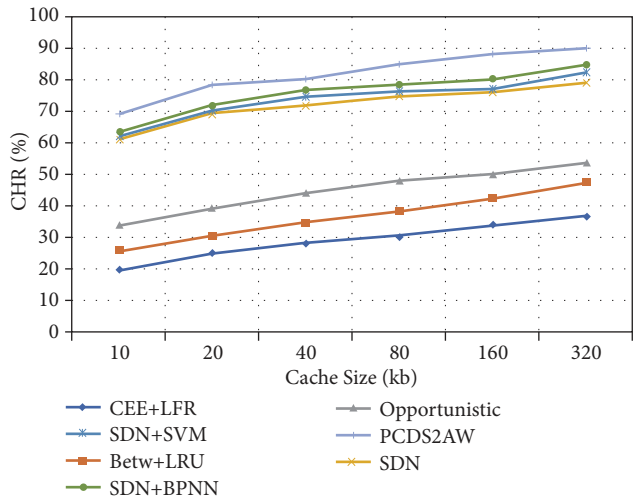

Figure 6: Effect of cache size.

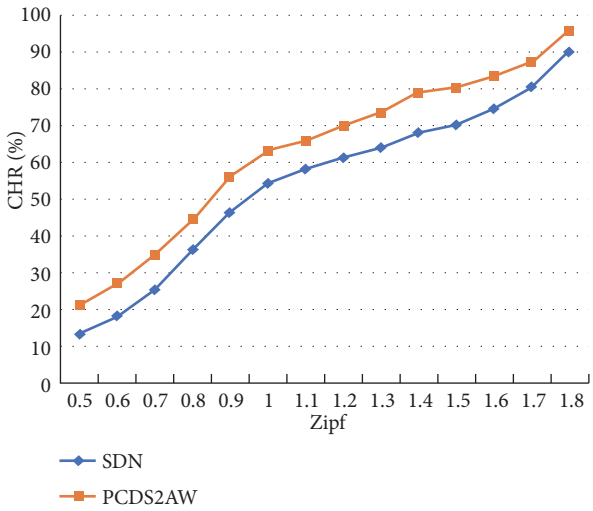

Figure 7: Effect of Zipf parameter α.

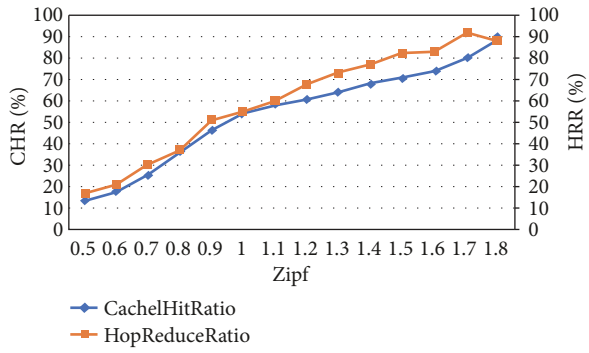

Figure 8: Effect of Zipf with CHR and HRR.

than SDN. The SDN+BPNN cache hit rate reach 64.2% and 76.4%. The PCDS2AW achieves the best than others; CHR reaches 69.6% and 80.3%, respectively.

Figure 6 shows the effect of cache size on the cache hit ratio. As the size of the cache space of the cache node increases, the cache hit ratio in all cache schemes also increases. This is because as the cache storage space increases, the cache node can cache more data content packets, so there is a higher probability of buffering the data packets requested by the user. As the cache space increased from 10kb to 320kb, the cache hit rate for CEE increased from 19.9% to 32.5%; the cache hit rate of the Betw scheme increased from 25.9% to 47.5%; Opportunistic increased from 34.1% to 53.4%; the cache hit rate of SDN increased from 61.4% to 79.1%; CHR of SDN+SVM increased from 62.3% to 82.4%, the SDN+BPNN increased from 63.7% to 84.7%, and the PCDS2AW increased from 69.3% to 90.1%.

Figure 7 shows the influence of the content popularity parameter α on the cache hit rate in the Zipf distribution. With the increase of content popularity, the concentration of requests for user content is increasing. In the case of cache space, the probability of the content in the cache is increased and the cache hit rate is increased. In Figure 6, cache popularity alpha increased by 1.8 from 0.5; the cache hit rate increased from 13.30% to 89.98% for SDN cache scheme and from 21.5% to 95.7% for PCDS2AW.

Figure 8 shows the CHR and HRR relation of PCDS2AW with the Zipf distribution. With Zipf increase, the CHR and HRR also increase. When cache popularity alpha equals 1.7, the HRR reaches the maximum about 91.88%.

Computational complexity: The PCDS2AW algorithm is mainly divided into two stages: offline feature model training and online content popularity prediction. In the offline phase, SSAEs feature training is mainly used to obtain model parameters. The online stage directly predicts the input value input into the model for content popularity prediction. The prediction model SSAE in this paper is three layers [300, 200, 100], and the parameter dimension of the input layer is 1200. The sparse parameter in our model is 0.10. Because multiplication is the most time-consuming, we only estimate the number of the multiplications of the online prediction in our model.

The number of multiplications in the prediction process is (1200∗300+300∗200+200∗100)∗0.10=44000. In our simulation, the predicted time was 0.0435 s. Therefore, the model can meet the needs of online prediction.

5. Conclusions

This paper presents a simple frame structure of SDN/NFV coupled with SSAEs to address the challenges of WSNs particularly in the issues of traffic loads and congestion management. We construct the distributed deep learning network, SSAEs by SDN/NFV technical. After detail analyzes the architecture parameters, the unsupervised training method is used to obtain the prediction model. The experiments show that the SSAEs can greatly improve the prediction accuracy, and then the performance of WSN can be improved based on proactive caching.

In the future, we plan to combine the SSAEs with Incremental Extreme Learning Machine (IELM) to online training and update the SSAE network parameter, while the user request dynamic changes to get data package prediction and continues to optimize our model to get better performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China [U170120078, 61571141, 61702120, and 61672008], the Scientific and Technological Projects of Guangdong Province [2017A050501039], the Guangdong Provincial Key Laboratory Project [2018B030322016], the Guangdong Provincial Application-Oriented Technical Research and Development Special Fund Project [2015B010131017, 2015B090923001, 2016B010127006, and 2017B010125003], the Guangdong Province General Colleges and Universities Featured Innovation [2015GXJK080], and the Qingyuan Science and Technology Plan Project [170809111721249 and 170802171710591].

References

[1] Cisco VNI, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021*, 2017.

[2] B. Raghavan, T. Koponen, A. Ghodsi, M. Casado, S. Ratnasamy, and S. Shenker, "Software-defined internet architecture:decoupling architecture from infrastructure," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets '12)*, 2012.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[4] M. Ojo, D. Adami, and S. Giordano, "A SDN-IoT architecture with NFV implementation," in *Proceedings of the GLOBECOM Workshops*, 2016.

[5] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.

[6] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: a deep learning approach with SDN," *IEEE Access*, vol. 99, p. 1, 2017.

[7] N. Abani, T. Braun, and M. Gerla, "Proactive caching with mobility prediction under uncertainty in information-centric networks," in *Proceedings of the 4th ACM Conference on Information-Centric Networking, ICN '17*, ACM, 2017.

[8] Y. L. Cun, B. Boser, J. S. Denker et al., "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems*, vol. 2, no. 2, pp. 396–404, 1990.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2012.

[10] F. Lei, Q. Dai, J. Cai et al., "A proactive caching strategy based on deep Learning in EPC of 5G," in *Proceedings of the International Conference on Brain Inspired Cognitive Systems*, Springer, 2018.

[11] G. Wu, J. Han, and Y. Guo, "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

[12] G. Wu, J. Han, and Z. Lin, "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, 2018.

[13] G. Ding, Y. Guo, K. Chen, C. Chu, J. Han, and Q. Dai, "DECODE: deep confidence network for robust image classification," *IEEE Transactions on Image Processing*, 2019.

[14] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[15] K. Suksomboon, S. Tarnoi, J. Yusheng et al., "PopCache: Cache more or less based on content popularity for information-centric networking," in *Proceedings of the 38th Annual IEEE Conference on Local Computer Networks (LCN '13)*, pp. 236–243, 2013.

[16] X. Hu and J. Gong, "Opportunistic on-path caching for named data networking," *IEICE Transactions on Communications*, vol. E97B, no. 11, pp. 2360–2367, 2014.

[17] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor openflow: enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[18] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proceedings of the International Conference on Computer Science and Network Technology*, 2012.

[19] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[20] W. Dong, Y. Liu, C. Chen, L. Gu, and X. Wu, "Elon: enabling efficient and long-term reprogramming for wireless sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 4, pp. 1–27, 2014.

[21] L. Jiang, L. Yan, Y. Xia, Q. Guo, M. Fu, and L. Li, "Distributed fusion in wireless sensor networks based on a novel event-triggered strategy," *Journal of The Franklin Institute*, 2018.

[22] B. T. De Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Proceedings of the Communications*, 2014.

[23] T. Miyazaki, S. Yamaguchi, K. Kobayashi et al., "A software defined wireless sensor network," in *Proceedings of the 2014 International Conference on Computing, Networking and Communications (ICNC '14)*, 2014.

[24] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Proceedings of the Communications*, 2014.

[25] P. Jayashree and F. Infant Princy, "Leveraging SDN to conserve energy in WSN-An analysis," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking, ICSCN '15*, 2015.

[26] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "UbiFlow: Mobility management in urban-scale software defined IoT," in *Proceedings of the IEEE Conference on Computer Communications*, 2015.

[27] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 513–521, 2015.

[28] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: unbridling SDNs," in *Proceedings of the 2012 European Workshop on Software Defined Networking*, pp. 1–6, 2012.

[29] B. R. Al-Kaseem and H. S. Al-Raweshidyhamed, "SD-NFV as an energy efficient approach for M2M networks using cloud-based 6LoWPAN Testbed," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1787–1797, 2017.

[30] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *Proceedings of the IEEE International Conference on Communications*, 2014.

[31] A. Mahmood, C. Casetti, C. F. Chiasserini et al., "Mobility-aware edge caching for connected cars," in *Proceedings of the Conference on Wireless On-Demand Network Systems and Services*, 2015.

[32] C. Jayapal, S. Jayavel, and V. P. Sumathi, "Enhanced service discovery protocol for MANET by effective cache management," *Wireless Personal Communications*, pp. 1–17, 2018.

[33] M. I. Alipio and N. M. C. Tiglao, "Dynamic source rate control for cache-based transport protocol in wireless sensor networks," *Computer Communications*, vol. 113, pp. 14–24, 2017.

[34] M. Alipio, N. M. Tiglao, A. Grilo, F. Bokhari, U. Chaudhry, and S. Qureshi, "Cache-based transport protocols in wireless sensor networks: a survey and future directions," *Journal of Network and Computer Applications*, vol. 88, pp. 29–49, 2017.

[35] C. Panagiotou, C. Antonopoulos, and S. Koubias, "A comprehensive evaluation of cache utilization characteristics in large scale WSN considering network driven cache replacement techniques," in *Proceedings of the MATEC Web of Conferences*, EDP Sciences, 2018.

[36] R. Sukjaimuk, Q. N. Nguyen, and T. Sato, "A smart congestion control mechanism for the green IoT sensor-enabled information-centric networking," *Sensors*, 2018.

[37] T. Trzcinski and P. Rokita, "Predicting popularity of online videos using support vector regression," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2561–2570, 2017.

[38] Y. Mao, Y. Shen, G. Qin, and L. Cai, "Predicting the Popularity of Online Videos via Deep Neural Networks," https://arxiv.org/abs/1711.10718, CoRR, 2017.

[39] W. Stokowiec, T. Trzciński, K. Wołk, K. Marasek, and P. Rokita, "Shallow reading with deep learning: predicting popularity of online content using only its title," in *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, 2017.

[40] H. Cai, Y. Zhang, Y. Wang et al., "Predicting relative popularity via an end-to-end multi-modality model," in *Proceedings of the International Forum on Digital TV and Wireless Multimedia Communications*, 2017.

[41] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: the role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[42] E. Bastug, M. Bennis, E. Zeydan et al., "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.

[43] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An intelligent traffic load prediction based adaptive channel assignment algorithm in SDN-IoT: a deep learning approach," *IEEE Internet of Things Journal*, p. 1, 2018.

[44] G. Yang, Y. Wang, H. Yu, Y. Ren, and J. Xie, "Short-term traffic state prediction based on the spatiotemporal features of critical road sections," *Sensors*, vol. 18, no. 7, p. 2287, 2018.