Fog Computing Based on Machine Learning: A Review

https://doi.org/10.3991/ijim.v15i12.21313

Fady E. F. Samann (✉), Adnan Mohsin Abdulazeez
Duhok Polytechnic University, Duhok, Iraq
fady.samann@dpu.edu.krd

Shavan Askar
Erbil Polytechnic University, Erbil, Iraq

Abstract—Internet of Things (IoT) systems usually produce massive amounts of data, while the number of devices connected to the internet might reach billions by now. Sending all this data over the internet will overhead the cloud and consume bandwidth. Fog Computing's (FC) promising technology can solve the issue of computing and networking bottlenecks in large-scale IoT applications. This technology complements cloud computing by providing processing power and storage to the edge of the network. However, it still suffers from performance and security issues. Thus, machine learning (ML) attracts attention for enabling FC to settle its issues. Lately, utilizing ML has been a growing trend to improve FC applications, like resource management, security, lessen latency, and power usage. Also, intelligent FC was studied to address industry 4.0, bioinformatics, blockchain, and vehicular communication system issues. Due to the ML vital role in the FC paradigm, this work will shed light on recent studies that utilized ML in an FC environment. Background knowledge about ML and FC was also presented. This paper categorized the surveyed studies into three groups according to the aim of ML implementation. These studies were thoroughly reviewed and compared using sum-up tables. The results showed that not all studies used the same performance metric except those worked on security issues. In conclusion, the proposed ML models' simulations are not sufficient due to the heterogeneous nature of the FC paradigm.

Keywords—Fog Computing, Machine Learning, IoT, Optimization

1 Introduction

Today, the world has seen a growth in mobile devices and computers' everyday usage by individuals and organizations [1]. Applications and sensors of electronic devices are used to produce data, usually a massive amount of it. As a result, many companies must take responsibility for routinely maintaining vast volumes of data [2]. Companies currently need a dynamic information management architecture because of the transition to cloud computing and the benefits of this shift, such as scalability, availability, and pay-as-you-use features [3]. Cloud computing has made many services available that include platform, software, and infrastructure as services heading

toward anything can be presented as service [4], [5]. Still, it is not always feasible to transfer big data generated by sensors to the cloud for processing and storage [6], [7].

Moreover, several IoT applications require faster processing, only present-day cloud incapable of meeting such applications' latency requirement [8]. The problem is approached through the use of FC, which involves harnessing the computing power of devices close to a user to help with the storage and processing of data [9]. FC's different goals include efficiency improvements, data size reduction required to be transmitted to the cloud for data processing, analysis, and storage [10], [11]. The leading cause for this is often performance, but security and compliance can be other reasons [12]. Recently the AI algorithms have been applied to IoT data analysis [13]–[15]. End-user devices at the lowest layer of the network carry many unwanted features, such as insufficient memory, an inadequate low communication bandwidth, low processing power, and heterogeneous hardware that differ from the cloud infrastructure [16]. Computing technologies have advanced in various areas throughout the last decade, such as AI, GPU computing, cloud computing, and various hardware enhancements [17]. ML is the most widely implemented AI algorithm in various fields. Researchers have used ML to overcome networking problems in several earlier studies, including network routing, security, traffic engineering, and resource allocation [18]–[23].

ML plays a significant role in creating a smart/intelligent environment where autonomous management and operation are concerned [24], [25]. The significance of ML is extended to IoT, as without it, IoT will not be possible in terms of performing functional, monitoring, or preprocessing tasks [26], [27]. Moreover, meeting the IoT's diverse QoS demands remains a formidable issue due to the IoT devices' resource-limited nature[28]. Thus, describing ML concerning fog, cloud, and edge computing for IoT deployment is critical [29]. The ability to analyze data on network devices such as routers and switches is easy to implement using current technologies like Cisco's IOS XR [30]. Usually, works conducted on ML involved actuators, sensors, and low-level fog nodes. However, at a higher level, fog nodes can handle frameworks like Weka [31] and Scikit-learn to implement many AI applications. ML is used to execute, optimize, assign, or monitor functional tasks such as clustering, routing, duty-cycle management, data aggregation, and medium access control [32]. It is not easy to manage the relevant processes in the fog nodes because they are dynamic, complex, and heterogeneous.

Moreover, FC services should be enhanced by improving their diversity and transmission efficiency. ML has been used in the development of FC systems. Thus, nodes can benefit from ML in many aspects. For instance, deep analytics can be applied by including ML in FC [33]. Intelligent fog applications are being developed because ML can deliver reliable AI. These algorithms extract valuable information and details from captured data. This paper aims to give a brief epitome about the related topics and investigate current trends in the research field via literature review, which will enrich the body of knowledge with a summary of recent work in the field. Section two of this paper will give background knowledge about FC and ML from the perspective of FC. The reviewing methodology will be explained in section three. The related work section follows, where the most recent studies about the topic will be reviewed.

Section four will compare, summarize and discuss these studies. Finally, the last section will conclude section four's findings and give general remarks about them.

2 Theory and Background Knowledge

The following subsections will introduce the concept of FC in terms of architecture and software features. Then it will briefly introduce ML from the viewpoint of FC.

2.1 Fog Computing (FC) network architecture

FC complements the operations of cloud computing to reinforce the Quality of Service (QoS) and Quality of Experience (QoE) of the End-Users (EU) [34]. The FC architecture is versatile; the number of layers between the EU and the cloud provider can vary between a single layer or any hierarchical layers of fog nodes [35], [36]. Figure 1 illustrates the layers of the FC paradigm. The architecture's main aim is to allow for the EU's requests to be served by the cloud or pass it to the closest available fog nodes within the EU's vicinity. The most important entities of this architecture are the fog nodes. The functions of these nodes are communication, computation, and storage.

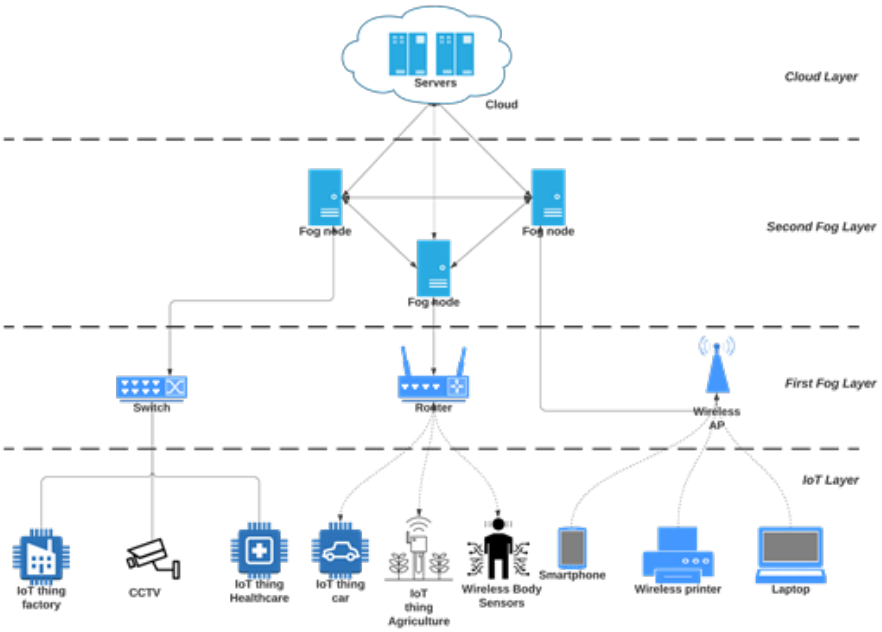

Fig. 1. FC Network Architecture

Moreover, fog nodes can be deployed in several locations with various functions and can have multiple forms [37]. The first fog layer from bottom to top (Figure 1) can include wireless access points (APs), routers, or IoT gateways. The IoT devices are usually one or two hops away from the nodes in that layer. A small number of hops allows services to be delivered with a couple of tens of milliseconds latency [38]. Moreover, this first layer usually provides single-purpose services at a small-scale, commonly relying on containerization technology. Any fog node in this layer can form a cluster with adjacent fog nodes or with the upper layer to cater to the EU request that requires substantial computational power. The second fog layer can give coverage to a large area as it can include fog nodes in the first layer. The second layer nodes are connected via a broadband connection, usual fiber to the cloud, and the first layer's nodes. The second layer's nodes typically have more power than those of the layer before it [39]. The second layer nodes can have faster CPUs with several cores, devoted Graphics Processing Units (GPUs), and bigger storage capacity. Hence, these nodes can provide quicker support to the first layer by scaling up computing capability. Due to the architecture's versatility, intelligence can be deployed and implemented anywhere in the cloud layer, the fog layers, or even at the EU devices at the network's edge.

2.2 FC software features

The paradigm of FC should be adaptable to support movability, scalability, and high availability for vast IoT devices. The paradigm utilizes advanced software technologies such as containers, virtual mechanics (VMs), and cluster orchestration to provide users' services and applications [40], [41]. There are significant differences between VMs and containers. However, they are both defined by isolated environments that run services requested by users.

A container is lightweight virtualized technology that packages a piece of software into isolated containers with everything it requires to run, including the code, runtime, system tools, system libraries, and settings [42]. As described above, containerization enables the service to be packed into an isolated software environment, which it can be fetched to run at different network locations. Many open-source and commercial containerization technologies are available, like Docker, LinuX Container, OpenVZ, Rocket, and LXD [43]. A container image can be fetched from a container registry by the fog node to directly serve requests without preconfiguring nodes or forcing users to rely on the cloud [44]. The container images can be quickly deployed cross platforms because it contains all software needed for running the software code.

Virtual Machine (VM) technology packages the services and applications with the full operating system as a VM image. Many prototypes for FC systems and academia in the mobile edge computing (MEC), like Cloudlet [45] and MAUI [46], used VM-based models to execute their systems. VMs are much larger and require more resources if compared to the containers [47]. Thence, a host machine (fog node) can hold far fewer VMs than containers. Moreover, during the fetching and initializing stage, a VM can require more storage, memory, and CPU than a container [48]. Finally, with fast deployment, resilient computing capacity, and fast migration, containers are favored over VMs in FC.

Applications like speech recognition and computer vision are process and storage demanding ones, so it might require to be distributed between a group of fog nodes to accomplish the rigorous latency requirements and high network availability. Thus, fog cluster management must manage adjacent fog nodes, usually of similar capabilities, to form clusters of nodes [49]. A cluster is composed logically of several containers that cooperate to split a task and process it in parallel [50]. To manage the fog cluster, a management mechanism will perform a critical role. Open-source and commercial orchestration frameworks such as Docker Swarm [43], Kubernetes [51], and Amazon ECS [52] are used by many academic works to implement the FC paradigm. A cluster manager is an assigned fog node in these frameworks to handle all the other nodes' operations, events, communications, orchestration, and load balancing [53]. The clustering feature allows flexibility in scaling up and down fog services [54]. This scaling happens when the computational need to pass the currently available resources. Thus, the manager orchestrates joining new fog nodes to the cluster. As it serves to deploy closer to the users, intelligence in FC is undoubtedly a massive beneficiary of container and cluster orchestration technologies [55]. For example, pre-trained ML-models can be packed in containers and deployed in fog nodes close to the users for processing data in real-time monitoring application such as e-healthcare system. The clustering of fog nodes can assist parallel processing for big data, keeping the flexible computing close to data sources to reduce overhead communication.

2.3 Machine Learning (ML) in FC

Adding intelligence capabilities to fog and IoT is done to provide services and applications or optimize system operations and improve network performance. FC is about moving some of the cloud's computing power towards the network's edge, where usually IoT devices and human users lay in the paradigm. Thus, two approaches are there to adopt intelligence in FC.

The first one is device-driven intelligence, as the IoT devices and the fog layers are getting smarter by adding sensing, more computing power, storage, and communications abilities [56]. Devices with such capabilities include local servers and access points, IoT gateways, and nodes for portable data aggregation carried by a human user. Researches aimed to enhance these devices' intelligence features, such as adding smart data processing and networking services to an e-health gateway [57] or gateway that can perform ML [58]. Also, monitoring wireless channel parameters via a neural network model can achieve efficient coverage and connectivity [59]. The edge devices with these abilities can provide excellent granularity in collecting data and thus enable the network to be context-conscious, make decisions and manage local resources.

The second approach is human-driven intelligence. The role of human users is essential, who shaped the design of the IoT landscape. Usually, data sources in the IoT networks are humans. Thus, their behavioral patterns play an essential role in training the network to be smarter. Many academic works are consecrated on serving human needs, such as e-butler to manage home appliances [60] and cognitive IoT-based smart home to improve living quality [61]. The vision here is to have a system that leverages human-related contexts while serving people and learning how to implement network-specific tasks, including resource scheduling efficiently and being

power consumption aware [62]. In this sense, human-driven intelligence maps human domain information into network domain decision-making that will benefit the network. Hence, the device and human-driven intelligence can be considered a possible solution while developing an FC system that meets the IoT application's QoS.

3 Methodology

This section describes the process followed to search, retrieve, and review the topic-related literature. This work used fog computing and machine learning as keywords to search for journal articles and conference proceedings on the Google Scholar search engine. The search filter was set to show studies published after 2017. The initial selection was based on reading the abstract and keywords provided by the corresponding websites. Then the most recent and related studies were retrieved. The second selection process was done after reading the main contributions and conclusion sections of the studies. Here, only the studies with the primary aim of implementing machine learning methods in fog computing environments were selected. The 19 final selected studies' literature was digested then reviewed by identifying the problems, objectives, methods, and results. According to how ML was implemented in the FC environment, the extracted information was presented in the next section as three subsections. Finally, the reviewed studies' work was discussed and compared through sum-up tables in section 7, while the conclusions were presented in the last section.

4 Related Work

This section will review the most recent and related academic works and compare them through tables in the next section. The reviewed studies are divided into three subsections. The first one will review the work that contributed to enhancing FC as technology, while the work focused on security and privacy preservation will be reviewed in the second one. Lastly, the studies that used ML-enabled FC as a solution for an application will be reviewed in the last subsection.

4.1 ML as paradigm enhancement

The subsequent studies used ML algorithms to enhance the FC paradigm's performance. A sum-up of this subsection can be found in Tables 1 and 2. Responding to how each fog device contributes to ML training and inference, Tu et al.[63] developed a method to optimize the distribution of processing through a network of fog devices. Multi-layer perceptron (MLP) and convolutional neural networks (CNN) were trained for image recognition with the MNIST data set to validate the algorithms. Six Raspberry Pis were used as nodes, and AWS DynamoDB acted as a cloud server to simulate network delays and calculate resource availability. Three Pis collect data and send it using Bluetooth to the other Pis, that act as gateways. The gateway nodes receive the data and either update local gradients or upload it to the cloud. The suggested algorithm almost halved overhead computation and equaled the centralized model's accuracy.

Tang et al.[64] addressed the lack of mobility support in FC networks with users of varied application quality requirements. Thus, they proposed algorithms for container migration and mobility support architecture in a three-layered FC architecture. The container migration manager is deployed in the fog layer as a container. The manager set the migration strategy base on users' GPS signal, communication delay, power consumption, and the container resource requirements. The cloud was used for load balancing and data replication. The migration problem is modeled as Markov Decision Process (MDP) spaces reduced by Q-learning algorithm. Moreover, a deep neural network training strategy made a fast decision. The training and testing were done using real-world data collected from tracking a taxi driver's GPS position. The baselines were the median absolute deviation algorithm, interquartile range regression algorithm, static threshold algorithm, and two versions of the Q-learning algorithm. The results showed that the proposed algorithm outperformed the other algorithms in terms of delay, power usage, and migration costs by 2.9 %, 48.5 %, and 58.4 %.

According to La et al.[65] latency and energy consumption issues in time-critical IoT applications can be solved by adding intelligent features to the FC layer. Hence, an FC optimization scheme that is device and human-driven for energy consumption and latency is proposed and presented as two case studies. The first one is based on adjusting the body sensor network's sampling rate to reduce energy usage and use high resolution when an urgent activity is detected. Here, an ML model was trained offline on a remote server to extract features and detect human activities then it was deployed as a container on the fog node. The authors tested three ML models (SVM, decision tree, and Gaussian Naive Bayes (GNB)). However, according to their past work [66], the selected model was the decision tree model because it gave higher accuracy than the others. The training and testing data-sets were taken from the data of accelerometer sensors attached to real human subjects. The performance test was carried out using OpenMote-CC2538 platforms running Contiki-OS with built-in sensors as the edge nodes and border-router, while a Raspberry-Pi 3 acted as the Fog node. The comparison baseline was the system without adaptability and intelligence. The proposed system showed high energy usage during the urgent-high state. Still, the authors stated that their system outperformed the baseline system and reduced the delay by ten times in the high and medium urgent states.

The second case study of this work simulated EU off-loading under dense fog deployment. Here, the ML model is trained to off-load the EU tasks for local processing or any of the fog nodes. The off-loading decision is based on an objective function that trades off between latency and energy consumption. The function was solved using Semi-Definite Relaxation (SDR) algorithm. This algorithm was compared with the local processing method, random assignment method, all-to-cloud method, and exhaustive search method. According to the results, the SDR algorithm achieved near-optimum efficiency and outperformed the other algorithms except for the exhaustive search method, which had slightly better results. Moreover, the power usage was reduced by increasing the number of active CPUs. However, as the number of fog nodes folds, the output benefit decreases slightly.

Patman et al.[67] indicated that there is a lack of FC testbeds that help to predict the network quality parameters. Hence, they presented a testbed to evaluate the accuracy of different ML models and the usefulness of deploying big data analytic at the network's edge. The testbed was based on GENI infrastructure to deploy image pro-

cessing services on SDN-operated hardware, while MDBMS was used for user requests' management. The authors performed two sets of experiments to test the accuracy and reaction time for the chosen ML models. The first one used a client-server topology to estimate the end-to-end delay of transmitting and rendering a set of 100 images on different system profiles and networking conditions. The selected ML models are Kernel-Ridge Regression, Support Vector Regression, Gaussian Process Regression, Random Forest Regression (RFR), and Multi-Layer Perceptron (MLP). The second experiment resolute the processing and transmission delay accuracy in each function's image processing pipeline. TensorFlow and OpenCV libraries were used for the image processing, while the K-fold cross-validation process evaluated the predictive models. The results showed that the RFR model was best used for predicting end-to-end delay and marginally better than MLP for predicting link usage.

Li et al.[68] considered managing VM allocation for the cloud and the fog as a solution for users trying to off-load their applications to local fog nodes only. This work formulated the VM allocation problem as a semi-Markov decision problem (SMDP). Two allocation methods based on SMDP were proposed; a model-based planning method and a model-free reinforcement learning (RL) method. The methods' objective was to create a VM allocation strategy to optimize revenue from a service system perspective. To promote their training, the SMDP degraded into a continuous-time Markov decision-making process (CTMDP). The Cloud-FC framework was simulated in MATLAB, and both methods' outputs are then compared with a greedy method and relative value iteration (RVI) algorithm under different parameters. The results showed that the RL algorithm's output would converge to a level close to that of the RVI algorithm and the model-based planning method within a short timeframe and surpass the greedy method.

Lavassani et al.[69] explored the advantages of FC by proposing a new distributed ML model that simulates the sensors' data stream in the fog instead of transmitting raw values to the cloud. The model's modified parameters at the sensor are transmitted to the FC system in long time intervals to conserve energy. The considered network architecture is composed of a cloud, a fog, and a lower WSN layer. Here, the sensor devices calculate the linear regression model parameters from the raw values and send them to the fog. Hence, new data are sent only in case of detecting changes and upgrading the model. An approximation of the original data stream is simulated in the fog layer using the modified model parameters. The state changing is based on the difference between the new trend's mean square error and an adequate error threshold.

The sensor devices used Contiki OS's multi-threading module to send the model and updates wirelessly to a gateway node connected to an FC system to push the information to the cloud. The fog layer was created using Raspberry Pi hardware, running the Raspbian OS. However, the cloud layer was implemented as standard desktop computer hardware with Ubuntu OS and ThingsBoard to store data and display sensor values. Mean Square Error (MSE) for the simulated data stream at the fog node and the original data stream was calculated to define the model-performance. Moreover, the considered baseline was a moving average model. According to the results, the model parameter approximations managed to recreate the original data stream with reasonable accuracy. Thus, the number of packets sent over the wireless link was reduced by 98%, with marginal accuracy loss. Moreover, the fog devices achieved 290 sensor model updates per second with an average end-to-end delay of 180ms.

4.2 ML for security and privacy preservation

The studies in this subsection are centered around the security enhancement of FC or the network in general. Also, some authors worked on preserving the privacy of user's data. The summary of these studies is presented in Tables 3 and 4. Alrashdi et al.[70] pointed out that the traditional intrusion detection systems (IDS) are not designed for limited resources IoT devices. They stated two main security issues; detecting zero-day attacks in IoT protocols and detecting Cyberattacks coming from the IoT networks. They added that network-based IDS (NIDS) consume storage for attacks' signatures and cannot detect new attacks in future network traffic. Also, host-based IDS (HIDS), such as anti-viruses, is not suitable for limited resources IoT devices. Thence, they proposed detecting threats in IoT systems using Random Forest ML algorithm deployed in the fog layer. The system design is based on a master security fog node that can monitor the network data, detect intrusions, and send alerts to the cloud. The UNSW-NB15 data set of Cyberattacks is used to train the RF model. The implementation and the training were done using Python. The result of the classification accuracy was 99.34%, while the false-positive rate was 0.02%. However, the confusion matrix showed that the precision is only 0.79% for attack detection, and recall is 0.97 %.

Rathore et al.[71] also worked on a method to detect Cyberattacks for IoT. They discussed that a centralized attack detection system does not work for IoT scenarios due to scalability, distribution, resource limitations, and latency characteristics. Moreover, most current solutions rely on supervised ML algorithms that require a considerable amount of labeled data for training. Also, the available data sets (like KDDCup99) are outdated and lead to less accurate results. Hence, the authors presented an attack detection framework that relies on the FC paradigm and uses an ELM-based semi-supervised Fuzzy C-means (ESFCM) method. The fog node senses the TCP/IP layers while connected to the IoT devices through a base station. SFCM and ELM were used to train the model with labeled data to label unlabeled data set. KDDTest-21 and KDDTest+ data-sets are used in this work. The evaluation base lines were centralized-method and other supervised algorithms (SVM, KNN, Logistic Regression, Random Forest, and Bayesian Network). According to the results, the framework achieved less detection time for both data-sets than the centralized framework and outperformed the other traditional classifiers.

Yang et al.[72] tried to resolve privacy issues raised by data aggregation in IoT systems. They discussed those solutions based on encryption technologies, like homomorphic encryption, usually cause large computational overhead on the cloud. Also, such encryption techniques are not designed to work on IoT devices with limited resources. Thus, the authors proposed a privacy preservation scheme for multi-functional data aggregation methods (such as additive and non-additive aggregation). The scheme deploys a trained ML model in the fog layer to predicate the aggregation query results, sent later by the fog to the cloud. In terms of privacy, the sensor IoT partitions the original data and report it separately to different fog nodes. The suggested system model has three entities; IoT sensors, Fog nodes, Fog centers, and the cloud. Here, the aggregation functions (min, max, medium, σ-percentile, average, and summation) are sent by the cloud to the fog center, which generates the proper queries for each sensor. After gathering all the sensors' reported data by the fog nodes, a train-

ing data set is generated to train a simple linear regression algorithm. The regression model is then used to predict the query set sent by the fog center, which can store the sensors' data to answer the fog centers' aggregation queries. The authors logically proved that generating the training data-set satisfies the differential privacy concept, which is achieved by adding Laplace noise to the entire training set. The ML-based Laplace differential privacy method (MLDP) was simulated and trained with two real-world data sets; Mobile Heath data set (MHEALTH) and Reference Energy Disaggregation Data Set (REDD). The mean absolute error (MAE) was calculated to evaluate the proposed model's accuracy and compared it with the traditional Laplace differential privacy method (LapDP). Results revealed that the performance of LapDP with a small enough query set is better than MLDP. However, MLDP outdid LapDP as the scale of the query set increased.

Huang et al.[73] tackled the issue of unauthorized access to sensitive data in the FC paradigm by presenting a novel Physical Unclonable Function (PUF)-based authentication scheme with a feedback loop to resist the ML attacks. The proposed scheme used an ML tool to extract a highly reliable PUF mathematical model during registration. Thus, unlike storing a large number of challenge-response pairs (CRPs), the server just needs to store the extraction model to save storage space. Adding a PUF feedback loop will make it harder for the opponent to create a useful ML attack model by introducing more noise into the CRPs. The PUF is based on a linear-feedback shift register (LFSR), and the loopback is an on-chip one-time programmable (OTP) eFuse. Without the eFuse, the scheme will become a traditional strong PUF. In this scheme, the genuine model is extracted and stored on a trusted server. Then the eFuse is burned after deployment to allow a feedback loop. The authors used the Xilinx Spartan-7 FPGA board to evaluate the proposed scheme. The control signals are produced by a PC connected through a serial port to the FPGA board. The eFuse is mimicked with a tri-state gate. MATLAB is used to process the data collected by a digital storage oscilloscope. Moreover, a temperature chamber is used to provide the changeable temperature. SVM and reliability-based ES-CMA ML attacks are also conducted on the proposed scheme. The results showed that the scheme could withstand 22X and 26X better than the basic PUF.

According to An et al.[74], fog nodes/MEC hosts are vulnerable to Cyberattacks due to their limited resources. Thus, they suggested a new lightweight IDS called Sample Selected Extreme Learning Machine (SS-ELM). In this scheme, the training data sets are stored in the cloud, and only samples of them are sent to the nodes for training the intrusion detection classifiers. Comparison baselines were backpropagation (BP) neural network algorithm, SVM, traditional ELM, and core vector machine ELM. In the MATLAB simulation, the KDDCup99 data set was used, and the cloud server was simulated on a PC running Windows 7. The SS-ELM showed high accuracy though slightly longer training time than the standard ELM. Also, as the training set's size increases, SS-ELM achieved the highest detection accuracy than ELM and SVM.

Abeshu et al.[75] argued that classical ML-based attack detection mechanisms have low accuracy and less scalability for Cyberattack detection. Moreover, classical mechanisms have no automated features, low detection rate, and cannot detect small mutants of existing attacks and zero-day attacks. Hence, they proposed deep learning-based IDS that multi-fold the process among the worker IDSs and the master IDS.

The worker fog node locally detects malicious events, while the master node is responsible for updating parameters using gradient descent. A modified version of the KDDCup99 data set called NSL-KDD was used for training and testing. Two Python libraries Keras and Theano, are combined with the Apache Spark framework to implement the system. Stacked Autoencoders (SAE) was trained to extract the hidden features without labels as a mean for the self-taught network. Then, end features were obtained by applying the features from the last step to the test data for SoftMax classification. The proposed IDS was evaluated against SoftMax without pre-training as a shallow learning model. The results revealed that the SAE model is superior to the shallow model in accuracy and detecting rate. The model's true-positive value was over 0.99, and the false-positive value for normal/attack classes was less than 1%.

4.3 ML as an application

The subsequent studies used ML with FC to solve a problem and produce an application. Tables 5 and 6 hold a summary of this subsection. Xu et al.[76] stated that processing a large amount of data in real-time for defect inspection on multiple assembly lines is challenging for industrial pipeline production. Hence, they proposed an intelligent and effective product inspection system that uses Lie Group ML. The system is designed as three modules (FC, server computing, and back-end interactive communication modules). The Lie Group classifier model was trained and tested with images of a mobile circuit board. The images were manually marked with 12 different categories of defects. The trained ML model is deployed to local fog nodes that receive images from camera sensors suspended over the assembly lines. The classification results are stored on the cloud server. The experiments were carried out as simulations in MATLAB. Receiver Operating Characteristic (ROC) curve was used to evaluate the system's performance, which is compared to the contour detection method, pixel-based method, and K-means algorithm. Also, efficiency runs for many products on the assembly line resolved the fog processing time and delay. Comparing to sending images directly to the cloud for defect detection, the system enhanced running performance by 53%. Moreover, it reduced the delay by 42%, while the accuracy increased by 28% compared to the other classification methods.

The main focus of Taneja et al.[77] work was on the health problem called lame in dairy cattle. According to the authors, this health problem is usually tackled by human visual inspection, which is not always accurate neither scalable. Besides, current automated solutions involve a managed environment in which the cows are walking on a particular track. Thus, they designed and implemented a fog-enabled IoT solution that utilizes ML and data analytic techniques for real-time cattle monitoring to detect lame early. The system relied on a long-range radio-based pedometer that attaches to the cow's leg. A receiver then sends the data to a local PC that acts like a Fog node. The application and management services are hosted on the IBM Watson IoT platform. The received data are stored on the node's database before processing. The node also extracts behavioral activities (step count, lying time, and swaps between lying down and standing up) to form a daily time series sum-up. For lame detection, mean analysis was used to investigate when lame animals start to deviate from the entire herd's mean. Here data from 146 cows were used in a six months trial. During this trial, the agricultural science student and the farmer reported 26 animals as lame cows. Be-

cause not all animals act in the same way, the Mean Absolute Deviation (MAD) between the cow activities and the herd mean was calculated for all activities over 14 days. The MAD set a threshold for assigning the animals into three clusters (Active, Normal, and Dormant). For lame early detection, the authors experimented with different ML classification algorithms (SVM, Random Forest, Decision Trees, and KNN). KNN was selected because it gave the right balance between early detection and accuracy. The detection process is implemented in the cloud, which sends an alert to the farmer if lame was detected. The system detected lameness with 87% accuracy three days before visual detection. Also, it decreased the amount of data transmitted to the cloud by 84%.

Blockchain applications require colossal computing power to solve the proof-of-work (PoW) puzzle. Thus, Luong et al.[78] argued that fog service providers should be incentivized to sell fog resources while ensuring QoS to the users. They designed a deep learning-based optimal auction for fog resource allocation to mine data in the blockchain network. A Feed-Forward neural (FFN) network was used to solve the assignment's optimization problem, while a second one for the payment's optimization problem. The first FFN used the sigmoid activation function in the hidden layer, and the output layer used the SoftMax function to produce the winners of the auction. The second FFN used the rectifier activation function in the output layer because the payments are non-negative. The Augmented Lagrangian method was also used for the loss function with weighted constraints in objective unsupervised learning. The deep learning was implemented using TensorFlow and deployed in the cloud after being trained by data sets to finds weights that minimize the loss function. The data sets for both FFNs include 5000 applicant rating profiles of the miners. The Adam optimizer is used in training to achieve fast and smooth convergence. The benchmark was the greedy algorithm, where the winners with the highest bids are selected to improve the revenue. The comparison was in terms of incentive compatibility (IC) and individual rationality (IR) violations versus several iterations for the Lagrange multiplier values. The proposed auction achieved rapid convergence to a higher revenue value compared to the greedy method.

Figuring out which vehicles are connected to which fog nodes is considered by Memon et al.[79]. They stated that the solution to this issue relies on how computation and data spread through the nodes. Also, the optimal fog node at a given time and location would vary due to load fluctuation. Thus, the authors presented fog and cost prediction models trained in the cloud and then deployed them to the vehicles. After deployment, the system is continuously trained by sensing and transmitting fog association and its corresponding cost at a given location and time to the cloud. The fog prediction model is based on a three-layered FFN. The cost prediction model utilized a dual-stacked Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells. GPS traces given by Shanghai Jiao Tong University were used to construct data sets for training and testing the ML models. A vehicular fog simulator was built using JAMScript to add fog access and cost information to vehicles moving according to the traces. The system was simulated in Python as a request routing system. For building the models, Keras API, Scikit-learn and TensorFlow were used. The two FFNs were benchmarked against KNN, SVR with RBF Kernel, and RNN with LSTM cells. The system allowed the vehicle to predict fog node handover points and low coverage areas accurately. The cost prediction model predicts a fog node's per-

formance at a given location and learns both geographic and temporal trends in the data. Still, it had the second-highest mean absolute error of 0.0518 after SVR with RBF Kernel.

Chen et al.[80] discussed that intelligent transportation systems have strict delay requirements for safety-related applications, especially in a dense traffic environment. Thus, they introduced the concept of Perception Reaction Time (PRT), a metric that reflects the combination of the packet transfer delay, off-loading task delay, and computation latency. A novel fog resource scheduling and allocation schemes were also designed based on a deep reinforcement learning algorithm. The resource allocation process is modeled as MDP solved to reduce the PRT metric. Moreover, they introduced the Information-Centric Network (ICN) concept. This concept enables the network to cache the various applications' contents in their respective vehicles and combine the packages with the same contents to minimize the delay. In the ICN-based vehicular fog framework, two scenarios were considered. The first one is when a vehicular-fog is formed by vehicles in close approximate with the same speed. The second scenario is when the vehicular fog near an RSU can access the MEC server for extra computer resources. Vehicles in the same vehicular fog off-load to each other and share resources. For safety-related applications, the optimization problem was solved to minimized the PRT. For non-safety ones, only the time-constraint was set to be satisfied. Deep reinforcement learning was compared with the Q-learning algorithm and the Adaptive Learning-based Task Offloading Algorithm (ALTO). The simulation showed that the algorithm converged faster than the Q-learning algorithm and behaved differently on the safety-related and non-safety-related applications.

Merelli et al.[81] stated that portable RNA/DNA sequencing systems, like Oxford Nanopore MinION, stream a large amount of data that can potentially be tricky to handle. The authors developed an FC framework to support a network of MinION devices operating in the field. Different System-on-Chip Low-Power Devices (SoCs) platforms were tested for real-time data processing before the results were obtained on a cloud-based analysis platform. These platforms (see Table 3) were integrated circuits designed for mobile and embedded applications, consisting of low-power multicore processors coupled with a GPU and multiple I/O ports. The MinION devices' throughput is usually in Gbps. The raw data is sent to Metrichor, a cloud-based software provided by Oxford Nanopore that uses Hidden Markov Models (HMMs). Thus, the authors deployed a counterpart open-source software called DeepNano on the SoCs platforms. Also, MiniKraken was used to classify and quantify bacteria within the samples. MiniKraken is open-source software for the allocation of taxonomic labels to metagenomic DNA sequences. The operations are performed locally on the SoCs (Fog nodes) to identify the genomic sequences and bacteria classification. Ubuntu operating system was used for all platforms, and Python and the Theano library were deployed to train the neural networks. The SoC platforms' performance was evaluated by using four data-sets from the work of another author. The eight-core processor XeonD performed better than the other platforms in terms of processed bases per second. N3700 presented the best result in terms of power usage, while the eight-core processors (XeonD and Avoton) were the most power-hungry when dealing with MiniKraken then with DeepNano.

O'Donovan et al.[82] argued that classic cloud computing is naturally at odds with Industry 4.0 decentralized decision-making and reliable real-time control. Although

cloud and service-based computing can support distributed engineering scenarios, intelligence and processing are usually centralized. This centralization means that distributed clients rely on consistent and resilient connections. This research introduces an industrial Cyber-physical architecture based on the FC framework. It integrates ready-made PMML-encoded ML models in factory operations. Once the cloud received contact from the fog node at the factory, the request is authenticated using the 128-bit Global Unique Identifier (GUID) of the node. A cloud database of registered devices is used to scan the GUID to classify the node's engineering programs and then return the appropriate PMML models to be deployed or synchronized. The system execution time and the number of failed communications were captured by a test machine hosting the JMeter open-source program for load monitoring. The OpenScoring engine was mounted behind the interfaces to manage JMeter queries and ran the PMML-encoded model using the presented data. The PMML model was derived from an existing SVM model, which forecasts defective heating operations within industrial air handling units. The fog interface achieved lower execution times than the cloud interface, with latency calculated at 92.9%, 99.4%, 67.7% and 91.0% for 50,100, 250, 500 connections, respectively. No failed communication registered by the fog interface at the stress test, but the cloud interface recorded 6.6% more failures for 500 connections.

5 Comparison and Discussion

In this section, the reviewed studies' choices and results will be compared through the comparison tables below. Then, it will be discussed with general remarks about the noticeable trends. Each of the subsections in section 4 has two tables in this part of the paper, where some cells marked with a dash cause the information not mentioned or used by the corresponding study. The entries of tables 1,3, and 5 are the authors' problems and methods to solve them, besides the baselines and results. Tables 2,4, and 6 include the data set, hardware, and software used for evaluation and simulation in each study.

Table 1. Studies Focused on ML as Paradigm Enhancement (Subsection 4.1)

Ref.	Details	
[63]	Problem	Decentralizing ML.
	Method	MLP and CNN algorithms.
	Baseline	Centralized model
	Results	Highest accuracy and lowest cost.
[64]	Problem	Container migration and mobility support.
	Method	Q-learning algorithm to reduce the MDP spaces.
	Baseline	Median absolute deviation algorithm, Interquartile range regression algorithm, Static threshold algorithm, and two versions of the Q-learning algorithm.
	Results	Reduced the delay by 2.9%, power consumption by 48.5%, and migration cost by 58.4%.
[65]	Problem	Latency and energy consumption issues in time-critical IoT applications.
	Method	SVM, decision tree, GNB, and SDR algorithms.
	Baseline	Without intelligence and adaptability approach/local processing method, random assignment method, all to cloud method, and exhaustive search method.
	Results	SDR-based algorithm outperformed the other schemes.

[67]	Problem	Lack of FC testbeds that help to predict the network quality parameters.
	Method	Kernel-Ridge Regression, Support Vector Regression, Gaussian Process Regression, RFR, and MLP algorithms.
	Baseline	The same techniques.
	Results	The RFR model was the best for predicting end-to-end delay and marginally better than MLP for predicting link usage.
[68]	Problem	VM allocation for the cloud and the fog to optimize revenue.
	Method	Solving the SMDP by a model-based planning method and a model-free reinforcement learning method.
	Baseline	A greedy method and relative value iteration (RVI) algorithm.
	Results	As the request rate increase, the RL performed better than other algorithms.
[69]	Problem	Simulating the data stream in the fog instead of sending raw data to the cloud.
	Method	A probability graph network model describes a succession of sequential multi-linear streams.
	Baseline	A moving average model.
	Results	98% decrease in the number of sent packets, 97% accuracy of stream simulation with 180ms end-to-end delay.

Table 2. Hardware and Software used by the Studies in Subsection 4.1

Ref.	Data	Edge node	Fog node	Cloud	Simulator
[63]	MNIST data set.	Raspberry Pis	Raspberry Pis	AWS DynamoDB	-
[64]	GPS position of a taxi driver.	VMs (Ubuntu 14.04 LTS)	VMs (Ubuntu 14.04 LTS with Docker 1.9 and CRIU 2.2)	-	-
[65]	Accelerometer sensors attached to real human subjects.	OpenMote-CC2538 platforms running Contiki-OS with built-in sensors.	Raspberry-Pi 3	-	-
[67]	100 images used ranged between 135KB and 3.4MB	-	OpenFlow-enabled switches configured with an Ethernet bridge and connected to an SDN-controller.	GENI infrastructure	-
[68]	The data obtained by Monte Carlo method under specific parameters.	-	-	-	MATLAB
[69]	Sensor's raw data	TelosB sensor motes.	Raspberry Pi	A desktop computer (Ubuntu OS and ThingsBoard)	MATLAB for processing results

In terms of the studies aimed at enhancing FC (Table 1), the ML models were trained on the cloud or offline then deployed on the fog nodes. Yet, Tu et al.[63] considered distributing the ML among the fog and cloud. Ref. [64] and Ref. [68] formulated optimization problems such as off-loading, scheduling, and resource allocation as a Markov Decision Problem and solved by an ML model. Usually, formulating a multi-objectives optimization problem with MDP is complex, time-consuming, and

might give undesired behaviors. Also, the location of fog nodes or distance from the end-user is not considered, while distance could affect latency calculations and network service migration. From Table1, the comparison's baselines were ML algorithms at the same level or less complex than the proposed ones. Still, La et al.[65] introduced an intelligent scheme to optimize energy consumption and reduce latency but compared it with non-intelligent ones. Even though SDN technology is popular with implementing and deploying FC, only Patman et al.[67] considered using SDN-operated hardware (Table 2). All the work in this group used supervised ML algorithms and relied on hardware to implement their systems rather than simulation. An exciting work by Lavassani et al.[69] proposed a scheme to simulate the IoT node's data stream at the fog layer rather than sending it directly to the cloud. The data simulation allowed for a significant reduction in the number of data sent across the network.

Table 3. Studies Focused on ML for Security and Privacy Preservation (Subsection 4.2)

Ref.	Details	
[70]	Problem	Detecting zero-day attacks in IoT protocols and detect cyberattacks coming from the IoT networks in a smart city.
	Method	Random Forest ML algorithm with Extra Trees classifier to reduce the data set's features.
	Baseline	-
	Results	Classification accuracy was 99.34%, and the false-positive rate was 0.02%
[71]	Problem	Centralized IDS do not work for IoT scenarios because of scalability, distribution, resource limitations, and latency characteristics.
	Method	ELM-based semi-supervised Fuzzy C-means method.
	Baseline	Centralized method, SVM, KNN, Logistic Regression, Random Forest, and Bayesian Network.
	Results	Detection time of 11ms and an accuracy rate of 86.53%.
[72]	Problem	Privacy issues raised by data aggregation in IoT systems.
	Method	ML-based Laplace differential privacy method (MLDP)
	Baseline	Traditional Laplace differential privacy method (LapDP).
	Results	LapDP with a small enough query set is better than MLDP. However, MLDP outdid LapDP as the scale of the query set increased.
[73]	Problem	Unauthorized access to sensitive data in the FC paradigm.
	Method	A PUF-based authentication scheme with a feedback loop to resist ML attacks.
	Baseline	Basic PUF, SVM, and reliability-based ES-CMA machine learning attacks
	Results	22X and 26X are more potent than the baseline.
[74]	Problem	Fog nodes/MEC hosts are vulnerable to cyberattacks due to their resources being limited.
	Method	A new lightweight IDS called Sample Selected Extreme Learning Machine.
	Baseline	Back Propagation (BP) neural network algorithm, SVM, traditional ELM, and core vector machine ELM
	Results	Accuracy of 99.07% and training time of 4.52s
[75]	Problem	Classical ML-based attack detection methods have low accuracy and less scalability for cyberattack detection in vastly distributed nodes such as IoT.
	Method	IDS based on deep learning (Stacked Autoencoders (SAE)).
	Baseline	SoftMax classification as a shallow learning model.
	Results	Accuracy of 99.2%, detection rate of 99.27%, and the false alarm rate of 0.85%

Table 4. Hardware and Software used by the Studies in Subsection 4.2

Ref.	Data	Edge node	Fog node	Cloud	Simulator
[70]	UNSW-NB15	-	-	-	Python
[71]	KDDTest^{-21} and KDDTest^{+}	-	-	-	-
[72]	MHEALTH and REDD	-	-	-	-
[73]	Control signals generated by a PC	-	Xilinx Sparten-7 evaluation FPGA board	-	-
[74]	KDD Cup 99	-	-	PC running Windows7	MATLAB
[75]	NSL-KDD	-	-	-	Python libraries (Keras and Theano) with Apache Spark framework.

The authors tackled FC security issues (Table 3), produced accurate and light-weight IDS. However, they only trained their ML models for detection and did not implement or simulate their network models. The KDD data set was the most used among these studies (Table 4). Still, Alrashdi et al.[70] pointed out that the KDDCup99 data set is outdated and cannot give high detection accuracy. That is why they used the UNSW-NB15 data set. Also, both Ref. [70] and Ref. [75] aimed at detecting zero-day attacks by reducing the false-positive rate. However, a zero-day is defined as taking advantage of unknown software vulnerability. So supervised learning algorithms might not be the answer to this issue. All the studies in this group used supervised learning algorithms except Rathore et al.[71] who used the ESFCM method. Still, they compared their results to only traditional supervised learning algorithms. Huang et al.[73] used a PUF-based authentication scheme to prevent unauthorized access to the FC paradigm's sensitive data. They used an FPGA board to implement the scheme, which proved resistive to ML attacks (Table 4). Still, due to the hardware nature of the scheme, a physical attack is unpreventable. In terms of privacy-preserving, Yang et al.[72] used an ML algorithm to implement multi-functional data aggregation instead of encryption technology, which lead to computational overhead. The original data is divided and sent to separate fog nodes assigned with the same privacy budget to preserve differential privacy.

Table 5. Studies Focused on ML as an Application (Subsection 4.3)

Ref.	Details	
[76]	Problem	Processing a large amount of data in real-time for defect inspection on multiple assembly lines is challenging.
	Method	Product inspection system based on Lie Group ML through image processing.
	Baseline	Contour detection method, pixel-based method, K-means algorithm, and all to the cloud model.
	Results	27.86% accuracy increase, 52.57% improvement in running performance, and 42.13% delay reduction.
[77]	Problem	The health problem called lame in dairy cattle.
	Method	Mean Absolute Deviation (MAD) for clustering animals' activities and ML classification algorithms (SVM, Random Forest, Decision Trees, and KNN) for lame early detection.
	Baseline	Traditional cloud-based method.
	Results	Overall accuracy of 87%, three days before visual detection, and 84% decrease in the amount of data transmitted to the cloud.
[78]	Problem	In a blockchain application, the fog service provider should sell fog resources while ensuring QoS to the users.
	Method	Two Feed-Forward neural networks for solving assignment and payment optimization problems.
	Baseline	Greedy algorithm.
	Results	The proposed auction achieved rapid convergence to a higher revenue value com-pared to the greedy method.
[79]	Problem	Handover optimization for the Internet of Vehicles
	Method	A fog prediction model based on three-layer FFN and a cost predication model utilizing dual-stacked RNN with LSTM cells
	Baseline	KNN, SVR with RBF Kernel, and RNN with LSTM cells.
	Results	The fog predictor model achieved an accuracy of 99.92%, and the cost predictor model achieved a mean absolute error of 0.0518.
[80]	Problem	Strict delay requirements for safety-related applications in a dense vehicular traffic environment.
	Method	Fog resource scheduling and allocation schemes based on a deep reinforcement learning algorithm.
	Baseline	The Q-Learning algorithm and the ALTO algorithm.
	Results	70% reduction in the RPT metric.
[81]	Problem	Portable RNA/DNA sequencing systems stream a large amount of data that is difficult to handle locally.
	Method	DeepNano describes the probability distribution and MiniKraken for classifying and quantifying bacteria within samples deployed on SoCs platforms.
	Baseline	The used SoCs (Intel Avoton C2750, Intel XeonD 1540, Intel Pentium J4205, Intel Pentium N3700, and Intel Pentium N3710).
	Results	Avoton SoC platform presents the best scalability, but the XeonD SoC platform has the highest performance in most cases.
[82]	Problem	Industry 4.0 decentralized decision-making and reliable real-time control challenges in cloud computing.
	Method	Cyber-physical architecture based on FC, which can integrate ready-made PMML-encoded SVM models in factory operations.
	Baseline	Cloud-based model.
	Results	The fog interface performed on average 87.75% lower than the cloud interface in terms of maximum latency.

Table 6. Hardware and Software used by the Studies in Subsection 4.3

Ref.	Data	Edge node	Fog node	Cloud	Simulator
[76]	Mobile circuit board images manually marked with 12 different defects.	Video sensor with network access adapter	PC running Windows7	-	-
[77]	Pedometer sensor data from 146 cows taken during a six months trial	A long-range radio-based pedometer that attaches to the cow's front leg	PC in the farm that also acts as an IoT gateway	IBM Watson IoT platform	-
[78]	Five thousand applicant rating profiles of the miners.	-	-	-	Python
[79]	GPS traces from Shanghai Jiao Tong University.	-	-	-	A vehicular fog simulator built using JAMScript and Python for implementing the neural networks
[80]	Simulated	-	-	-	-
[81]	Four data-sets from the work of other researchers.	Oxford Nanopore MinION	Intel Avoton C2750, Intel XeonD 1540, Intel Pentium J4205, Intel Pentium N3700 and Intel Pentium N3710. All SoCs are running Ubuntu OS.	AWS IoT platform	Python for training the neural networks
[82]	Ready SVM models	Temperature sensors of industrial air handling units.	AHU9-PLC with python ingestion engine and Cylon BMS	AWS IoT platform	A PC hosting (JMeter) for load monitoring

The last group of reviewed studies (Table 5) used ML algorithms in FC environment to solve problems in the intelligent industry [76], [82], vehicular communication system [79], [80], health and bioinformatics [77], [81], and blockchain technology [78]. Some of them used neural networks and deep learning to tackle optimization problems [78]–[80], while others did the same but to address classification and data analysis tasks [81]. Some researchers opt for statistical models and traditional ML algorithms to solve their problems [77]. Even though most of the work listed in Table 6 used sensors to produce data sets or ready-made ones, some authors were concerned about the lack of training samples [76], [77], [81]. O'Donovan et al.[82] showed how time-sensitive applications in Industry 4.0 could utilize deploying ML models on the fog nodes to enhance performance and reduce relying on the cloud. Among all the studies, there was a trend of using cloud platforms such as AWT IoT platform and IBM Watson IoT platform to manage and provide storage and advanced analysis for their models or systems. Moreover, Python and MATLAB were the most used for evaluation and simulation. In terms of the results, it is hard to compare the figures in the tables above cause the authors used different performance metrics that fit their cases and compared their model to various baseline ones. Still, it is possible to compare the results in Table 3 because most of the authors used the confusion matrix and ROC curve to measure the IDS's accuracy.

6 Conclusion

This paper examined ML's role in the FC area. The selected studies were limited by searching through the Google Scholar engine to focus the search. The second limitation of this work is the initial selection of the studies based on the abstract and keywords provided by the corresponding websites. Machine learning has tremendous potential to become the primary technology in many domains. Thus, it can function as a powerful analytic tool for FC. This work presented the ML's involvement in three prospects of FC: paradigm enhancer, security and privacy-preserving, and application solution. The reviewed studies exhibit outstanding intelligent solutions based on the FC environment. However, the lack of testbeds and the heterogeneous characteristics of FC made it harder to evaluate these solutions in the real world. According to the tables in section 4, the most common ML methods are statistical or supervised learning algorithms. Future works should include unsupervised learning ML algorithms in an SDN-enabled FC environment. Also, hardware implementation of the IDS instead of relying only on simulation. Finally, the reviewed studies addressed numerous challenges and open issues and verified the flexibility and efficiency of executing ML in the FC environment.

7 References

[1] J. C. Guevara, R. da S. Torres, and N. L. S. da Fonseca, "On the classification of fog computing applications: A machine learning perspective," J. Netw. Comput. Appl., vol. 159, p. 102596, Jun. 2020, https://doi.org/10.1016/j.jnca.2020.102596

[2] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and Fog Computing Enabled AI for IoT-An Overview," in 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Mar. 2019, pp. 51–56, https://doi.org/10.1109/AICAS.2019.8771621

[3] L. M. Herger, M. Bodarky, and C. Fonseca, "Breaking Down the Barriers for Moving an Enterprise to Cloud," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), Jul. 2018, vol. 2018-July, pp. 572–576, https://doi.org/10.1109/CLOUD.2018.00079

[4] A. Rashid and A. Chaturvedi, "Cloud Computing Characteristics and Services A Brief Review," Int. J. Comput. Sci. Eng., vol. 7, no. 2, pp. 421–426, Feb. 2019, https://doi.org/10.26438/ijcse/v7i2.421426

[5] M. I. Khaleel and A. M. Ahmed, "Green Cloud Framework for Reducing Carbon Dioxide Emissions in Cloud Infrastructure," in 2019 International Conference on Advanced Science and Engineering (ICOASE), Apr. 2019, pp. 29–34, https://doi.org/10.1109/ICOASE.2019.8723701

[6] H.-C. Chao, B. Hu, and C.-Y. Chen, "Fog computing and internet of everything for emerging enterprise information systems," Enterp. Inf. Syst., vol. 12, no. 4, pp. 371–372, Apr. 2018, https://doi.org/10.1080/17517575.2017.1405076

[7] A. Mijuskovic, R. Bemthuis, A. Aldea, and P. Havinga, "An Enterprise Architecture based on Cloud, Fog and Edge Computing for an Airfield Lighting Management System," in Proceedings - IEEE International Enterprise Distributed Object Computing Workshop,

EDOCW, Oct. 2020, vol. 2020-October, pp. 63–73, https://doi.org/10.1109/EDOCW49879.2020.00021

[8] D. Minoli, K. Sohraby, and B. Occhiogrosso, "IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems," IEEE Internet Things J., vol. 4, no. 1, pp. 269–283, Feb. 2017, https://doi.org/10.1109/JIOT.2017.2647881

[9] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog Computing: A Comprehensive Architectural Survey," IEEE Access, vol. 8, pp. 69105–69133, 2020, https://doi.org/10.1109/ACCESS.2020.2983253

[10] M. Al Yami and D. Schaefer, "Fog Computing as a Complementary Approach to Cloud Computing," in 2019 International Conference on Computer and Information Sciences (ICCIS), Apr. 2019, no. April, pp. 1–5, https://doi.org/10.1109/ICCISci.2019.8716402

[11] A. S. Raman, "Potentials of Fog Computing in Higher Education," Int. J. Emerg. Technol. Learn., vol. 14, no. 18, p. 194, Sep. 2019, https://doi.org/10.3991/ijet.v14i18.10765

[12] M. Moh and R. Raju, "Machine Learning Techniques for Security of Internet of Things (IoT) and Fog Computing Systems," in 2018 International Conference on High Performance Computing & Simulation (HPCS), Jul. 2018, pp. 709–715, https://doi.org/10.1109/HPCS.2018.00116

[13] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," Futur. Gener. Comput. Syst., vol. 101, pp. 747–759, Dec. 2019, https://doi.org/10.1016/j.future.2019.07.030

[14] J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "Edge-AI in LoRa-based Health Monitoring: Fall Detection System with Fog Computing and LSTM Recurrent Neural Networks," in 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Jul. 2019, pp. 601–604, https://doi.org/10.1109/TSP.2019.8768883

[15] T. N. Gia, L. Qingqing, J. P. Queralta, Z. Zou, H. Tenhunen, and T. Westerlund, "Edge AI in Smart Farming IoT: CNNs at the Edge and Fog Computing with LoRa," in 2019 IEEE AFRICON, Sep. 2019, vol. 2019-Septe, no. September, pp. 1–6, https://doi.org/10.1109/AFRICON46755.2019.9134049

[16] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W.-Z. Song, "Pushing Analytics to the Edge," in 2016 IEEE Global Communications Conference (GLOBECOM), Dec. 2016, no. Ml, pp. 1–6, https://doi.org/10.1109/GLOCOM.2016.7842181

[17] M. Usama et al., "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," IEEE Access, vol. 7, pp. 65579–65615, 2019, https://doi.org/10.1109/ACCESS.2019.2916648

[18] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive IoT Gateways: Automatic Task Sharing and Switching between Cloud and Edge/Fog Computing Fatemeh," in Proceedings of the SIGCOMM Posters and Demos on - SIGCOMM Posters and Demos '17, 2017, no. November, pp. 121–123, https://doi.org/10.1145/3123878.3132008

[19] S. Ayoubi et al., "Machine Learning for Cognitive Network Management," IEEE Commun. Mag., vol. 56, no. 1, pp. 158–165, Jan. 2018, https://doi.org/10.1109/MCOM.2018.1700560

[20] Z. M. Fadlullah et al., "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems," IEEE Commun. Surv. Tutorials, vol. 19, no. 4, pp. 2432–2455, 2017, https://doi.org/10.1109/COMST.2017.2707140

[21] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine Learning for Networking: Workflow, Advances and Opportunities," IEEE Netw., vol. 32, no. 2, pp. 92–99, Mar. 2018, https://doi.org/10.1109/MNET.2017.1700200

[22] C. A. Hammerschmidt, S. Garcia, S. Verwer, and R. State, "Reliable Machine Learning for Networking: Key Issues and Approaches," in 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Oct. 2017, vol. 2017-Octob, pp. 167–170, https://doi.org/10.1109/LCN.2017.74

[23] P. Casas, J. Vanerio, and K. Fukuda, "GML learning, a generic machine learning model for network measurements analysis," in 2017 13th International Conference on Network and Service Management (CNSM), Nov. 2017, vol. 2018-Janua, pp. 1–9, https://doi.org/10.23919/CNSM.2017.8255998

[24] S. I. Saleem, S. R. M. Zeebaree, D. Q. Zeebaree, and A. M. Abdulazeez, "Building smart cities applications based on IoT technologies: A review," Technol. Reports Kansai Univ., vol. 62, no. 3, pp. 1083–1092, 2020 .

[25] K. Mason and S. Grijalva, "A Review of Reinforcement Learning for Autonomous Building Energy Management," Comput. Electr. Eng., vol. 78, pp. 300–312, Mar. 2019, https://doi.org/10.1016/j.compeleceng.2019.07.019

[26] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A Review of Machine Learning and IoT in Smart Transportation," Futur. Internet, vol. 11, no. 4, p. 94, Apr. 2019, https://doi.org/10.3390/fi11040094

[27] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and IoT applications," in 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), Aug. 2017, vol. 2018-Janua, pp. 1–8, https://doi.org/10.1109/IISA.2017.8316459

[28] F. E. F. Samann, S. R. M. Zeebaree, and S. Askar, "IoT Provisioning QoS based on Cloud and Fog Computing," J. Appl. Sci. Technol. Trends, vol. 2, no. 01, pp. 29–40, Mar. 2021, https://doi.org/10.38094/jastt20190

[29] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" in 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), May 2017, pp. 62–67, https://doi.org/10.1109/FMEC.2017.7946409

[30] A. Jain and P. Singhal, "Fog computing: Driving force behind the emergence of edge computing," in 2016 International Conference System Modeling & Advancement in Research Trends (SMART), 2016, no. January, pp. 294–297, https://doi.org/10.1109/SYSMART.2016.7894538

[31] S. Bosaeed, I. Katib, and R. Mehmood, "A Fog-Augmented Machine Learning based SMS Spam Detection and Classification System," in 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Apr. 2020, pp. 325–330, https://doi.org/10.1109/FMEC49853.2020.9144833

[32] M. Barzegaran, A. Cervin, and P. Pop, "Performance Optimization of Control Applications on Fog Computing Platforms Using Scheduling and Isolation," IEEE Access, vol. 8, pp. 104085–104098, 2020, https://doi.org/10.1109/ACCESS.2020.2999322

[33] S. Dey and A. Mukherjee, "Implementing Deep Learning and Inferencing on Fog and Edge Computing Systems," in 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Mar. 2018, pp. 818–823, https://doi.org/10.1109/PERCOMW.2018.8480168

[34] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-Defined Fog Network Architecture for IoT," Wirel. Pers. Commun., vol. 92, no. 1, pp. 181–196, Jan. 2017, https://doi.org/10.1007/s11277-016-3845-0

[35] V. B. C. Souza, W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in 2016 IEEE International Conference on Communications (ICC), May 2016, no. May, pp. 1–5, https://doi.org/10.1109/ICC.2016.7511465

[36] M. Haj Qasem, A. Abu-Srhan, H. Natoureah, and E. Alzaghoul, "Fog Computing Framework for Smart City Design," Int. J. Interact. Mob. Technol., vol. 14, no. 01, p. 109, Jan. 2020, https://doi.org/10.3991/ijim.v14i01.9762

[37] Z. Mahmood and M. Ramachandran, Fog Computing: Concepts, Frameworks and Technologies. Cham: Springer International Publishing, 2018.

[38] A. Karamoozian, A. Hafid, and E. M. Aboulhamid, "On the Fog-Cloud Cooperation: How Fog Computing can address latency concerns of IoT applications," in 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Jun. 2019, pp. 166–172, https://doi.org/10.1109/FMEC.2019.8795320

[39] H. F. Atlam, R. J. Walters, and G. B. Wills, "Internet of Things: State-of-the-art, Challenges, Applications, and Open Issues," Int. J. Intell. Comput. Res., vol. 9, no. 3, pp. 928–938, Sep. 2018, https://doi.org/10.20533/ijicr.2042.4655.2018.0112

[40] S. Hoque, M. S. De Brito, A. Willner, O. Keil, and T. Magedanz, "Towards Container Orchestration in Fog Computing Infrastructures," Proc. - Int. Comput. Softw. Appl. Conf., vol. 2, no. July, pp. 294–299, 2017, https://doi.org/10.1109/COMPSAC.2017.248

[41] S. P. Singh, A. Nayyar, H. Kaur, and A. Singla, "Dynamic Task Scheduling using Balanced VM Allocation Policy for Fog Computing Platforms," Scalable Comput. Pract. Exp., vol. 20, no. 2, pp. 433–456, May 2019, https://doi.org/10.12694/scpe.v20i2.1538

[42] J. Luo et al., "Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT," Futur. Gener. Comput. Syst., vol. 97, pp. 50–60, 2019, https://doi.org/10.1016/j.future.2018.12.063

[43] A. Ahmed and G. Pierre, "Docker Container Deployment in Fog Computing Infrastructures," in 2018 IEEE International Conference on Edge Computing (EDGE), Jul. 2018, pp. 1–8, https://doi.org/10.1109/EDGE.2018.00008

[44] N. D. Nguyen, L.-A. Phan, D.-H. Park, S. Kim, and T. Kim, "ElasticFog: Elastic Resource Provisioning in Container-Based Fog Computing," IEEE Access, vol. 8, pp. 183879–183890, 2020, https://doi.org/10.1109/ACCESS.2020.3029583

[45] T. 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, Jun. 2018, https://doi.org/10.3390/s18061798

[46] E. Cuervo et al., "MAUI: Making Smartphones Last Longer with Code Offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10, 2010, no. March 2014, p. 49, https://doi.org/10.1145/1814433.1814441

[47] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "Performance comparison between container-based and VM-based services," in 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Mar. 2017, vol. 2, pp. 185–190, https://doi.org/10.1109/ICIN.2017.7899408

[48] L. Yin, J. Luo, and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," IEEE Trans. Ind. Informatics, vol. 14, no. 10, pp. 4712–4721, Oct. 2018, https://doi.org/10.1109/TII.2018.2851241

[49] S. Luo, X. Chen, Z. Zhou, X. Chen, and W. Wu, "Incentive-Aware Micro Computing Cluster Formation for Cooperative Fog Computing," IEEE Trans. Wirel. Commun., vol. 19, no. 4, pp. 2643–2657, Apr. 2020, https://doi.org/10.1109/TWC.2020.2967371

[50] P. Wiener, P. Zehnder, and D. Riemer, "Towards Context-Aware and Dynamic Management of Stream Processing Pipelines for Fog Computing," in 2019 IEEE 3rd International

Conference on Fog and Edge Computing (ICFEC), May 2019, pp. 1–6, https://doi.org/10.1109/CFEC.2019.8733145

[51] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications," in 2019 IEEE Conference on Network Softwarization (NetSoft), Jun. 2019, pp. 351–359, https://doi.org/10.1109/NETSOFT.2019.8806671

[52] D. Wu et al., "A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing," J. Manuf. Syst., vol. 43, no. 2017, pp. 25–34, Apr. 2017, https://doi.org/10.1016/j.jmsy.2017.02.011

[53] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource Management Approaches in Fog Computing: a Comprehensive Review," J. Grid Comput., vol. 18, no. 1, pp. 1–42, Mar. 2020, https://doi.org/10.1007/s10723-019-09491-1

[54] C. Wobker, A. Seitz, H. Mueller, and B. Bruegge, "Fogernetes: Deployment and management of fog computing applications," in NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, Apr. 2018, pp. 1–7, https://doi.org/10.1109/NOMS.2018.8406321

[55] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A Survey on the Computation Offloading Approaches in Mobile Edge/Cloud Computing Environment: A Stochastic-based Perspective," J. Grid Comput., vol. 18, no. 4, pp. 639–671, Dec. 2020, https://doi.org/10.1007/s10723-020-09530-2

[56] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," IEEE Commun. Surv. Tutorials, vol. 22, no. 4, pp. 2489–2520, 2020, https://doi.org/10.1109/COMST.2020.3011208

[57] I. Azimi, A. Anzanpour, A. M. Rahmani, P. Liljeberg, and T. Salakoski, "Medical warning system based on Internet of Things using fog computing," in 2016 International Workshop on Big Data and Information Security (IWBIS), Oct. 2016, pp. 19–24, https://doi.org/10.1109/IWBIS.2016.7872884

[58] H. Huang, Y. Cai, and H. Yu, "Distributed-neuron-network based Machine Learning on Smart-gateway Network Towards Real-time Indoor Data Analytics," in Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016, pp. 720–725, https://doi.org/10.3850/9783981537079_0531

[59] F. A. M. Bargarai, A. M. Abdulazeez, V. M. Tiryaki, and D. Q. Zeebaree, "Management of Wireless Communication Systems Using Artificial Intelligence-Based Software Defined Radio," Int. J. Interact. Mob. Technol., vol. 14, no. 13, p. 107, Aug. 2020, https://doi.org/10.3991/ijim.v14i13.14211

[60] S. Chen et al., "Butler, Not Servant: A Human-Centric Smart Home Energy Management System," IEEE Commun. Mag., vol. 55, no. 2, pp. 27–33, Feb. 2017, https://doi.org/10.1109/MCOM.2017.1600699CM

[61] S. Feng, P. Setoodeh, and S. Haykin, "Smart Home: Cognitive Interactive People-Centric Internet of Things," IEEE Commun. Mag., vol. 55, no. 2, pp. 34–39, Feb. 2017, https://doi.org/10.1109/MCOM.2017.1600682CM

[62] M. Bhatia, "Fog Computing-inspired Smart Home Framework for Predictive Veterinary Healthcare," Microprocess. Microsyst., vol. 78, no. July, p. 103227, Oct. 2020, https://doi.org/10.1016/j.micpro.2020.103227

[63] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-Aware Optimization of Distributed Learning for Fog Computing," in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Jul. 2020, vol. 2020-July, pp. 2509–2518, https://doi.org/10.1109/INFOCOM41043.2020.9155372

[64] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration Modeling and Learning Algorithms for Containers in Fog Computing," IEEE Trans. Serv. Comput., vol. 12, no. 5, pp. 712–725, 2019, https://doi.org/10.1109/TSC.2018.2827070

[65] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. S. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," Digit. Commun. Networks, vol. 5, no. 1, pp. 3–9, Feb. 2019, https://doi.org/10.1016/j.dcan.2018.10.008

[66] M. V. Ngo, Q. D. La, D. Leong, T. Q. S. Quek, and H. Shin, "User Behavior Driven MAC Scheduling for Body Sensor Networks: A Cross-Layer Approach," IEEE Sens. J., vol. 19, no. 17, pp. 7755–7765, Sep. 2019, https://doi.org/10.1109/JSEN.2019.2915635

[67] J. Patman, M. Alfarhood, S. Islam, M. Lemus, P. Calyam, and K. Palaniappan, "Predictive analytics for fog computing using machine learning and GENI," INFOCOM 2018 - IEEE Conf. Comput. Commun. Work., pp. 790–795, 2018, https://doi.org/10.1109/INFCOMW.2018.8407027

[68] Q. Li, L. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "SMDP-Based Coordinated Virtual Machine Allocations in Cloud-Fog Computing Systems," IEEE Internet Things J., vol. 5, no. 3, pp. 1977–1988, 2018, https://doi.org/10.1109/JIOT.2018.2818680

[69] M. Lavassani, S. Forsström, U. Jennehag, and T. Zhang, "Combining Fog Computing with Sensor Mote Machine Learning for Industrial IoT," Sensors, vol. 18, no. 5, p. 1532, May 2018, https://doi.org/10.3390/s18051532

[70] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning," in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Jan. 2019, pp. 0305–0310, https://doi.org/10.1109/CCWC.2019.8666450

[71] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for IoT," Appl. Soft Comput. J., vol. 72, pp. 79–89, 2018, https://doi.org/10.1016/j.asoc.2018.05.049

[72] M. Yang, T. Zhu, B. Liu, Y. Xiang, and W. Zhou, "Machine Learning Differential Privacy with Multifunctional Aggregation in a Fog Computing Architecture," IEEE Access, vol. 6, pp. 17119–17129, 2018, https://doi.org/10.1109/ACCESS.2018.2817523

[73] B. Huang, X. Cheng, Y. Cao, and L. Zhang, "Lightweight Hardware Based Secure Authentication Scheme for Fog Computing," in 2018 IEEE/ACM Symposium on Edge Computing (SEC), Oct. 2018, pp. 433–439, https://doi.org/10.1109/SEC.2018.00059

[74] X. An, X. Zhou, X. Lü, F. Lin, and L. Yang, "Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC," Wirel. Commun. Mob. Comput., vol. 2018, pp. 1–10, 2018, https://doi.org/10.1155/2018/7472095

[75] A. Abeshu and N. Chilamkurti, "Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing," IEEE Commun. Mag., vol. 56, no. 2, pp. 169–175, Feb. 2018, https://doi.org/10.1109/MCOM.2018.1700332

[76] C. Xu and G. Zhu, "Intelligent manufacturing Lie Group Machine Learning: real-time and efficient inspection system based on fog computing," J. Intell. Manuf., vol. 32, no. 1, pp. 237–249, Jan. 2020, https://doi.org/10.1007/s10845-020-01570-5

[77] M. Taneja, J. Byabazaire, N. Jalodia, A. Davy, C. Olariu, and P. Malone, "Machine learning based fog computing assisted data-driven approach for early lameness detection in dairy cattle," Comput. Electron. Agric., vol. 171, no. March, p. 105286, 2020, https://doi.org/10.1016/j.compag.2020.105286

[78] N. C. Luong, Y. Jiao, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "A Machine-Learning-Based Auction for Resource Trading in Fog Computing," IEEE Commun. Mag., vol. 58, no. 3, pp. 82–88, Mar. 2020, https://doi.org/10.1109/MCOM.001.1900136

[79] S. Memon and M. Maheswaran, "Using machine learning for handover optimization in vehicular fog computing," in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Apr. 2019, vol. Part F1477, pp. 182–190, https://doi.org/10.1145/3297280.3297300

[80] X. Chen, S. Leng, K. Zhang, and K. Xiong, "A machine-learning based time constrained resource allocation scheme for vehicular fog computing," China Commun., vol. 16, no. 11, pp. 29–41, Nov. 2019, https://doi.org/10.23919/JCC.2019.11.003

[81] I. Merelli et al., "Low-power portable devices for metagenomics analysis: Fog computing makes bioinformatics ready for the Internet of Things," Futur. Gener. Comput. Syst., vol. 88, pp. 467–478, 2018, https://doi.org/10.1016/j.future.2018.05.010

[82] O'Donovan, C. Gallagher, K. Bruton, and D. T. J. O'Sullivan, "A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications," Manuf. Lett., vol. 15, pp. 139–142, 2018, https://doi.org/10.1016/j.mfglet.2018.01.005

9 Authors

Fady E. F. Samann holds an M.Sc. degree in Electronic Communications and Computer Engineering from the University of Nottingham-UK. Currently, he is studying for a Ph.D. in Information and Communications Technologies at the Duhok Polytechnic University, Kurdistan Region, Duhok, Iraq. He published articles in multiple fields.

Adnan Mohsin Abdulazeez received the B.Sc. degree in electrical and electronic engineering, the M.Sc. degree in computer and control engineering, and the Ph.D. degree in computer engineering. He is currently the President of Duhok Polytechnic University and a Professor of computer engineering and science. He has been awarded the title of a Professor since 2013. He is keen to carry out his administrative and academic responsibilities simultaneously. He supervised and still supervises a large number of master and doctoral students. Besides, he focuses his attention on publishing scientific researches in esteemed international scientific journals.

Shavan Askar (Associate Professor of Computer Networks) is currently a faculty member at the College of Engineering\EPU. He received his Ph.D. degree in Electronic Systems Engineering from the University of Essex\UK in 2012. He obtained his MSc (2003) and BSc (2001) degrees from the Control and Systems Engineering Dept. of the University of Baghdad. His interest fields include IoT, SDN, Optical Networks, and 5G. While he was in the UK, he worked as a Researcher on two European projects; the MAINS project and the ADDONAS project.

Article submitted 2021-01-19. Resubmitted 2021-04-11. Final acceptance 2021-04-13. Final version published as submitted by the authors.