Data Caching at Fog Nodes Under IoT Networks: Review of Machine Learning Approaches

Riya Tapwal [1], Nitin Gupta [1], and Qin Xin [1]

[1]Affiliation not available

October 30, 2023

Abstract

IoT devices (wireless sensors, actuators, computer devices) produce large volume and variety of data and the data produced by the IoT devices are transient. In order to overcome the problem of traditional IoT architecture where data is sent to the cloud for processing, an emerging technology known as fog computing is proposed recently. Fog computing brings storage, computing and control near to the end devices. Fog computing complements the cloud and provide services to the IoT devices. Hence, data used by the IoT devices must be cached at the fog nodes in order to reduce the bandwidth utilization and latency. This chapter discusses the utility of data caching at the fog nodes. Further, various machine learning techniques can be used to reduce the latency by caching the data near to the IoT devices by predicting their future demands. Therefore, this chapter also discusses various machine learning techniques that can be used to extract the accurate data and predict future requests of IoT devices.

Chapter 1

Data Caching at Fog Nodes Under IoT Networks: Review of Machine Learning Approaches

RIYA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, NATIONAL INSTITUTE OF TECHNOLOGY, HAMIRPUR, HIMACHAL PRADESH, INDIA

NITIN GUPTA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, NATIONAL INSTITUTE OF TECHNOLOGY, HAMIRPUR, HIMACHAL PRADESH, INDIA

QIN XIN
FACULTY OF SCIENCE AND TECHNOLOGY, UNIVERSITY OF THE FAROE ISLANDS
VESTARABRYGGJA 15, FO 100 TORSHAVN, FAROE ISLANDS

IoT devices (wireless sensors, actuators, computer devices) produce large volume and variety of data and the data produced by the IoT devices are transient. In order to overcome the problem of traditional IoT architecture where data is sent to the cloud for processing, an emerging technology known as fog computing is proposed recently. Fog computing brings storage, computing and control near to the end devices. Fog computing complements the cloud and provide services to the IoT devices. Hence, data used by the IoT devices must be cached at the fog nodes in order to reduce the bandwidth utilization and latency. This chapter discusses the utility of data caching at the fog nodes. Further, various machine learning techniques can be used to reduce the latency by caching the data near to the IoT devices by predicting their future demands. Therefore, this chapter also discusses various machine learning techniques that can be used to extract the accurate data and predict future requests of IoT devices.

Keywords: Fog Computing, Caching, Machine Learning, Cloud Computing, IoT

1.1 Introduction

In the recent years, small devices embedded with sensors produce large amount of data by sensing the real time information from the environment. The network of these devices communicating with each other is recognized as IoT (Internet of Things) sometimes called as Internet of Everything [1]. The data produced by the IoT devices need to be delivered to the users using IoT applications after processing and analyzing. Further, data produced by the IoT devices is transient which means that generated data has certain lifetime and after that lifetime the data become useless and hence discarded [2]. Therefore, it is required to store the data somewhere near to the IoT devices [3]. At the same time, if data produced by the IoT devices is stored at the cloud server then it adds communication overhead, as the IoT users need to contact to the cloud server whenever they require any data.

Fog Computing is a decentralized approach to bring the advantages and intelligence of cloud computing such as storage, applications and computing services near to the end devices somewhere between the cloud and the end devices [4, 5]. Fog nodes can be anything such as servers, networking devices (routers and gateways), cloudlets and base stations. These nodes are aware of their geographical distribution as well as the logical location in the cluster. They can operate in centralized or in distributed manner and can also act as stand-alone device. These nodes receive inputs from the data generators (IoT devices), process it and provide transient storage to the data. Fog nodes are intelligent devices which also decide that what data to store and what to send at the cloud for historical analysis. These devices can be either software or hardware, arranged in a hierarchy and are used for filtering of data send by the sensors devices. These devices should have less latency, high response time, optimal bandwidth, optimal storage and decision making capability. At the fog nodes, intelligent algorithms are embedded for the process of storing of data, computing and forwarding of data between various layers. The member function of fog node in fog-cloud network is depicted in figure 1.1. In this figure, the *compute* module is responsible for the processing of data and computing the desired result. The *storage* module is responsible for storing data reliably so that the robustness can be achieved. Further, various *accelerator* units such as Digital Signal Processors, Graphics Processing Units etc. are used in critical tasks in order to provide additional power whereas the *network* module is responsible for the guaranteed delivery of data.

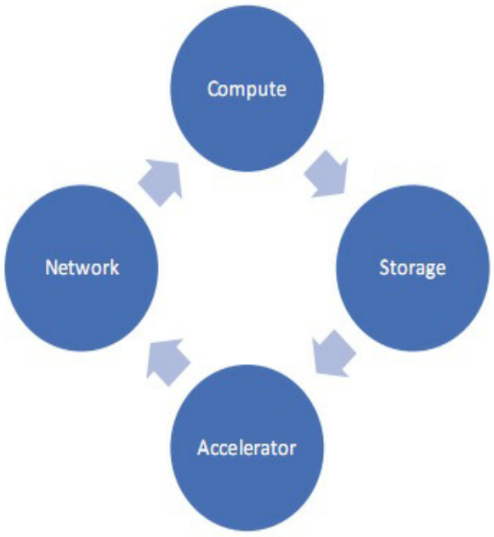

Figure 1.1: Functions of Fog Nodes

Fog computing only complements the cloud computing by providing short term analytics unlike cloud computing which provide long term analytics. However, it is to be mentioned that fog computing does not replace the cloud computing [6]. There are majorly six characteristics which differentiate fog computing from other computing paradigms [7, 8].

a) **Awareness and Low Latency:** Fog nodes are aware of their logical location in the context of whole system and offer a very low latency and cost for the communication. Fog nodes are frequently placed near to the edge devices, and hence they are able to return reply and other analysis much faster than the cloud nodes.

b) **Heterogeneity:** Fog nodes generally collect different form of data and from different types of devices through different types of networks.

c) **Adaptive:** In many situations, fog computing deals with uncertain load patterns of various requests submitted by the different IoT applications. Adaptive and scaling features of fog computing help it to deal with the above mentioned scenario.

d) **Real Time Interaction:** Unlike cloud computing, which support batch processing, fog computing support real time interaction. The real time data which is time sensitive, is processed and stored at the fog nodes and sent back to the users whenever required. Whereas, the data which is not time sensitive and whose life cycle is long, is sent to the cloud for processing.

e) **Interoperability:** Since, fog computing support real time interaction therefore, it require the co-operation of various providers leads to interoperable property of fog computing .

f) **Geographically Distributed:** Unlike centralized cloud the applications serviced by fog nodes are geographically distributed like delivering seamless quality videos to the moving vehicles.

Further, the processing time of fog nodes is very less (millisecond to subsecond). This technique avoids the need of costly bandwidth and helps the cloud by handling the transient data. In order to facilitate the fog computing, the node should exhibit autonomy (property to take decision independently without the intervention of other nodes), heterogeneity, manageability and programmability. Figure 1.2 shows the architecture of fog computing where IoT devices are connected to the fog nodes and then fog nodes are further connected to the cloud nodes [9].

The architecture of fog computing consists of three layers [10]:

a) **Terminal layer:** This is the lower most layer and consists of the IoT devices such as mobile phones, sensors etc. which detect the information from the environment by sensing it and then transmit the detected information to the upper layer. The information is transmitted in the form of data streams. The IoT data streams are the sequence of values which are emitted by the IoT devices or may be produced by one application module for other application module and send to the higher layer for processing.

b) **Fog layer:** This layer consist of various switches, portals, base stations, specific servers etc. This layer lies between the IoT devices and the cloud and is used for the processing of data near to the IoT devices. If fog nodes are not able to fulfill the request of the terminal layer then the request is forwarded to the cloud layer.

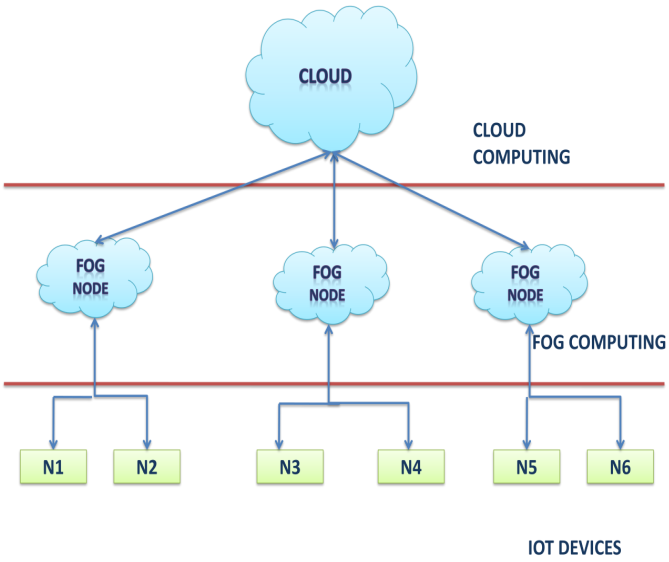

Figure 1.2: An Architecture of Fog Computing

c) **Cloud layer:** This layer consists of high performance servers which are used for the storage of data as well as performing powerful computing.

Generally, IoT devices do not have processing power and storage, due to which they suffer from many problems such as performance, reliability and security [11]. The fog nodes are capable of performing the operations which require large amount of resources on the behalf of IoT devices which are generally resource constrained devices. This makes end-devices less complex and also reduces the power consumption. Further, fog computing also supports real time interactions between the IoT devices and the fog nodes as the data is available to the IoT devices quickly unlike the cloud computing where batch processing is mostly used. Further, IOT devices are resource constrained and generally do not have security features for which fog nodes act like the proxy servers and provide extra security features. Fog nodes regularly update the software and security credentials and check the safety status of these devices.

Fog computing also offers the implementation of various service models such as Software as a Service (SaaS), Platform as service(PaaS) and Infrastructure as a service(IaaS) [12, 13]. Due to such advantages, various frameworks such as Google App Engine, Microsoft Azure and Amazon Web Services which have been using cloud computing, have also started supporting the fog computing for providing solutions to the develop distributed applications which are geographically dispersed and require low latency computational resources. They are also using dedicated nodes with low latency computational power also called as mist nodes(lightweight fog nodes) and are sometimes placed more closer to the IoT devices than the fog nodes [14, 15]. Hence, the integration of IoT with fog computing brings many such advantages.

1.1.1 Importance of Caching at the Fog Nodes

The IoT devices do not have to contact to the remote server i.e. cloud every time when they require some data. The IoT devices first check data in the cache of the fog nodes. If required data is present then the fog nodes return the data to the IoT devices, otherwise they contact the cloud for the required data. Hence, caching of data at the

fog nodes reduces the transactional latency. Moreover, fog computing requires lesser bandwidth to transfer the data [16]. As fog computing supports hierarchical processing, the amount of data required to be transferred from the IoT devices to the clouds is less, whereas, the amount of data transferred per unit of time from the fog node to the the IoT devices is more, which leads to the improvement in overall throughput. Hence, caching data at the fog nodes decreases the overall operational expenses. Data is stored in the distributed manner at the fog nodes which can be deployed anywhere according to the requirements. Further, caching of data at the fog nodes helps in the reduction of load at the cloud servers as the data whose frequency of interest is more among IoT devices and the probability of reusing the same data is also high is cached at the fog nodes. Hence, only selected data is transferred for storage and processing to the cloud which reduces the latency of contacting the remote server which is far away from the IoT devices/sensors. Further, storing of data at the fog nodes ensures continuous services to the IoT devices irrespective of irregular network connectivity.

Along with the advantages there are some challenges that need to be addressed in order to cache data at the fog nodes. The biggest challenge of this technique is to decide what to store at the cloud and what to cache at the fog nodes. The decision to cache the data at the fog node should be taken in such a way that the hit rate of data at the fog node should be maximized such that the overall throughput is maximized [17, 18]. Further, storage capacity of the fog nodes is limited and they can only store the selected data. Therefore, it is necessary to predict the future demand of the users such that the data frequently required by the users in the future can be stored at the fog node to maximize the hit rate. However, it is difficult to predict the future requirement of the users.

Another challange which need to be addressed is to maintain synchronization between the data cached at the fog node or at different fog nodes and data at the cloud nodes. Further, the security of data at the fog nodes and to select the ideal fog node is also an issue of concern [19]. Further, mobility of nodes or virtual machines which requires for maintenance, balancing and power management is also a challenge which need to be addressed. Each fog node may have one or more virtual machines depending upon requests and traffic conditions. The computation and communication required for the process of hand-off and its effect on caching is very much complicated and expensive [20, 21].

As discussed above, this chapter focuses on important aspect of caching, which is to predict the future demands of the IoT users such that effective caching of data can be done at the fog nodes. In order to address this problem, various machine learning techniques are discussed which helps in learning the behavior and pattern of demands of IoT devices and also add auto processing and auto computing capability to the fog nodes. Before exploring the machine learning techniques, in the next section various applications of caching at fog nodes and the life cycle of fog data are discussed.

1.2 Applications Of Data Caching at Fog Nodes for IoT Devices

In this section, some of the real scenarios are discussed where data caching at fog nodes can be very useful [22–29].

a) **Dynamic Content Delivery and Video Streaming:** With the increase in multimedia contents, the conventional network suffer from congestion. Further, video traffic acquire half of the traffic and frames to play the video are required at faster rate such that there is no interruption. Hence caching of data at the fog nodes is suitable approach for faster delivery of the multimedia contents.

b) **Virtual Reality and Online Gaming:** Virtual Reality and online gaming require real time data. In virtual reality it is required to provide the status of the user as well as the location of the users. Hence, it is required to process the data and provide data to the user as soon as possible where fog computing seems to be the promising approach for this purpose.

c) **Smart Cities:** In the smart cities, various IOTs are connected together to share data with each other. These IoTs generate large amount of data which need to be processed near to the IoTs. For example in case of smart traffic lights, data can be stored and processed at fog nodes and used to send warning signals to the approaching vehicles.

d) **Smart Grids:** The data generated by the smart grids contain complex parameters which are hard to analyze. Fog nodes have the power to analyze, process the complex data to perform heavy computations. Hence, fog nodes can be used in order to store and process the local data generated by the smart grids and various IoT devices used in the smart cities.

e) **Smart Health-care:** Real time data processing make smart health-care more efficient and faster. Hence fog computing can be used in health-care in order to make their working more efficient. For example fog computing may be used to detect falling of the stroke patients.

f) **Intensive Computation Systems:** The systems which require intensive computations require low processing and latency time. Hence the data produced by these systems must be processed and stored at the fog nodes and provided to the systems whenever required.

g) **Internet of Vehicles:** Fog Computing plays important role in vehicle to vehicle communication and taking safety measures on the road by providing data to the vehicles which is required to take decisions for traffic control and smart parking. Fog nodes obtain data from the sensors deployed and take decisions for traffic control measures.

h) **Wireless Sensors Systems:** The data produced by wireless sensors system such as oil and gas industries, chemical factories is transient which need to be stored near the users. Hence the data produced by these systems should be cached at the fog nodes in order to improve the performance of the systems [30].

In all of aforementioned scenarios it is suitable to store the real time or dynamic content near to the users that are generating the data and also may require it in near future. This requirement can be easily fulfilled by caching the data at the fog nodes located near to the users or IoT devices.

1.3 Life cycle of Fog Data

As discussed in the introduction secion, depending upon the various layers in fog computing, fog data goes through various steps from acquiring data at the terminal layer to the processing of data and the execution of tasks to constitute a life cycle. The life cycle of the data is shown in figure 1.3 and various steps involved during the life cycle are explained as follows [31–36]:

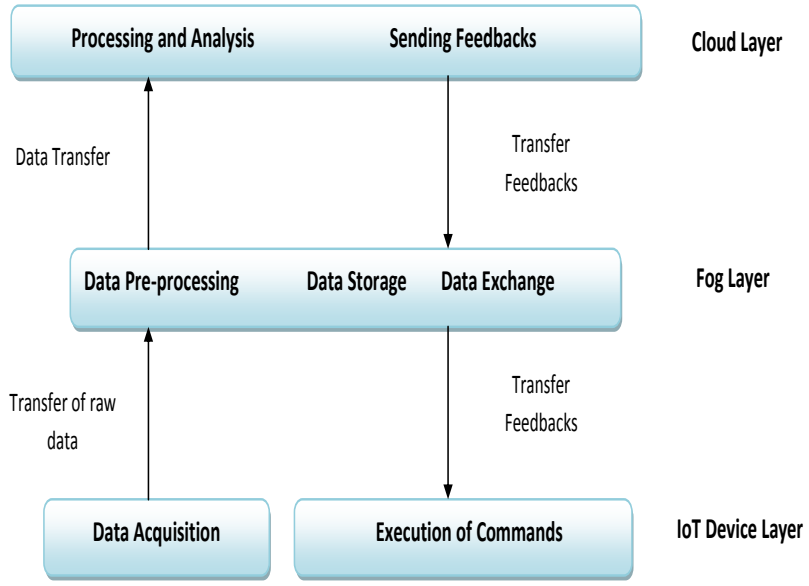

Figure 1.3: Life cycle of Fog Data

a) **Data Acquisition:** The sensors present in the device layer/ terminal layer sense the environment and collect data. The acquired data is either sent to the sink node or directly transferred to the fog node for processing.

b) **Lightweight Processing:** Lightweight processing is done at the fog layer and hence include various tasks such as filtering of data, cleaning of data, eliminating the unwanted data, lightweight manipulation of data, compression/decompression of data, encryption/decryption of data. Some data is stored at this layer in order to support real time processing and rest of the data is transferred to the cloud layer for further processing. Further, the feedbacks and the data is exchanged by the fog layer as shown in figure 1.3

c) **Processing And Analysis:** The data received from the fog layer is processed by using different types of analysis in order to extract the important data. The data is permanently stored at the cloud server. According to the processing performed at the data received from the fog layer, reports are generated. Various technologies such as map reduce is used for data processing at the cloud.

d) **Sending Feedback:** On the basis of reports generated during the process of data processing, cloud server send feedback such as data required by the end devices, proper commands to the the device layer in order to perform required action.

e) **Command Execution:** Based on the feedbacks received from the cloud server, the actuators perform the respective action and then required actions are performed on the environment.

It is evident from the above sections that caching played a major role in the fog computing. Efficient caching will be helpful in achieving low latency requirement, and to maintain high QoS and QoE of 5G. Caching is classified as reactive caching where data caching is done on request and proactive caching where pre-fetching

of data is done. To achieve higher spectrum efficiency proactive caching is better if prediction errors are nearly zero [37]. Therefore, it is important to design various techniques to predict the future requests of the users, which can be cached at the fog nodes such that repetitive requests to the cloud can be avoided.

In the literature, various techniques have been used for data prediction and caching like fog to fog (F2F) caching [38] where multi-agent cooperation is used. Authors in [39] proposed location customized regression based caching algorithm to predict the future content demands. Authors in [40] distinguished requests on three different popularity levels and then strategically cached data at the fog nodes according to various activity levels. Apart from caching at the fog nodes, Device to Device (D2D) caching has also been done in the fog computing environment where direct communication between the nodes (IoT devices) takes place at a short distance without any infrastructure [41, 42]. Whenever data is required by the device, it checks its local cache for the data. If data is not available, it broadcasts the request to the other devices. The other IoT devices present at the ground tier in hierarchy check for the data. If the data is present then the respective device replies with the data to the requesting device otherwise it replies with the negative acknowledgement. Then requesting device requests for the data at the fog servers. As stated earlier, if data is not available at the fog server then it sends the request to the cloud server. Cloud server in return sends the data to the fog server and then fog server sends data to the requesting nodes. Figure 1.4 shows the interaction between various IoT devices as well as the interaction between IoT devices and fog nodes.

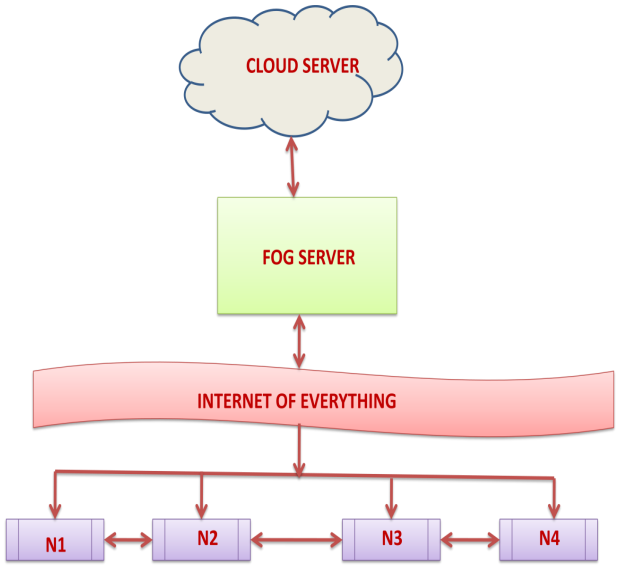

Figure 1.4: Caching Using D2D Technique

As mentioned before, the problem of content placement relies on the prediction accuracy of the user requirement, popularity of content and caching strategy design. To predict the data content demand, large amount of available data related to similar interests, social and geographic data and history data of users can be used for better prediction of user demand [43]. This is effectively implemented using machine learning schemes. In the following section, various machine learning techniques used for data caching at the fog nodes are investigated.

1.4 Machine Learning for Data Caching and Replacement

Sr. No.	Techniques	Description
1	Residual Nets [44]	In this method in order to reduce the difficulty of training models, shortcut connections are introduced into the convolutional Neural Networks. Visual inputs are mainly focused in residual nets.
2	Long Term Recurrent Convulational Network [45]	In this method, convolutinal neural networks are applied in order to extract the features. In video, frame sequences are combined with the long short term memory [46]. Further, the spatial and temporal relationships are exploited between the inputs.
3	Restricted Boltzman Machine [47]	The performance of human activities are improved by deep Boltzman Machine. In case of multi-restricted Boltzman Machine, the performance is improved by multi-nodal DBM [48]
4	IDNET [49]	Convolutinal Neural Networkis applied by the IDNET for biometric analysis tasks
5	DeepX [50] and RedEye [51]	In these methods,on the basis of hardware and software, the energy consumption of deep neural network is reduced.

Table 1.1: Machine Learning methods used at terminal layer for Data Sensing

Caching at the fog nodes is influenced by various factors like varying interest of different IoT users which changes with different contexts, changed locations, network topology, and so on. Therefore, future content request is highly unknown before making any caching decision [52]. Machine learning based algorithms enable each fog node having limited storage to make the right decision in selecting right contents to cache such that the caching performance of the fog node is maximized. Machine learning is used for predicting the user demands and mapping users input to the output actions. Machine learning is a promising approach which is used for improving the network efficiency by predicting user's demand and is used for discovery of early knowledge from large data streams [43]. In machine learning approach, large amount of data is exploited in order to determine the content popularity and also useful for filtering the data and knowledge [53–56]. The further processing of this data is helful in order to analyze correlation between the features and respective output of the data [57]. Further, machine learning techniques can be categorized into two types: Unsupervised learning and Supervised learning. In supervised learning, learning systems are provided with learning algorithms with known quantities which help these algorithms for making future judgments. Whereas in unsupervised learning the learning system is provided with unlabeled data, and algorithm is allowed to act upon it without any guidance. Machine learning can be used at any layer of fog computing i.e at terminal layer, at fog layer or at the cloud. At terminal layer, machine learning is used for data sensing. There are various methods used for the sensing of data and are described in table 1.1. The complex features of datasets like videos, vibrations, various modeled readings from the IoT devices can be recognized by the machine learning methods [58–63]. Various machine learning algorithms such as recurrent neural network(RNN), convolutional neural network(CNN), generative adversarial network(GAN) etc. have been

used recently [62].

At fog layer machine learning is used for data storage and resource management [64]. Using machine learning algorithms, data is sampled from the IoT devices, compressed and aggregated at the fog nodes for further processing. Figure 1.5 shows the data analysis methods for the data produced by the IoT devices and various machine learning techniques that can be used in order to analyze the data and then decide that what to cache at fog nodes.

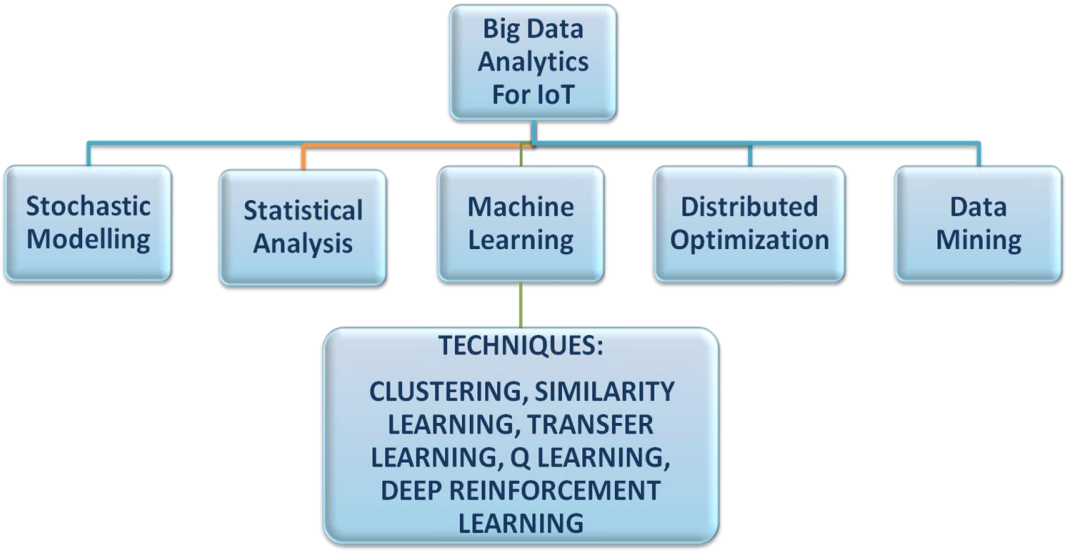

Figure 1.5: Big Data Analytics for wireless network

Various machine learning techniques which are used by the researchers in the literature for data caching at fog nodes are as follows:

1. **Clustering of fog severs:** Clustering is a technique used in unsupervised learning in which the information is not guided. In this technique, the fog severs are clustered in order to fulfill the demands of IoT devices [23]. Data is stored at various fog servers after being coded into segments. When the user raises the content request, it is served by the group of fog servers which are clustered on the basis of content stored in them. If the requested content is cached at the fog servers then the IoT device fetches the data from the fog servers in ascending order of transmission distance until the obtained segments are sufficient for decoding. Further, if the obtained segments are not sufficient then the nearest fog server contacts the cloud for the data and fetches the remaining data from the cloud and deliver it to the IoT device. Further, cluster size influences the efficiency of the system as the benefit of cooperative caching is vanished if the size of the cluster is very large but at the same time IoT devices can fetch data from the various nodes hence increases the cache diversity. Therefore, cluster size should be optimal which balances the trade-off.

2. **Similarity Learning Approach:** In this approach, the fog nodes are given with the pair of similar IoT devices as well as the pair of less similar devices. From the given set of IoT devices, the intelligent fog node finds the similarity function(or the distance metric function) between the pair of similar devices by learning about their various features [43]. In this technique, two parameters i.e. common interest and physical relations (link quality)

are considered in order to find the similarity between the IoT devices. One to one matching scheme is also used for the pairing of the IoT devices. With the help of this function, the intelligent fog node finds whether the new device is similar or not and hence find the future interest of new device whose interests are unknown.

3. **Transfer Learning Approach:** In this approach the intelligent fog node gain the knowledge while solving the problem and store it. It then solves the new problem by exploiting the knowledge obtained from the data of existing tasks [65]. The fog node makes use of two domains for training: source domain and target domain. Source domain refers to the knowledge obtained from the interactions of the IoT devices while target domain refers to the pattern of requests made by the IoT devices. The source domain consists of IoT devices and the data requested by the devices in the past. Additionally, source domain contain popularity matrix(number of times the particular data is requested by the IoT devices). This technique smartly borrows the user-data relationship from source domain in order to understand the target domain by establishing correspondence between both. The correspondence is established in order to identify the similarly rated content in both the domains. Further, optimization problem is formulated by combining both the domains in order to evaluate the popularity matrix of target domain [66]. According to the values present in popularity matrix of target domain the most popular content is cached at the fog node. Figure 1.6 illustrates the proposed transfer learning caching procedure.

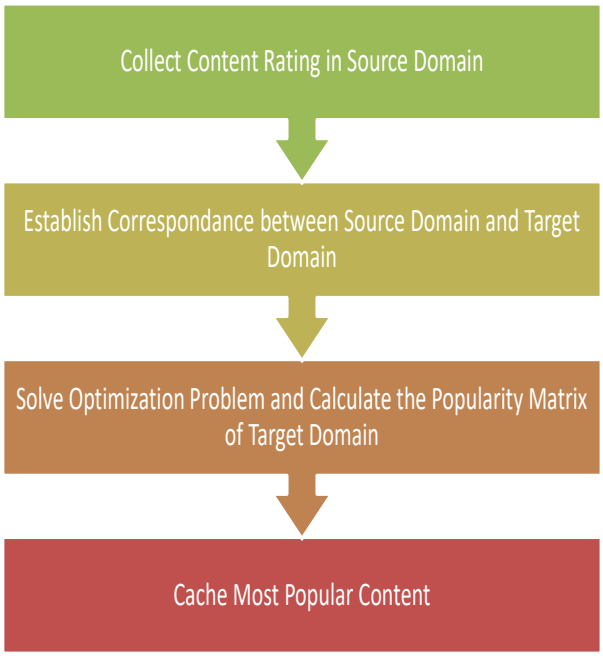

Figure 1.6: Training Using Transfer Learning

The only problem with this technique is that it can not give appropriate results if the relation between the source domain and the target domain is not efficient, which means that if the information demanded by the IoT devices is not related to the present information in the system, then Transfer Learning is not able to take accurate decision for caching.

4. **Recommendation Via Q Learning:** In existing local caching systems, the users do not know about the cached data. Therefore, they are not able to send their request despite of the fact that requested file is available in

the cache which decreases the efficiency of the system considerably. Hence in order to improve the efficiency of the system recommender algorithm is used [67]. In this system, the fog server broadcast an abstract to the users so that they gain knowledge about the files which are presently cached. An abstract contains one line introduction about the file and the ranking of the file in terms of the number of requests to the file. The value of the abstract also influences the decision of the user to request the file, since the request rate of a file, arrival and departure rate of the IoT devices is unknown in advance. In order to conquer this problem, Q learning is used which is a form of deep learning approach used to improve the performance of the system by reducing the latency and improving the throughput. It shows very promising accuracy in determining the future demand of the nodes by determining the Q value. Multiple layers are used in this network and these layers are used to process data and predict future demand of the users. Since more data is generated and processed by the lower layers as compared to the higher layers, more layers should be deployed near to the users in order to reduce the network traffic and improve the performance of the system.

The request rate for the i^{th} file depends upon the number of IoT devices which are present in the system and the number of IoT devices which arrived at that particular amount of time. As a result, unknown number of requests to the i^{th} file depend upon the caching action in the previous and present interval. During learning process, the Q value is selected for each state-action pair which maximizes the reward. Then number of remaining IoT devices are counted in order to select the action for current interval. At the end of the interval, the reward for the respective action is calculated and next state is observed. Then using these values a new Q value is calculated [67]. This approach increases the long term reward of the system and hence improves the performance.

5. **Deep Reinforcement Learning:**

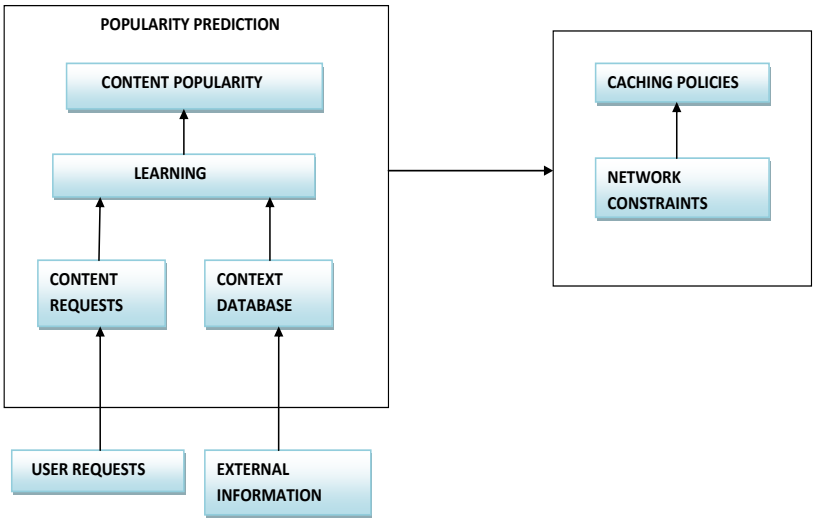

Figure 1.7: Popularity Prediction Approach

Deep Reinforcement Learning approach intelligently perceive the environment and automatically learns about the caching policy according to the history [68, 69]. Emergence of deep neural network has made it feasible to automatically learn from raw and possibly high-dimensional data. Learning-based caching techniques can be categorized into two approaches, popularity prediction and reinforcement learning approach. In popularity pre-

diction approach, first content popularity is predicted, and then according to popularity predictions caching policy is devised. This approach is summarized in figure 1.7 [52].

Various information like traffic patterns and context information are used to predict the content popularity. Content popularity is predicted in [70], by using users–content correlations and users' social ties through D2D communication. Authors in [71–73] have used various online learning algorithms to predict the content popularity. After the popularity prediction procedure, various caching policies and algorithms can be devised after solving optimization problems by combining estimated popularity with few network constraints or traditional caching algorithms. However these cachong problems are usually complex and are NP-hard.

In the second approach which is Reinforcement Learning Approach (RL), in place of separating popularity prediction and content placement, RL based approach consider both as a single entity and shown in figure 1.8.

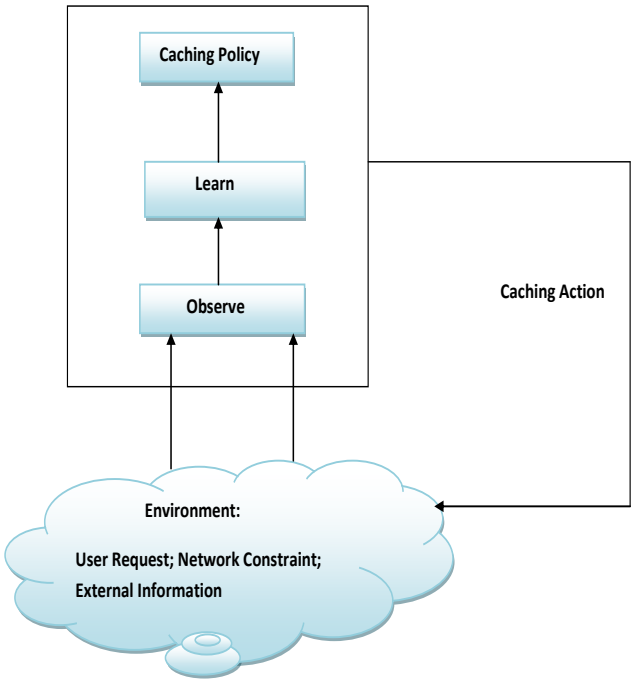

Figure 1.8: Popularity Prediction Approach

The RL agent train caching policy with observations which are based upon its action like QoE or offloaded traffic. This also avoid handling other factors affecting caching performance like nodes mobility or network topology etc. Various RL algorithms have been applied for designing fog nodes caching schemes. Advantage of RL based approaches is that RL agents can be trained directly on raw and high dimensional observations.

One of the RL based caching scheme is proposed by Zhu et al. in [68]. The authors defined the action space where each action indicates the data item to be replaced in the cache. Initially the fog node observes the environment and according to the environment it will obtain the state of the environment. Then according to the caching policy used, it takes the required action and attains the corresponding state. The action vector is represented as:

$$A_n = \{a_0, a_1, a_2, ..., a_c\}$$

where c is the number of files present in the cache, a_0 represents no caching action took place, a_v represents the

new file is cached at the v^{th} position and the corresponding state vector is represented as:

$$S_n = \{s_1, s_2, s_3, ..., s_n\}$$

After the action is taken, the system attains the reward which is fed back to the edge node and the process is repeated. Further, each data item is associated with two data fields: a) time stamp field and b) lifetime field. The time stamp field t_{gen} is used to indicate about the time when data is created or generated, while lifetime field t_{life} indicates the time upto which the value is valid in the item. The age of the data t_{age} is predicted by finding the difference between the current time and the time when it is generated. If $t_{age} < t_{life}$, then the requested data is available at the cache and is fresh and then data is directly returned to the user from the cache, otherwise the data available at the cache is not fresh. In that case, when data is not fresh and also if data is not available, then the node fetch fresh data from the cloud and return it to the IoT device. Deep reinforcement learning aims at maximizing the reward when the agent takes an action at the particular state. Figure 1.9 illustrates the application of deep reinforcement learning at fog nodes and to know the future demands of the IoT devices.

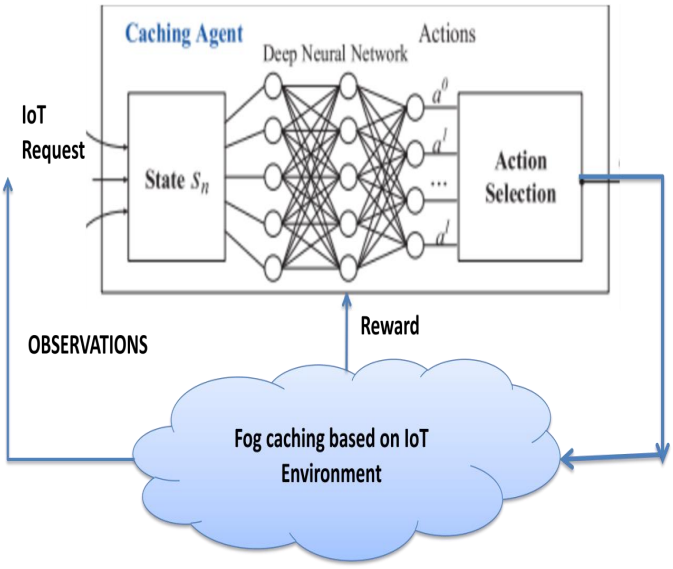

Figure 1.9: Applying DRL to Fog Caching

6. **Federated Learning:** Conventional machine learning approaches depend upon the data and processing in a central entity. However, this is not always possible as the private data is not sometimes accessible and also it requires great communication overhead to transmit initial data that is generated by large number of IoT devices to the central machine learning processors [74–78]. Therefore federated learning is the decentralized machine learning approach which keeps the data at the generation point itself and then locally trained models are only transmitted to the central processor. These algorithms also reduce the overall energy consumption and network bandwidth significantly by only transmitting the features rather than the whole data stream. It also respond in real time to reduce the latency. These machine learning algorithms exploit on-device processing power and efficiently use private data as model training is performed in a distributed manner and keep the data in-place i.e. place of generation. In the content popularity approaches discussed above, direct access of the private user data for

Techniques	Pros	Cons
Clustering	1. Coded caching is used which improves efficiency 2. Load balancing	1. If the size of cluster is large then efficiency is compromised.
Similarity Learning	1. Use only previous knowledge to find the similarity 2. Efficient if similar device arrives.	1. Can't predict demand if completely new node arrives.
Transfer Learning	1. Solve the new problem by exploiting existing knowledge.	1. Difficult to find the correspondence between the source domain and target domain.
Q Learning	1. End devices know which data is cached and where it is cached and use priority of data in caching which improves the efficiency. 2. Extra space is needed for storing priority of the data.	1. Delay in forecasting abstract. 2. Better accuracy in predicting demands.
Deep learning Reinforcement	1. Long term cost of user fetching data is reduced. 2.Overlapping coverage of fog nodes is considered	1. Co-operative caching is not considered. 2. Better accuracy in predicting demands
Federated Learning	1. Able to use the private data 2. Reduced communication overhead	1. Complex machine learning technique may have to be applied at the various levels.

Table 1.2: Pros and Cons of Machine Learning Algorithms

differentiating content may not be feasible. Till now federated learning has not been explored for caching in fog computing environment. Authors in [74] have successfully demonstrated that content popularity prediction can be done with the help of federated learning.

Pros and cons of various Machine learning techniques discussed above are summarized in table 1.2.

1.5 Future Research Directions

In this section various research issues related to the fog nodes caching are discussed. These points may help readers for their future research directions in the area of fog nodes caching.

a) **Lack of memory space:** In order to implement machine learning based system, it is necessary to have sufficient data at the learning system for learning purpose. However, the fog nodes do not have enough memory space hence, it is of profound importance to investigate an effective machine learning technique that can learn from limited available data. As discussed before, reader may explore federated learning which is not exploited much yet for content prediction in caching.

b) **Heterogeneous IoT Devices:** Most of the times, IoT devices are heterogeneous in nature, e.g. in smart homes various types of sensors like light, temperature etc. may be installed which generate lot of different kind of traffic. Till now, the impact of heterogeneity of IoT devices is not well addressed. In this kind of scenario, the network connectivity methods, protocols to handle these devices and the communication methods are not discussed which increases the latency while communicating with the fog nodes.

c) **Synchronization among fog nodes:** In the present works, the synchronization of data present at the various fog servers and the cloud servers is not discussed. Since the data produced by IoT devices is transient and become useless after sometime hence it is necessary to address the problem of synchronization of data at various fog servers and also with the cloud server.

d) **Game theoretic/ Auction models:** In various business models, the fog nodes earn by serving the IoT Devices. In this kind of systems, fog nodes may not cooperate with each other and may act selfish. Therefore, various game theory based or auction based theories may be applied to solve non-cooperation among fog nodes.

1.6 Conclusion

IoT devices generate lot of data which is stored and processed at cloud servers. To reduce the latency, fog computing has been introduced. However, there is a need of caching data at fog nodes to reduce the further communication with the cloud nodes. This chapter introduces various advantages of storing data of IoT devices at the fog nodes and subsequently the challenges faced in order to store data at the fog nodes. This article also describe how various machine learning techniques are used in order to predict the future demand of IoT devices and store most requested data at the fog nodes. The article is then concluded with the future research directions for the readers.

References

[1] B. Kang, D. Kim, and H. Choo, "Internet of everything: A large-scale autonomic iot gateway," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 206–214, 2017.

[2] N. C. Narendra, S. Nayak, and A. Shukla, "Managing large-scale transient data in iot systems," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2018, pp. 565–568.

[3] S. Vural, N. Wang, P. Navaratnam, and R. Tafazolli, "Caching transient data in internet content routers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1048–1061, 2016.

[4] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[5] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[6] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big iot data analytics: architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[7] S. Kitanov and T. Janevski, "State of the art: Fog computing for 5g networks," in *2016 24th Telecommunications Forum (TELFOR)*. IEEE, 2016, pp. 1–4.

[8] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[9] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, 2017.

[10] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[11] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[12] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, and C. Mahmoudi, "The nist definition of fog computing," National Institute of Standards and Technology, Tech. Rep., 2017.

[13] R. K. Barik, A. Tripathi, H. Dubey, R. K. Lenka, T. Pratik, S. Sharma, K. Mankodiya, V. Kumar, and H. Das, "Mistgis: Optimizing geospatial data analysis using mist computing," in *Progress in Computing, Analytics and Networking*. Springer, 2018, pp. 733–742.

[14] J. S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in fog and mist computing," *Computer*, vol. 48, no. 7, pp. 37–45, 2015.

[15] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[16] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 46–52, 2018.

[17] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[18] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[19] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.

[20] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future internet," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 48–54, 2019.

[21] A. Strunk, "Costs of virtual machine live migration: A survey," in *2012 IEEE Eighth World Congress on Services*. IEEE, 2012, pp. 323–329.

[22] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[23] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1791–1805, 2017.

[24] T. M. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, and M. Vilar-Montesinos, "A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard," *Sensors*, vol. 18, no. 6, p. 1798, 2018.

[25] J. P. Martin, A. Kandasamy, and K. Chandrasekaran, "Unraveling the challenges for the application of fog computing in different realms: A multifaceted study," in *Integrated Intelligent Computing, Communication and Security*. Springer, 2019, pp. 481–492.

[26] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," *The Journal of Supercomputing*, vol. 75, no. 4, pp. 2070–2105, 2019.

[27] A. Ahmed, H. Arkian, D. Battulga, A. J. Fahs, M. Farhadi, D. Giouroukis, A. Gougeon, F. O. Gutierrez, G. Pierre, P. R. Souza Jr *et al.*, "Fog computing applications: Taxonomy and requirements," *arXiv preprint arXiv:1907.11621*, 2019.

[28] P. H. Vilela, J. J. Rodrigues, P. Solic, K. Saleem, and V. Furtado, "Performance evaluation of a fog-assisted iot solution for e-health applications," *Future Generation Computer Systems*, vol. 97, pp. 379–386, 2019.

[29] N. M. A. Brahin, H. M. Nasir, A. Z. Jidin, M. F. Zulkifli, and T. Sutikno, "Development of vocabulary learning application by using machine learning technique," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 1, pp. 362–369, 2020.

[30] U. Vijay and N. Gupta, "Clustering in wsn based on minimum spanning tree using divide and conquer approach," in *Proceedings of World Academy of Science, Engineering and Technology*, no. 79. World Academy of Science, Engineering and Technology), 2013, p. 578.

[31] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing: Principles and paradigms*. John Wiley & Sons, 2010, vol. 87.

[32] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[33] M. Asemani, F. Jabbari, F. Abdollahei, and P. Bellavista, "A comprehensive fog-enabled architecture for iot platforms," in *International Congress on High-Performance Computing and Big Data Analysis*. Springer, 2019, pp. 177–190.

[34] M. I. Pramanik, R. Y. Lau, H. Demirkan, and M. A. K. Azad, "Smart health: Big data enabled health paradigm within smart cities," *Expert Systems with Applications*, vol. 87, pp. 370–383, 2017.

[35] C. Avasalcai, I. Murturi, and S. Dustdar, "Edge and fog: A survey, use cases, and future challenges," *Fog Computing: Theory and Practice*, 2020.

[36] L. Andrade, C. Lira, B. de Mello, A. Andrade, A. Coutinho, and C. Prazeres, "Fog of things: Fog computing in internet of things environments," in *Special Topics in Multimedia, IoT and Web Technologies*. Springer, 2020, pp. 23–50.

[37] E. K. Markakis, K. Karras, A. Sideris, G. Alexiou, and E. Pallis, "Computing, caching, and communication at the edge: The cornerstone for building a versatile 5g ecosystem," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 152–157, 2017.

[38] I. Al Ridhawi, N. Mostafa, Y. Kotb, M. Aloqaily, and I. Abualhaol, "Data caching and selection in 5g networks using f2f communication," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–6.

[39] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. S. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[40] I. Althamary, C.-W. Huang, P. Lin, S.-R. Yang, and C.-W. Cheng, "Popularity-based cache placement for fog networks," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 800–804.

[41] S. Wang, X. Huang, Y. Liu, and R. Yu, "Cachinmobile: An energy-efficient users caching scheme for fog computing," in *2016 IEEE/CIC international conference on communications in China (ICCC)*. IEEE, 2016, pp. 1–6.

[42] Z. Chen and M. Kountouris, "D2d caching vs. small cell caching: Where to cache content in a wireless network?" in *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2016, pp. 1–6.

[43] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 28–35, 2018.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[45] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[46] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[47] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 2016, pp. 1–6.

[48] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, 2016, pp. 185–188.

[49] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[50] C.-Y. Li, C.-H. Yen, K.-C. Wang, C.-W. You, S.-Y. Lau, C. C.-H. Chen, P. Huang, and H.-H. Chu, "Bioscope: an extensible bandage system for facilitating data collection in nursing assessments," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 477–480.

[51] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 323–336.

[52] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Network*, vol. 32, no. 6, pp. 50–57, 2018.

[53] M. Habib ur Rehman, P. P. Jayaraman, S. U. R. Malik, A. U. R. Khan, and M. Medhat Gaber, "Rededge: A novel architecture for big data processing in mobile edge computing environments," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, p. 17, 2017.

[54] C. K.-S. Leung, R. K. MacKinnon, and F. Jiang, "Reducing the search space for big data mining for interesting patterns from uncertain data," in *2014 IEEE International Congress on Big Data*. IEEE, 2014, pp. 315–322.

[55] A. Stateczny and M. Wlodarczyk-Sielicka, "Self-organizing artificial neural networks into hydrographic big data reduction process," in *Rough sets and intelligent systems paradigms*. Springer, 2014, pp. 335–342.

[56] A. Rágyanszki, K. Z. Gerlei, A. Surányi, A. Kelemen, S. J. K. Jensen, I. G. Csizmadia, and B. Viskolcz, "Big data reduction by fitting mathematical functions: A search for appropriate functions to fit ramachandran surfaces," *Chemical Physics Letters*, vol. 625, pp. 91–97, 2015.

[57] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, 2017.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[61] K. Katevas, I. Leontiadis, M. Pielot, and J. Serrà, "Practical processing of mobile sensor data for continual deep learning predictions," in *Proceedings of the 1st International Workshop on Deep Learning for mobile systems and applications*, 2017, pp. 19–24.

[62] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 351–360.

[63] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[64] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, 2018.

[65] B. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[66] E. Baştuğ, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2015, pp. 161–166.

[67] K. Guo, C. Yang, and T. Liu, "Caching in base station with recommendation via q-learning," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–6.

[68] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for internet of things: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2074–2083, 2018.

[69] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1024–1033, 2019.

[70] 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.

[71] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[72] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, 2016.

[73] X. Zhang, Y. Li, Y. Zhang, J. Zhang, H. Li, S. Wang, and D. Wang, "Information caching strategy for cyber social computing based wireless networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 391–402, 2017.

[74] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," *arXiv preprint arXiv:1908.06847*, 2019.

[75] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *arXiv preprint arXiv:1909.07972*, 2019.

[76] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *arXiv preprint arXiv:1911.02417*, 2019.

[77] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *arXiv preprint arXiv:1909.11875*, 2019.

[78] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.