AI-Enabled Secure Microservices in Edge Computing: Opportunities and Challenges

Firas Al-Doghman, *Member, IEEE*, Nour Moustafa, *Senior Member, IEEE*, Ibrahim Khalil, *Senior Member, IEEE*, Nasrin Sohrabi, *Student Member, IEEE*, Zahir Tari, *Senior Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—The paradigm of edge computing has formed an innovative scope within the domain of the Internet of Things (IoT) through expanding the services of the cloud to the network edge to design distributed architectures and securely enhance decision-making applications. Due to the heterogeneous, distributed and resource-constrained essence of edge Computing, edge applications are required to be developed as a set of lightweight and interdependent modules. As this concept aligns with the objectives of microservice architecture, effective implementation of microservices-based edge applications within IoT networks has the prospective of fully leveraging edge nodes capabilities. Deploying microservices at IoT edge faces plenty of challenges associated with security and privacy. Advances in Artificial Intelligence (AI) (especially Machine Learning), and the easy access to resources with powerful computing providing opportunities for deriving precise models and developing different intelligent applications at the edge of network. In this study, an extensive survey is presented for securing edge computing-based AI Microservices to elucidate the challenges of IoT management and enable secure decision-making systems at the edge. We present recent research studies on edge AI and microservices orchestration and highlight key requirements as well as challenges of securing Microservices at IoT edge. We also propose a Microservices-based edge computing framework that provides secure edge AI algorithms as Microservices utilizing the containerization technology to offer automated and secure AI-based applications at the network edge.

Index Terms—Edge computing, microservices, edge security, edge privacy, edge AI

1 Introduction

The Internet of Things (IoT) is a distributed inter-network of embedded devices which connect via communication techniques [1] and is empowered with limited communication, storage, and computation capabilities for offering a broad domain including applications and services. Microservices are an arising paradigm for developing service-oriented software. They offer the development of applications which include a set of small services operating on processes [2]. The concepts of containers and Microservices are closely associated in literature [3]. Containers are environments that in some ways resemble Virtual Machines (VMs), with some significant differences [4]. Recent literature points out that Microservices have been successfully applied to implement cloud computing applications. The deployment of edge applications (also known as fog applications) using Microservices has a great interest in the research community [5].

There is a decrease in the efficiency of the centralised cloud computing paradigms regarding the processing and analysis of big data, because of the difficulties in transferring such data amount, which has limited the performance of edge networks [6]. As edge computing brings computing resources to front-end devices, the bandwidth demands are reduced, enhancing network performance [7]. Edge computing paradigm utilises the resources of the connected nodes in order to collaborate in reaching mutual aims in the architecture of distributed software [8].

As edge nodes generally perform in an unattended environment and are connected over wireless networks, cyber-attackers would access these nodes or access a communication channel to gain private information via eavesdropping or unintended discovery techniques [9]. Also, data privacy at edge computing has to be outsourced by third parties, leading to the possibility of illegal data operations and data leakage, so the confidentiality and integrity of data can not be ensured [10]. With the numerous amount of unsecured devices connected within IoT networks, the potential risks are high [11]. IoT systems have to autonomously and continuously survive and adapt, considering safety as a main priority. Additionally, the IoT setting has introduced new attack surfaces resulted from the interconnected and interdependent IoT environments. Therefore, IoT systems face a higher security risk compared to other computing systems, and the conventional security solutions would not be effective for protecting edge networks [12] [13]. It is quite challenging

- *Firas Al-Doghman and Nour Moustafa are with the School of Engineering and Information Technology, the University of New South Wales @ ADFA, Canberra 2612, Australia. E-mail: f.aldoughman@adfa.edu.au, nour.moustafa@uns.edu.au.*
- *Ibrahim Khalil, Zahir Tari, and Nasrin Sohrabi are with RMIT University, Melbourne, VIC 3000, Australia. E-mail: {Ibrahim.Khalil, Zahir.Tari} @rmit.edu.au, 3732890@student.rmit.edu.au.*
- *Albert Y. Zomaya is with the University of Sydney, Camperdown, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 7 Apr. 2021; revised 13 Dec. 2021; accepted 25 Feb. 2022. Date of publication 1 Mar. 2022; date of current version 10 Apr. 2023.
This work was supported in part by UNSW Canberra at ADFA and in part by the Defence Science and Technology Group (DSTG) under Grant RG201280.
(Corresponding author: Nour Moustafa.)
Recommended for acceptance by K. Ren.
Digital Object Identifier no. 10.1109/TSC.2022.3155447

and inadequate to apply the current security protection methods like authentication, encryption, network security access control, and application security for wide-scale systems containing various connected nodes, including inherent vulnerabilities in every part [14].

For configuring the most appropriate architecture for edge computing, modular properties of microservices should be considered to enhance the performances of deploying applications, especially AI systems [15]. In a distributed IoT system, the instances of the microservices are scaled based on the load, which make the number of microservices and their locations to be changed dynamically. Several considerations need to be taken when deploying edge microservices including service discovery, inter-service communication, data integrity, security, monitoring and health check, and quality assurance [16]. The heterogeneity of IoT data creates a serious challenge for existing data processing techniques when they handle heterogeneous data sources in edge networks nodes. Therefore, new techniques are needed so as to harness the value of data and enhance the manners of processing heterogeneous data sources. In this context, Artificial Intelligence (AI) is seen as one of the best computational models that support embedded intelligence within IoT environment [17]. AI could assist in inferring valuable knowledge from data generated by a device or human, and deploying AI through microservices will enhance the process of learning large-scale and heterogeneous data sources at edge nodes [18]. AI microservices can be leveraged within IoT systems for supporting intelligent services and enable the implementation of modular security techniques.

Microservices has become a vital technology for providing a network service because of its management flexibility and programmability. They provide a distributed platform, where every entity works separately, encompassing control, data, and application entities [19]. Novel microservices and the implementations of edge IoT bring a fresh communication view for efficient service providing [20]. An architectural style based on microservices is a method to develop an application as a set of smaller services, with each functioning in its own process and being communicated through lightweight procedures. Such services are based on business power and can be implemented independently [21]. However, despite the great interest in microservices [22], it is not possible to observe proposals that illustrate recurring problems (architecture patterns) or design decisions (architecture tactics) that are emerging in such direction. Architecture patterns describe a high-level system structure with its associated behaviour [23]. Many of the benefits and responsibilities of these patterns are related to Quality Attributes (QA) [24]. Based on the above considerations, The orientation in this paper is towards the discussion of executing AI-enabled microservices in edge computing, with a concentration on providing communication security and data privacy services to IoT networks. We discuss a secure AI framework based on microservices at edge networks.

1.1 Existing Surveys

Various surveys have been conducted to separate microservices, security, and AI methods within IoT-edge ecosystems. In this section, a summary of the existing surveys and a comparison with this paper's work are conducted in the last five years. To the best of our knowledge, the majority of surveys have not concentrated on edge AI models deployed as microservices in IoT systems. Moreover, the existent surveys are either very specific to the topic or do not include a full scope of microservices AI models, security, and privacy within IoT networks. Current studies have also discussed ML/DL-based solutions or security at the network Edge investigating the existing cloud solutions or those supplied by the newly emerging techniques. However, surveys covering AI applications as microservices at IoT-edge ecosystems are scarce.

Several surveys covering multiple sides of Edge computing technologies and services for IoT have been published throughout the last decays including Machine and Deep Learning (ML/DL) [25], [26], [27], [28], Federated Learning (FL) [29], Security practices [30], distributed models [31], and Resources and services management [32], [33]. A few research surveys address microservices (MS) as an implicit perspective of IoT-Edge [34]. Generally, most of these surveys has dealt with an individual side of IoT Edge computing such as security, DL, or resource management. However, there is a lack of survey papers addressing the deployment of AI models as microservices on edge devices, considering the architecture, virtualisation, security and privacy perspectives.

Table 1 outlines the latest surveys on the IoT-Edge computing taxonomy from the viewpoint of ML/DL, FL, security, resources, and services management technologies, and architectures. The available literature gives an essential understanding of fog or edge computing and their integration with resource management or AI practices like DL. Moreover, most of them skip a crucial implementation infrastructure of microservices and their deployment on a container alike method. In [34], the authors considered various application scenarios and fundamental techniques for edge intelligence and the crucial challenge of expanding DL from the cloud to edge computing; however, they did not consider the containerized microservices as their main architecture segment at edge computing, which has become an essential component for applications with latency-sensitive properties.

1.2 Contributions and paper Organisation

This paper illustrates the existing literature concerning microservices, AI models such as ML and DL, data privacy and network security within Edge-IoT nodes. Additionally, the concept of AI-IoT as a future IoT system is quite novel and is not adequately handled in existing research studies [38]. As previously mentioned, most of the existing surveys on Edge AI applications have not considered the implementation of microservices in the context of IoT Edge Computing, while surveys on IoT security have not considered the challenges of implementing microservices or the possibility of applying microservice architecture at edge network. This is our motivation to write this review and have the following contributions:

- We present the evolution of AI microservices through conveying relevant literature on microservices, AI, edge Computing and their security aspects. Additionally, the main requirements of the various underlying technologies are introduced which are crucial in envisioning the concept of secure AI microservices.

TABLE 1
Summary of Related IoT-Edge Ecosystems Surveys, Considering AI and Microservices if Exist

Papers	Technology Scope	Topic Focus and Taxonomy	Reviewed Edge AI MS?	Security Practices
[25]	ML/DL	ML/DL methods for IoT security	x	ML/DL and RL
[30]	Digital Forensics	Consider vulnerability issues within IoT systems and examine the Digital Forensics approaches	x	Blockchain
[26]	DL	discuss various techniques for overcoming edge AI communication issues & introduce efficient methods for training & inference processes at the network edge	x	Privacy-preserving ML with vertically partitioned data
[34]	DL	explain different applicable use cases & basic methods for edge intelligence & main challenges of expanding DL to the edge	✓	FL/DRL
[35]	Edge security and privacy	summarises data security & privacy requirements & challenges in edge computing, also present recent data security & privacy techniques with their cryptography-based techniques, analysis & solution were explored	x	Cryptography
[31]	Distributed Edge Models	illustrates a thematic taxonomy of Decentralized Consensus for edge-centric IoT and a debate about its critical analysis and research issues	x	Blockchain /DAG-IoT combination
[27]	DRL	survey existing works on Deep Reinforcement Learning and highlight related open issues	x	x
[29]	FL	research resource allocation mothods for FL along with their security & privacy issues and the applications of FL in mobile edge network optimization to reduce communication costs	x	Differential Privacy and Encryption FL solutions
[36]	DL	research multiple architectures and techniques that speed up DL inference at the edge server, as well as distributively training DL models on edge devices considering privacy and describing several application domains	x	Secure two-party communication
[37]	ML/DL	explore the current ML/DL-based techniques within IoT networks	x	ML security applications
[28]	ML	survey existing work utilises ML addressing MEC issues, detailing such issues and their solutions, introducing special ML algorithms exploring their methods & discussing their strengths & weaknesses	x	MEC privacy preserving
[32]	resource sharing models	address the taxonomy of resource sharing trends and models in a generalised end-to-end Future Communication Network's architecture, presenting hierarchical view of the issues and solutions, per model: business, geographic, and technical; and per layer: infrastructure, orchestration, and service	x	Authentication, Access Control, and Integrity Assurance

- We examine security and privacy challenges of AI microservices and introduce detailed classifications about the notable attack issues which could utilize vulnerabilities in the existing infrastructure of edge computing.
- We propose an AI microservices framework that demonstrates secure microservices for the improved deployment of intelligent resource management at edge networks.
- we introduce open issues and future research directions of this work, including data encryption, cross-domain authentication, multi-authority access control system, fine-grained privacy-preservation microservices at the edge.

The remainder of this paper is organized as what follows. Next section 2 presents a background on the main categories of this work related to AI microservices and edge computing. After that, in section 3, we show existing research studies related to a microservices architecture, AI at edge Computing, AI microservices at IoT ecosystems, and microservices security challenges. Section 4 presents a secure AI-Edge framework to provide protected AI algorithms as microservices. After that, lessons learnt and future research directions are introduced in Section 5. Finally, Section 6 concludes the work.

2 Background and Fundamentals

2.1 Microservices

Microservices are small autonomous units of executable code, like functions, combined to form a complete application [39]. They are distributed, can be called remotely, and help to build robust and extendable applications by decentralizing them for avoiding a single point of failure. Because of their modular architecture, they allow easy augmentation of applications, lessen the impact of errors and ease the correction process. By design, they promote a software development paradigm, where common functions are called as a service, and dynamic scaling becomes possible [40].

Additionally, as the microservices are autonomous, with an application calling it like it would a regular function, and as microservices are hosted remotely, either on VMs or in containers, applications based are independent of the underline technology and programming language used to code a Microservice that the application calls. Furthermore, through microservices, applications can be designed to utilize various databases in the back-end, relying for example on cloud, relational and NoSQL databases. Microservices are mostly deployed with the use of Docker, Kubernetes or other Infrastructures [39], [40].

2.1.1 Monolithic versus Microservice architectures

Typically, application software has been developed as monolithic architecture where it is deployed as a single solution. In this architecture design, few programming languages can be used to make a single application or process consists of multiple classes, procedures and packages, in which the whole application or process is executed in one server irrespective of the application requirements [41]. This design is module independent, uniform standards/stack, and simple in developing, testing, and the horizontal scaling [42]. However, it has poor scalability as any overload in its functions can create a bottleneck, and any change in one function can affect other dependent functions. Furthermore, this design rises the complication of redeployment and maintenance, and the difficulty of solving the physical heterogeneity problem [41], [43].

To overcome these defects, Microservice architecture has recently been adopted in which the single solution or application is divided into small manageable services. Every single Microservice performs a single function and independent from others. Besides, any programming language can be used to realize each provided Microservice [41]. Thus, Microservice Architecture can be defined as fragmenting a Software System into independently deployable autonomous components communicating through lightweight, language-agnostic means and collaborating to attain the goal of the particular business. Additionally, it utilizes the divide and conquers technique to address software systems' complexity similar to Modular Monolithic architecture wherein a sophisticated Software system is split into several microservices communicating through external Interfaces.

In contrast to monolithic architecture, a microservices architecture reduces the complexity of redeploying and maintenance as its microservices are deployed independently and can easily be modified and changed whereas in Modular Monolithic all Modules normally are deployed as a whole. Moreover, the microservices architecture supports many technology stacks and fault tolerance which in return makes it more scalable [44].

2.1.2 Advantages and Disadvantages of Microservices

The design of the microservice architecture has many advantages which encouraged many corporations and individuals to adopt it. The following are some of the most important advantages:

- Application Scaling – This is one of the main strengths of microservices. Since microservices are usually stateless, they can offer fast horizontal scaling if they are carefully deployed. Additionally, microservices could be performed utilizing various programming languages and then they cooperate to deliver the service [45] [46].
- Rapid Development – It is ordinarily faster to add new features in microservices because they are normally are small in size [47].
- Development Scaling – the Scalability of development is very efficient as microservices are autonomous and could be developed independently which enables developers to elaborate on various microservices autonomously. Also, as microservices are small in size, their development would take less time [48].
- Release Cycle – As a Microservice is deployed independently, the software release cycle would be much smaller in microservice applications [49].
- Modularity – In a microservice architecture, microservices are decentralized, small in size, bounded by contexts, messaging enabled, independently deployable, and autonomously developed which make it naturally enforces a modular structure [50] [51].
- Modernization – It is feasible to Modernise Microservices since they are loosely coupled and only communicate through language-agnostic method within the system which enables the easy replacement of individual Microservice without affecting the entire settings [52].

Microservice Architecture has also some disadvantages when it is treated without proper consideration for each particular kind of issue within a software application. The following are some disadvantages of Microservice Architecture:

- Design Complexity – In a microservice architecture, many solutions are possible based on the applications and use cases. So, when applying the wrong solution for a not matching application type/size, the microservice architecture is destined for failure. Additionally, the design of microservices is difficult as there are a lot of moving parts in comparison with monoliths. Generally, an inadequate designed Microservice is inferior to a Monolith [53].
- Distributed Systems Complexity – Microservices are usually distributed systems that make them more complex and having their unique challenges set in comparison with single-machine systems [54]. In general, some of the main issues that can arise in a distributed microservices system compared to the single-machine system are: higher overall system latency, higher Operational complexities, and the failure of Network or Individual Node could bring down the entire system [55].
- Operational Complexity – When decomposing a complex Monolithic application into several microservices, its complexity usually shifts from source code to operations. As moving from one system to multiple systems, handling simple operations such as Logging and Monitoring became more complex. Other crucial operations in microservices that are quite complex include complete System test and Tracing, which measure the service request performance and latency of each microservice. In addition to the complexity, new infrastructures are required to achieve resiliency and service discovery [56].
- Security – One of the most challenging aspects of microservices architecture is security. Securing a single Software Application is arduous, in consequence securing a lot of microservices, which are usually Distributed Systems, is much harder [57].
- Data Sharing and Data Consistency – To achieve the business goal, microservices are required for sharing data. This is very challenging since ideally, each Microservice needs to have its Data Store. Another challenge is data consistency since most of the recent NoSQL databases only provide eventual consistency in which it requires accurate design [52].

- Communication Complexities – As microservices attain modularity and development autonomy through network boundaries, the services could only transmit through the physical network leading to higher latency. There are a lot of methods where microservices can perform communication among themselves, selecting the correct application-based Communication technique comes with big challenges in Microservice Architecture [58].

2.2 AI and Edge Computing

2.2.1 Artificial Intelligence (AI)

AI has come to be an important field nowadays, especially after the latest developments and successes of deep learning in multiple areas including cyber security, computer vision as well as natural language processing. AI is a set of algorithms that gradually learn and improve dealing with their task and make decisions maximizing the probability of successfully attaining its goals[59]. One of the main subsets of AI is Machine Learning (ML) which its fundamental application is big data analysis accuracy and quality. The algorithms related to Machine Leaning detect data patterns of a base dataset and then the ML trained models follow those patterns to predict the useful insights of a new dataset [60]. The domain of ML learning has two major sub-domains: supervised learning and unsupervised learning. In the supervised learning, the input and the output data labels produced by humans is utilized for structuring the data, whereas unsupervised learning utilizes unlabeled data. In supervised learning, ML models include linear or logistic regression training, random decision forests, SVM, and Naive Bayes algorithms. On the other hand, in unsupervised ML, the AI model groups unsorted data following differences and similarities even when no categories are provided [61].

Deep Learning (DL) has latterly been very effective throughout various application fields, including computer vision, natural language processing, and the analysis of Big Data [62]. The main defining characteristics of DL are high precision and high resource consumption [63]. High precision results from the cost of high memory and computational requirements the deep learning training and inference phases need. When a DL model is trained, huge memory and computations are required as millions of parameters which require to be repeatedly refined through several time intervals. Also, the inference is computationally costly because of the high input data dimensionality as well as the number of computations required to be performed on input data [36]. The algorithms of DL employ several layers of nodes having different weights forming what is called Deep Neural Networks (DNNs). They include input and output nodes, and in between, there are multiple layers. To reach the required outcomes at scale, the model is to be trained through the adjustment of the weight of every DNN node so that the outcome can be influenced. Every time the model weights are readapted, the model better understands the needed object features. The usefulness of this operation is that the features do not require to be defined at the beginning and the model would gradually learn to point out the best results by itself [64].

2.2.2 Edge AI

Edge computing is a feasible paradigm that meets the challenges of scalability, latency, and privacy [65]. Within edge computing, a set of computing resources equips computational capabilities at the front-end nodes [66]. Recently, DL is used within lots of applications and boost its outstanding performance, and lots of GPUs, TPUs, or FPGAs are in demand for deployment at the cloud to handle DL service inquiries. However, as edge computing architecture encompasses many distributed devices, it could be employed to better fit DL. For sure edge nodes in most cases hold limited computation power in comparison with the cloud. Consequently, applying DL on edge computing devices is not easy and demands a thorough realization of DL models, as well as the design and deployment features of edge computing [36]. To reduce the costs of resources, DL models can be optimized and their weights can be quantized. Generally, to achieve model optimization, model redundancies can be used. The optimization process should re-design or transform DL models and suitably shape them to fit in edge nodes [67]. However, the main issue is to make sure there is no remarkable dropping in model precision after optimizing it.

Federated learning can fit the norm of edge computing, as it trains machine learning algorithms like DNN on several local datasets to hold within local devices without explicitly interchanging samples of data. Its main concept involves the training of local models over local data samples as well as the interchanging of parameters such as weights and biases of DNN among the local devices at a particular frequency aiming to create a global model which will be shared by all devices [68]. FL comprises collaboratively train DNN models at end nodes. Mainly, the process of training FL models involves two phases that are: local training at end nodes and global aggregation of upgraded parameters at FL server. Thus, the above description indicates the suitability of implementing FL at Edge networks as it clearly can leverage on the growing computation power of data gathered via distributed front-end nodes [69].

2.3 Microservices-enabled AI Technology

Recently, AI has a great impact on many aspects of our life. However, it still needs the accurate tools to package up this intelligence for ubiquitous, fast, and flexible implementation and deployment. Deep learning techniques have come into the environment of cloud services via integrating the techniques into tools supporting microservices architectures. This indicates the increased popularity to develop cloud applications in the forms of modular, reusable, and carefully scoped functions. Within microservices settings, every function is executed within an individual container (Docker as an example). Additionally, every individual microservice interoperates with the other microservices through RESTful APIs in a lightweight and loosely coupled manner. Deep learning is growingly mounted on containerized microservices that perform in compound multi-clouds. Ideally, it is implemented via abstract serverless interfaces which allow microservices to implement transparently on the infrastructures of the cloud without requiring the developers to be informed whether the resources are being supplied from where or how. Thus, the serverless back-end allows the infrastructure to automatically provide real-time the essential resources for Microservices such as bandwidth, storage, compute power, and any other distributed resources.

In moving to this mode of synthesizing applications, the developers do not consider pre-provisioning infrastructure,

for example, servers or operations. Alternatively, they would be able to just concentrate on coding and modeling. However, a back-end middleware fabric is required to interoperate seamlessly within applications running complex deep-learning to achieve transactional rollback, reliable messaging, and long-running orchestration intelligence like what Kubernetes provides. To deploy AI capabilities at the network edge, machine learning algorithms should be infused within the components of the edge platform [70]. Therefore, edge nodes should be equipped with tools such that intelligent services that are allowed to be constructed as data-driven microservices [71]. Existing monolithic cloud-based AI services cannot meet the needs for real-time applications. Instead of transferring data to the cloud data centers for incorporating the capability and intelligent decision making, AI models could be deployed and executed near to data sources through factoring the functionality of AI into sub-functions which could be performed as distributed microservices [72].

The AI algorithms should be deployed and implemented near loT front-end devices via distributing the functionality of AI into sub-functions which can be implemented as distributed microservices [73]. When developers consider building DL microservices to be executed within IoT, the complexity of the back-end interoperability fabric increases to a higher extent than it is in the cloud. The reason for that is because DL is becoming an embedded intelligence of the entire IoT front-end devices and also a service that is provided to applications by IoT hubs and centralized cloud services. DL services should be considered as "micro" in the direction of being narrowly scoped, front-end devices in the IoT environment being the executors of algorithms will become more and more "micro" in their resource limitations. Embedded DL microservices within IoT will manage the plenteous flows of sensor readings gathered by front-end devices in real-time. Embedded DL Microservices within IoT will manage the plenteous flows of sensor readings gathered by front-end devices in real-time.

Microservices can manage pattern sensing applications like motion detection, video recognition, natural language and clickstream processing, and many more on which IoT apps relay. Consequently, each object of any kind within IoT would be permeated with IoT management properties such as environmental awareness, continuous data-driven intelligence, and situational autonomy. [74]. When developing and composing IoT DL applications, developers need a middleware back-end that spreads microservices to execute at the network front-end devices. To support such IoT use cases, the architectures of microservices will advance to support the fog/edge computing [75]. Within this paradigm, developers create DL and any other distributed intelligence by utilizing microservices APIs as well as serverless back-ends that perform distribution, parallelization, and optimization to the massive front-end nodes of edge [76]. However, we should keep in mind that the DL analytics zone will be enabled by containerized microservices for pervading the whole way from the cloud down to edge nodes like sensors, smart devices, and gateways.

2.4 Containers-based Microservices in Edge Computing

Edge computing has been emerged to perform data processing locally at front-end nodes and to resolve the performance issues of the network. Containers are programs that resemble VMs, to run various modules independently and flexibly [4], [77]. To begin with, containers can be installed on either VMs or bare-metal hosts, and require to have the same operating system as the host machine, for example, Linux containers are compatible with Linux or Unix machines. By sharing the same OS kernel with the host, containers demand less storage and fewer resources than traditional VMs at run-time because VMs require a separate copy of their OS's kernel. Containers are designed to be light-weight, compact, and easily transferable, enabling developers to more easily develop a container image that contains all the required programming resources, such as binaries, libraries, and upload this image to a container registry, from which it can then be easily deployed. these container registries are often distributed in remote geo-locations, with the container images being either public or private.

Edge containers give the organizations the ability to decentralize services through shifting the main elements of their application to the network edge [78]. When moving intelligence to the edge, network costs can be reduced and the response times will be increased. Edge containers are decentralized computing means placed closer to the front-end nodes to decrease the latency, improve the overall system functionality, and save bandwidth. Containers are software packages which deployed easily and the containerized applications are generally distributed with ease, which leads them to fit naturally for edge computing services.

Unlike cloud containers which operate at distant regional data centers, Edge containers are mainly situated at the edge closer to the end-user. As the location is the key difference, edge containers utilize similar tools as used in cloud containers enabling developers to employ their Docker proficiencies for edge computing [79]. Docker is a group of platform-as-a-service products that use OS-level virtualization to provide the capability of packaging and running application software in a form of container packages. Docker is mainly used for developing and running distributed applications in a much faster way and without depending on the underlying OS. The main advantages when deploying a Docker for distributed applications enhance network performance, less CPU overhead, ease portability, and versioning control [80].

Containers are isolated from each other and they can pack their software, configuration files, and libraries as well as communicate via well-defined channels among themselves. Docker has been applied in plenty of recent projects to perform various tasks. Some examples include employing Docker as a platform for distributed service providing fault tolerance [81], or acting as Docker-based gateways for IoT applications or testing architectures' scalability and flexibility via microservices [82]. Consequently, a Docker or any other container-based platform has adequately made its way through the market of distributed application development and it can be reasonably suited for cloud and edge computing. It also can be utilized for customizing IoT platform by offering services for data processing near the end-user. Nowadays, Docker is used for building microservices modules to divide tasks into stand-alone applications and then these microservices would operate together to achieve the aimed operation and deliver the overall service. Developing microservices in Docker demands novel thinking and

strategies, yet it makes unparalleled capabilities to build stable and scalable applications.

When building an application, which uses microservice architecture design, there is a need for decisions about the way that those microservices interact with one another. There are two main approaches for microservices interactions (also sometimes there would be a hybrid approach of them), Orchestration and Choreography (Reactive). Orchestration is the conventional method to handle interactions among various services within Service-Oriented Architecture (SOA). It usually contains one controlling entity that functions as the orchestrator or manager of all service interactions. Orchestration mainly accommodates a request/response pattern type providing a good method for application flow control when having synchronous processing. With service choreography (reactive pattern), there is no need for a central orchestration layer that coordinates the performance. The purpose of this is to avoid making dependencies among microservices i.e., giving every microservice the ability to stand on its own. The logic of what should happen and when is built into the individual microservices beforehand. Generally, the most popular way to deploy AI models as containerized microservices is with the support of using an orchestration approach such as Kubernetes on edge nodes. Kubernetes is a system of container orchestration employed to automate the administration, deployment, and scaling of microservice applications.

2.5 Security Aspects of AI-enabled Microservices in Edge

IoT techniques are complicated and hold integrative processes. Thus, it is challenging to maintain the security requirements within the IoT sophisticated attack site [83]. Solutions have to encompass comprehensive considerations to fulfill security requirements. Although IoT nodes generally perform within an unattended medium, an outsider might still physically access those nodes. IoT nodes are coupled ordinarily via wireless networks wherein an intruder might have the ability to access private data from communication channels through eavesdropping. They are not capable of supporting complicated security systems given due to their limitations in computing, communication, and power means [84]. Besides, the complexity in IoT security systems also comes from the unpredictable modes of interactions within the physical scope. IoT systems have to continuously modify and sustain in a particular and predictable way and maintain safety against threats within their environment [85].

To learn about what are 'normal' or 'abnormal' behaviors when exploring data within IoT, ML and DL techniques are to be utilized following the interactions among components and nodes in the environment to develop intrusion detection and prevention systems. Gathering and investigating data from each IoT system's section can determine normal interaction patterns and consequently identify malicious behaviors earlier using various cybersecurity systems, such as intrusion detection, threat intelligence, and digital forensics. Furthermore, DL processes could predict new attacks, which are usually mutations of former ones, due to their ability to intelligently predict upcoming unknown attacks through utilizing existing samples to learn from [86]. Therefore, IoT systems have to shift from only facilitate secure communication between nodes in the environment to a security-based intelligence paradigm administered by DL processes to fulfill effective and secure IoT systems.

Generally, the authentication process for microservices is complicated because most microservices utilize API gateways; thus it requires intelligent and lightweight access control and authentical methods. The API gateway provides a reverse proxy for redirecting/routing requests towards the endpoints of the inner microservices. The gateway offers an individual endpoint for client applications and then performs an internal mapping linking the requests to a set of inner microservices. However, having individual APIs permits these microservices to be reconfigured and updated individually. To maintain the security of microservices, their APIs have to be reliable, integrated, confidential, and available. For implementing intelligent security solutions for microservice architecture, there are four major domains: design, implementation, deployment, and management. With every domain, a set of security principles should be associated. These principles involve serving only the targeted services which are required by the user at a specific time, using standards and protocols for protection, perform multi-keyed cryptography for multiple layers of security, and handling high traffic demands. Through following such microservices security principles, a better outcome can be acquired to protect the processes against attacks.

3 Extant Literature

3.1 Edge-enabled AI Computing

AI techniques are being shifted to the edge of networks to meet the increasing demands for real-time response and decrease data congestion [87]. Various research studies have introduced the deployment of the machine and deep learning algorithms within edge computing nodes [88]. Such practices assist our research direction in deploying AI microservices at the network edge to achieve a fully dependent edge computing system. In this section, recent studies utilizing AI techniques at edge/fog computing will be introduced, as listed in Table 2.

3.1.1 Machine Learning Models

Kochoviski *et al.* [89] introduced a design considering big data applications in the civil engineering field to gather and analyze data from different sensors. Docker containers were utilized to implement the design of an orchestrator that was used to deliver network utilities near the front-end which minimizes the need for moving the big amount of data throughout the network having the potential for improving QoS along with privacy and security. Zhou *et al.* [90] argued that the stand-alone machine learning training models are not meeting the current needs and they proposed a machine learning adaptive scheduling framework based on a heterogeneous distributed environment to maximize the usage of distributed system resources.

3.1.2 Deep Learning implementation

Deep learning can be implemented at edge servers and their parameters can be tuned to enhance the performance of models at microservices in the edge. For example, Dhakal *et al.* [92] suggested a coded computation architecture for

TABLE 2
Edge AI Computing

Proposed Schemes	AI Method	Technical Approach	Data Dimentionality	Model merits	Containerised Microservices
[89]	ML	Docker containers	Sensory data	reduces the need to move large quantities of data across the network, improve the QoS, privacy & security	✓
[90]	ML	Resource Detection System (RDS) and Task Scheduling System (TAS)	network traffic data	maximise the usage of distributed system resources	x
[6]	DL	online scheduling algorithm	raw sensor data	optimize network performance and protect user privacy in uploading data	x
[91]	DL	transfer learning using K-means	multimedia content popularity	estimate multimedia content popularity	x
[92]	ML	Distributed Gradient Descent	resource heterogeneity statistical data	calculate a near-optimal coding redundancy for encoding training data and partitioning the coded data across worker nodes	x
[93]	ML	Distributed Gradient Descent	IoT, crowd-sourcing,& social networking	minimize loss function	x
[94]	FL	Scheduling protocol allowing the AP to collect timely updates from the User Equipment for FL training	end-user devices	enhance the running efficiency	x
[95]	FL	selective aggregation of image classification	network traffic data	Deal with information asymmetry, privacy, and selecting approperiate DNN models	x

distributed gradient descent computation depending on statistical information of communication and compute delays. The architecture uses resource heterogeneity statistical data for determining the training data's load balancing and optimal encoding utilizing random linear codes without the use of decoding gradients in mobile edge computing. Dean *et al.* in [96] proposed the model of parameter server that fastens DNN training by distributing computations throughout several nodes. The major introduced concept was paralleling data computation wherein every compute node handles asynchronously a subset of the training data. Ye *et al.* [95] examined the use of federated learning within Vehicular edge computing (VEC) through describing selective model aggregation approach of image classification as a typical VEC-AI application. Within the model selection procedure, local DNN models were picked and forwarded to the central server through evaluating the quality of local image and the capability of computation.

3.1.3 Deep Learning Deployment

The deployment of DL algorithms at the network edge requires specific considerations. Li *et al.* [6] stated that a deep learning model is quite suitable for the environment of edge computing due to its multi-layer structure. Also, they explained that it is sensible to offload chunks of the learning layers within its nodes, then move the decreased intermediate data, resulted from pre-processing procedures to the main cloud server because edge computing could execute well as intermediate data size gets less than the size of input data. Jiang *et al.* [97] proposed a two-stage pipeline for optimizing the practices of deep learning inference on edge nodes to reduce latency. Graph transformation was first utilized for inference workload optimization, and then search the implementations of the optimized kernel on the target node. Hou *et al.* [91] exploited an approach for developing a transfer learning-based technique to estimate multimedia content popularity using a K-means clustering algorithm, as well as formulate a proactive content caching optimization model in a mobile edge computing network.

3.2 Microservices Frameworks at Edge Computing

The target of this section is to find a suitable framework where microservices can fit within fog/edge computing paradigms. For this reason, a thorough search has been carried out for investigating the recent research directions related to designing microservices frameworks at edge computing.

3.2.1 Network Function Virtualization (NVF)-based Architectures

After analyzing the key challenges related to modularisation and security aspects in the IoT-edge field, Shih *et al.* [98] suggested that a resilient and agile management architecture is needed for the NFV-based edge computing paradigm to boost recent IoT applications. The challenges were: the increase in bandwidth consumption resulting from the remote function sharing, the lack of comprehensive system operations, and the necessity to handle mobility. For all of that, they propose a comprehensive microservices and NFV technology-based Edge computing framework for agile IoT application service provider which enables the sharing of remote virtual function. The framework incorporates system operations protocols ensuring the continuity of service, a function module allocation algorithm, and a testbed showing the practicability of the proposed framework.

The work presented by Chaudhry *et al.* [99] included a serverless-edge computing system that integrates the Multi-access Edge Computing (MEC) functional blocks with NFV orchestrator utilizing a cluster of Kubernetes. The VNFs

were combined as microservices, then deployed on-demand via containers on Edge devices. Zuo *et al.* [100] proposed a generic runtime anomaly detection in microservices architecture. The architecture is splited into two steps: preparation step for data representation learning and two-stage anomaly detection outliner identification composed of timely-based logs modeling for service execution and locative service query traces analysis. The design can compatibly work with several algorithms such as doc2vec, LSTM model, unsupervised outliers identification, and tracing matrix.

3.2.2 Hybrid Frameworks

Fully shift from Cloud to Edge Computing is not feasible yet, which makes a lot of designs to consider a hybrid cloud-fog-edge framework as a primary stage in achieving edge microservice platform. The issues arises from having a wide spread resources over heterogeneous system were discussed to be reaching isolation, scalability and provisioning in the article introduced by Filip *et al.* [101]. In order to tackle them, the paper proposes a microservice scheduling model over the environments of heterogeneous hybrid cloud-edge. Making use of a specific mathematical formulation, the paper could describe an architecture that is capale of handling various microservices within heterogeneous machines. The paper mentioned that when facing critical power limits in a nanodatacenter (NaDa), it is crucial to offload tasks to an additional processing unit. Thus, the discussion in the paper involved scheduling and cloud-edge nanodatacenter considerations regarding scenarios of IoT energy efficient and Microservices-oriented platform.

Calder *et al.* [102] proposed a hybrid cloud blueprint utilising microservices in the edge and cloud computing support high-performance applications in order to pre-diagnose infectious illnesses of older patients. The microservices architecture manages a continuously updated medical database, which is fed with essential signs from biosensor units applied by medical caretakers to old individuals every day. The blueprint was created by microservices design, including nine services, where each of which resides in a separate container, and they work as distributed groups of asynchronous nodes communicating through a REST communication protocol. This allows the usage of different kinds of databases based on demands that permits agile scaling of individual microservices, which are generally utilized by replication dependent on multiple containers, without expecting to duplicate underutilised services.

Similarly, Mendes *et al.* [103] stated that microservices were utilised to perform remote observation for biometric metrics, such as blood pressure and electrocardiograms, as well as local environmental data in a specially designed settings for elderly individuals. The authers pick this paradigm in consolidation with the fog computing scheme targeting a secure and adaptable framework wherein each service scales freely depends on data processing and analysis demand. A pre-processing and validation of data collected at the front-end permit a better computing distribution.

The paper presented by Alturki *et al.* [104] suggests the decomposition of services creating Linked-Microservices (LMS) which define services that run at several devices while closely linked to a linked-partners allowing computation distribution throughout various nodes in IoT systems. The authored explored and demonstrates service decomposition efficacy on cloud, fog, hybrid and fog+cloud architectures through applying experiments on various kinds of datasets.

3.2.3 Containerised Microservices

CA Containerised microservice is one of the most famous techniques that has been implemented within the cloud paradigm [105]. Recently, there were multiple attempts to implement this technology at the network edge. For instance, Zhao and Huang [106] presented a microservice container-based fog system (MSCFS) considering the issue of cost-efficient task scheduling within heterogeneous fog-cloud nodes for the applications of IoT. In the MSCFS system, microservices were provided by Fog servers, each of the servers was made of a single virtual machine (VM). While every VM can support multiple container-Microservices at the same time. Internally, a docker-engine was deployed to add or remove container Microservices for the offloaded tasks. Every container holds a single business Microservice for every task, and it used REST API to communicate with the Microservice.

Harjula *et al.* [107] stated that a virtualized local service could be deployed and performed using a well-executed distributed mechanism utilizing Docker-based nano-services which are deployed on single-board-computers (SBC) devices, such as Raspberry Pi, and that virtual functions can be installed by mobile agent technology on further resource-constrained microcontroller devices. It proposed a service model where nodes collaboratively supply the services with the required processing, management, interfaces, storage, and security functions, regardless of the need to rely on a lot of centralized servers. Shadija *et al.* in their paper [108] examined the microservices granularity issue and its effects on latency. Microservices deployment was simulated using two methods the first involves Microservices within one container and the second was to divide them into multiple containers. The latency for the second method was slightly more than the first one when deploying microservices.

Alam *et al.* [8] introduced a modular and layered platform that runs on the cloud, fog, and edge devices as well as provides services and microservices in a containerized manner. Their approach involved combining Docker and microservices running as a scalable layer on top of the edge architecture. Achieving services isolation was done via containerization, whereas the scalability of containerized service was achieved via Docker tools application. Additionally, the applications were considered as independent microservices, making the system more decentralized. The application, in the design, represents a set of stateless services that could be distributed throughout single or several devices.

3.2.4 Osmotic Computing

Some frameworks have been introduced using osmotic computing to achieve the goal of building a distributed microservices system. Villari *et al.* [110] presented the paradigm of osmotic computing as a one capable of responding to secure data exchange, resource heterogeneity, as well as effective deployment of microservices throughout federated cloud systems. Osmotic computing can be defined as an innovation paradigm which combines and extends multiple

TABLE 3
Microservices Architectures at Edge Computing

Proposed Schemes	Architecture Technology	Technical Approach	Data Dimentionality	Security Practices	Model Focus
[99]	Edge NFV-based	Kubernetes cluster	IoT devices Data	Encryption, Firewalls, Filtering, Intrusion detection	Integration which can leverage serverless computing for merging MEC and NFV at system level & deploying VNFs on demand, through combining MEC functional blocks with NFV orchestrator
[100]	5G Mobile Systems	OpenStack, LSTM and SVM	enormous and multi-source networking data	future work	learning-based anomaly detection system for service-provision systems with microservices architectures utilising service execution logs (temporally) & query traces (spatially)
[102]	hybrid cloud-fog-edge	telemonitoring system	biosensor data	asynchronous HTTPS server based on TLS for client authentication	enables detection and assisted clinical diagnosis of infectious diseases of elderly patients
[103]	hybrid cloud-fog-edge	WSN, VITABOX, and cloud PaaS	home environment and biometric sensors	secure channels (https, wss, ampqs and authentication in the MySQL and MongoDB drivers)	allowing a dynamic and intuitive management of patients and equipment in healthcare
[109]	hybrid cloud-fog-edge	blockchain-enhanced microservices platform	video data	blockchain techniques and Smart Contracts	improve smart surveillance systems
[104]	hybrid cloud-fog-edge	containers	IoT devices Data	future work	decomposing services to create Linked-Microservices (LMS)
[106]	hybrid cloud-fog-edge	Containers	network devices Data	future work	reduce communication and computation cost
[107]	Decentralised IoT platform	VMs	Heterogeneous IoT Data	localisation and orchestration	Model nodes collaboratively provide virtual services functions that need to be deployed locally
[108]	hybrid cloud-fog-edge	Containers	IoT devices Data	container validation	provide applications that can scale in response to emerging requirements
[110]	Osmosis	Containers	Heterogeneous IoT Data	localisation and Osmotic engine	present Osmotic computing paradigm considering resource heterogeneity, security, and microservices deployment across federated cloud systems

distributed computing technologies. The osmosis paradigm boosts the management of IoT micro-elements (MELs), which is an abstraction determined and utilised in the paradigm to create and deploy IoT applications throughout various resource kinds. MELs could be migrated throughout various resources within the system as well.

Similarly, another paper of Villari *et al.* [72] explained osmotic computing in detail as a basis of providing a unifying paradigm for the reverse offloading, which refers to functionality movement from cloud to edge devices. The purpose is to decrease the sizes of data which must be transferred throughout a network as well as a counter for latency-sensitive applications. Several research directions for microservice were introduced, including configuration, security, networking, orchestration and elasticity control, and workload contention and interference evaluation, and containers within the setting. In the platform, the cloud system hosted Docker managers, whereas Docker workers were deployed in the fog/edge devices.

3.3 IoT-based AI Microservices

This section aims to explain the existing methods of employing microservices to perform AI algorithms at IoT-edge nodes. We explore how microservices, containers, machine learning, and IoT/edge systems, have been correlated in one way or another, as demonstrated in Tables 3 and 4.

3.3.1 Multi-layered AI Microservices

The concept of microservices is to divide a service into small sub-services that are to be deployed in a distributed way. Sun *et al.* [42] proposed a generic microservice IoT architecture for multiple applications. The key idea of this work was for reconstructing and decoupling IoT system functions into independent and specific services. Their architecture consists of several microservices including Geo, tenant, devices, big data, artificial intelligence, application, and security services. These modules are wired and connected to provide a complete IoT system, and they are also managed and coordinated using a core microservice. In a qualitative comparison with existing approaches, the architecture proved its efficiency. However, many aspects have not been addressed by this model such as the network delay, faults in a network, and security of systems.

Bierzynski *et al.* [111] introduced a technique for monitoring, distributing, learning, and maintenance the components of a self-learning lighting system as microservices. The principal intention of this method is to help the lighting systems to learn from real-time data and adapt themselves

TABLE 4
IoT AI Microservices Comparison

Group →	Group 1					Group 2			Group 3			
Papers / Category	Sun *et al.*[42]	Bierzynski *et al.*[111]	Rychener *et al.*[112]	Liu *et al.* [113]	Debauche *et al.* [114]	Ali *et al.* [115]	Alves *et al.* [116]	Ribeiro *et al.*[43]	Servia *et al.* [117]	Pahl & Loipfinger [118]	Shahoud *et al.* [119]	Chen *et al.* [120]
Fog/Edge	✓	x	✓	✓	✓	✓	x	x	✓	✓	x	✓
Cloud	x	✓	x	x	✓	✓	✓	✓	✓	✓	✓	✓
ML classifier	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
DL	x	x	x	x	✓	✓	x	x	✓	✓	✓	✓
Security/Privacy Constraints	✓	x	x	x	x	x	x	x	✓	✓	x	x
Docker	✓	✓	✓	x	✓	✓	✓	✓	✓	x	✓	✓
Heterogeneous Data	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Orchestration	x	✓	✓	x	✓	✓	✓	✓	x	✓	x	x

for data change or drift. For example, in their approach, the monitoring service was divided into small services, each of which monitors specific system's components such as machine learning model, pre-processing component, and the analyzer component. To handle the drift problem, the authors adopted the ensemble approach from machine learning where the possibility of reusing previously recognized drift patterns to faster recognize the data with the same patterns. In fact, their proposed approach is highly complex and it is not evaluated properly in terms of reusability, complexity, and loosely coupling as there are a lot of services that depend on each other in which a failure of one may affect others.

A three layered microservices approach for detecting anomalies in an industrial application was introduced by Rychener *et al.*, in [112]. The first layer represented the industrial sensor and machinery wherein data was sensed, collected, and stored. The second layer provided several small services for data processing and event detection based on rules and machine learning. The third layer was created to show the notification and alarms for end users. For constructing an IoT environment safety monitoring system, the technique of microservice was used by Liu *et al.* [113]. The proposed approach utilized API or an HTTP-RESTFUL interface to allow the users to interact with provided services in the service layer and the infrastructure service at the business layer (i.e., cloud). Each of the system's components was represented as a set of microservices. For example, the data pre-processing was divide into multiple microservices including data integration, statute, transformation, and cleaning, and the machine learning component is represented as offline and online prediction and each of which can provide two types of learning supervised or unsupervised depended on the type of data. The message-driven middleware was used to enable communication among all the provided microservices.

3.3.2 Machine Learning-based Microservices

The development of machine learning at the edge paradigm has a great research interest. Debauche *et al.* [114] proposed a heterogeneous architecture for the distribution of machine learning models to IoT devices in the edge, through microservices. Their approach utilized three entities. A cloud server that hosts services such as an MQTT broker, knowledge base, neural network, and container orchestration. Edge IA-IoT micro-cluster, which plays the role of the bridge/gateway for the IoT infrastructure, maintains an MQTT client, a local semantic reasoner, microservices, and tiny neural networks which are the lower layers of a distributed neural network, with the remaining NN being deployed in the cloud. IoT nodes, that connect to the cloud through the Edge IA-IoT micro-cluster, collect data and enrich them semantically. Results support the validity of deploying ML and DL models on IoT/Edge devices, although more research is required to increase the speed of execution and the accuracy of these systems.

Ali *et al.* [115] proposed a system based on containers that provide machine learning-based microservices to power analytics for IoT devices. In the proposed system, initially, IoT devices collect data from their environment, which may include missing values, mismatching scales, and other imperfections that impede the training of ML and DL models. After the data has been collected at the IoT devices, microservices are deployed to perform pre-processing, and enrich the data.

Machine Learning Framework for IoT data (ML41oT) was presented in the study of [116]. The key objective of this framework is to address the challenge of orchestrating the machine-learning workflows and big data tools on high volume, velocity, and variety IoT data. The technique of microservices was implementing on data aggregation and training data production, executing the workflow as microservices, defining the utilized ML, and deployed the trained model in production. The proposed framework introduced two workflows including batch learning and online learning based on container and docker resources.

3.3.3 Deep Learning Microservices

Deep learning, as the main practice of Machine Learning and consequently AI, is considered to be one of the most effective techniques in recent times. Implementing DL requires powerful capabilities that mostly Edge devices do not have. However, there were several efforts to overcome this obstacle through utilizing the collective effort of multiple microservices to achieve a distributed Deep Learning solution. Teerapittayanon *et al.* [121] produced a Distributed Deep Neural Network (DDNN), a type of neural network with layers that span from the cloud back-end server to the edge and individual IoT devices. Portions of the proposed DNN are thus

spread in remote geographic areas, where devices are deployed on the edge. The DDNN is trained in an end-to-end way, and can be used in its entirety at the cloud, or locally in the edge devices where a low-depth subset of the network process sensor data. The benefit of this kind of DNN is that initial classification can be performed on the device itself, while for more complex tasks, the other layers at the cloud can be leveraged [122]. The evaluation of the proposed DDNN framework indicated that a DNN can be trained and achieve adequate accuracy while benefiting from the parallel processing paradigm and gaining robustness.

Shahoud *et al.* [119] presented the technique of managing machine learning and data analytics for smart grid big data based on web application and microservices. The main aim of this proposed framework was to simplify the training, testing, storing, and retrieving the desired machine learning model, and configuring the appropriate big data run-time infrastructure. Although their proposed framework achieved a good performance, it showed an overhead in the case of using a large size dataset. Dividing neural network architecture into small or micro-networks that can be deployed in the limited IoT resources was presented by Coninck *et al.* [123]. The main reason behind this design is prioritizing the decision process for IoT applications and speeds up the response process where a small neural network is used to deal with most priority output at IoT devices and then a bigger neural network for more output is used at the cloud. This proposed approach reduces the size of data that is sent to the cloud and minimize latency and bandwidth usage. However, it needs extensive pre-processing to define the training data for each neural network and it is clear how the defining process for the most priority outputs for a small neural network. Further, it is not clear how the model decides which data is sent to the cloud or not.

3.4 Microservices Security Challenges At Edge Computing

When considering security issues related to microservices at edge computing, many issues are raised. This section will mainly focus on four main security challenges related to service transmission and analysis, that is: containers, data, permission, and network.

3.4.1 Containers Challenges

When considering the first challenge, containers provide a typical domain for microservices. There are a few advantages when using containers to wrap distributed microservices, such as reducing complexity when dealing with various platforms through the removal of dependencies on infrastructure services. Thus, a microservice architecture can utilize Docker containers for testing and deploying individual services in separate Dockers throughout the available computing devices at the edge network. Furthermore, Dockers could offer standardized building as well as continuous integration and delivery. In summary, container existence is highly relevant to microservice development and it ties together forming an ecosystem. However, if the containers encounter security issues, it will result in causing a real influence on the microservices. Various sophisticated attacks, such as DoS and DDoS, and API attacks, would exploit vulnerabilities of microservices and their networks [124].

For the security purpose, the authors of [125] utilized a microservice architecture to provide a deception network, i.e., sandnet, where the full application was divided into multiple containers that communicate over network-based API. Each container represents a uni-purpose or function and all these containers are controlled using a software-defined network (SDN). In this architecture design, the intrusion detection system sends an alarm to the controller whenever it detects an intrusion or suspicious container inside the production network, thus the controller will sandbox the network connections from and the to the suspicious container. The experiment results proved a good performance in terms of quality of deception compare with traditional deception techniques (e.g., virtual machine). However, this framework can only deal with insider adversary in the internal network and cannot work with an external adversary.

When running microservices on a Docker, there should be some security considerations. An overview of how to analyze the security level of Docker and how it affects the security of a deployed application is introduced by Bui [126]. The analysis considers two fields, Docker's internal security, and its interactions with the security characteristics of the Linux kernel. A few current pieces of research present some practices to mitigate the vulnerability problem within the kernel of containers. Jian and Chen [127] propose a solution mainly based on detecting the behaviors of escape attack through the comparison between namespaces status in every container. Escape attack is penetrating through hosts by exploiting the kernel vulnerability so it could attack more containers. The presented approach is effective for escape attacks but it does not suit other threats like DoS and botnet attacks.

Gao *et al.* [128] introduced further security properties in their power-based namespace technique like implementing additional namespaces and control groups to detect data leakage for a multi-tenancy containers service. Their proposal was able to extract power usage statistics for each container as well as dynamically limit the computation power of any container which have surpassed its pre-defined power thresholds, achieving good resource isolation. However, a few resources in the system are still hard to get partitioned. Bacis *et al.* [129] discussed a technique that adds a policy module of Linux-based sound security subsystem (SELinux) for Docker to boost security. However, it could only be applied to primary kernel vulnerability like data leakage.

3.4.2 Data Challenges

The architecture of microservices has a fine granularity property that is why a big microservice-based software system normally holds a large number of services. Such services would introduce extra attacked surfaces within the architecture of microservices because they could be developed by separate technical teams utilizing varied technologies, with their data could consequently be saved in multiple databases. This is similar to security issues introduced when deploying microservices within Cloud computing platform, that data privacy of users could be affected or misapplied. Security risks not only involve data interception, other than

the inference of business operations from messages flow by competitors. Overall, there is a necessity for looking into several data security issues within the Microservices architecture, including, authenticity detection, tamper-proof data sources issues, and data transmission protection.

Zhang *et al.* in their review paper [35] analyzed and summarized the potential challenges about data privacy and security in the Edge computing paradigm as well as the possible security mechanisms along with the open research directions and issues. The summary of data privacy and security requirements were based on five metrics, incorporating availability, confidentiality, authentication, access control, integrity, and privacy requirements. Besides, describing a detailed summary about cryptography technologies solving data privacy and security issues incorporating attribute-based encryption, identity-based encryption, homomorphic encryption, proxy re-encryption, and searchable encryption. Barhamgi *et al.* in their article [130] explained that data in distributed architectures are saved in various locations and that privacy and security could be crucial challenges because of that. It identifies three main challenges to be considered to attain efficacious privacy protection in IoT as the lack of significant strategies for data degradation, models of rich pricing for privacy-sensitive data, and the modeling and monitoring of context-dependent data which triggers privacy decision adaptation. A widely utilized procedure ensuring data security and privacy within the communication of microservices is Encryption, though it includes some shortcomings.

There are a lot of existing encryption methods widely used for the protection of data. For example, Somani *et al.* [131] had an attempt for evaluating the security of data as well as the methodology of cloud storage through implementing RSA algorithm. RSA is a public key data encryption standard and it is one of the mostly utilized cryptographic algorithms, that could resist lots of the know password attacks which are considered as a very strong and asymmetric encryption method. Even though the RSA key length is quite long and hardly break, the decryption process can take a very long time and consequently reducing the performance and efficiency of the system. Agarwal *et al.* [132] introduced a data encryption service within a cloud environment as well as offer to manage data encryption requirements through a centralized framework dealing with different applications.

3.4.3 Permission Challenges

It is crucial to address permission problems in microservice architectures because one of the most substantial measures in safety is trust. Apart from that, the authenticity of all services has to be verified in the microservice architecture. If an attacker controls one service, that service could maliciously affect more services in the environment. An example of that is when a malicious service would seize the majority of resources to reduce other services' performances, or it could intervene with the network creating a critical software system's disturbance. In other respects, upon receiving a message by service, there is a need to check whether the source service is authorized and if the message is spurious. Additionally, authorization may be required for a Microservice to access the resources of users and perform a data exchange with third-party services. To protect the edge IoT devices managed by a Microservice architecture, authentication-based methods need to be investigated [133]. In a nutshell, authentication and authorization problems are to be addressed for the cases when services communicate with one another in Microservice architectures.

There exist several ways ensuring security through permission. For instance, Grid Security Infrastructure (GSI) is vastly used to address access control issues. The GSI contains communication encryption and private key protection [134]. Also, Community Authorization Service (CAS) model could support distributed administration mechanisms which are crucial to solve flexibility and scalability issues [135]. Patanjali *et al.* [136] explained a novel microservices architecture design that emphasizing the validation to develop a dynamic billing, rating, and charging for cloud service providers. The security of every Microservice is ensured via the fusion of OAuth. OAuth is a famouse authorization protocol as well, it also a well-known method for protecting Representational State Transfer (REST) APIs against unauthorized access [137]. Also, the protocol has an authentication layer on top of OAuth named as OpenID Connect extension which permits the services to read fundamental user data.

Li *et al.* [137] proposed a microservice architecture utilizing OAuth to ensure permission security. OAuth authorization technique can only read the public information of a user. When an application requires to have extra information, it needs to have advanced permission which requires a clear statement and user's consent. However, some applications employ OAuth as a default authorization method, that means it will accord OAuthor higher advanced authorizations by default which could leak out the information of users while they are quite a negligence of such breach.

3.4.4 Network Challenges

Network and secure communication issues of microservices could be only assured by a secure network. Microservices are normally implemented within Software Defined Network (SDN), which can separate the control of the network from its physical network to effectively eliminate the restriction on the hardware of the network imposed by the supply manufacturer [138]. Consequently, organizations could adjust the architecture of the network similar to the way software is installed and upgraded which will support the upgrading of the whole software architecture, the adjustment of the organization, and its expansion. SDN usually monitors the flow of the network within a microservice architecture due to its flexibility. Apart from that, microservices architecture confronts some conceivable safety dangers of SDN. Particularly when we have a more complex network and the communication within it happens more recurrently due to the large number of microservices. That is why a comprehensive security analysis is required about the network challenges in microservice architectures.

Porambager *et al.* [139] discussed their vision and security impact within edgeAI paradigm. They emphasize that edge Computing faces multiple security threats regarding services, network, and virtualization infrastructure. The security threats that mostly occur on edge would be service or resource manipulation, Denial of Service (DoS) attacks,

man-in-the-middle attacks, and privacy leakage. To protect the system in opposition to a threat actor who takes possession of the legitimate network credentials, deep learning procedures would be a necessity to analyze the behavior of a specific user over a sequence of actions. Besides that, applications need a run-time intelligence to make situational based decisions so they need to comprise security primitives based on the dynamic requirement. This would help to make smart decisions at the edge and to rapidly deliver security services to keep high-security objectives through fully utilize advances in network AI technologies.

Villari *et al.* [140] focused on security concerns embedded in devices as well as the network among them. It explains integrating a security strategy layer between Edge and Cloud resources to enable optimal microservices migration along with their data (referred to as MELs). The secure software-defined layer is configured according to a security policy realized by an attacker model involving security analysis that specifies possible attack kinds and resources which can be affected. Some methods were proposed to mitigate the network issues related to microservices. Aliyu *et al.* [141] introduced a trust framework addressing a vulnerability within SDN architecture which resides between the network applications and the controller. As trust issues could result in various kinds of attacks and significant consequences affecting the whole microservice fog operations.

The framework proposed modules that verify network applications' authenticity and assign privileges. But, whenever the authentication is requested, the framework needs to inquire the Trust database twice for permission data which would increase systems complexity. Also, when the database involves a large number of records, the permission query will result in system delay. Additionally, Sun *et al.* [57] proposed an API for the cloud infrastructure named FlowTap enabling the support of virtual network observation. It initiates observation relations between security monitors and the microservices and permits security monitors to deploy policies on network traffic. The approach would supports cloud vendors in a way that they could supply the applications with security as a service based on Microservice architecture. However, vendors also need to apply other defensive measures to address network attacks if some malicious attackers occur.

Given these security flaws, the privacy information that may leak, like data, identity, and location, can cause a lot of serious consequences. First, unauthorised entities or curious adversaries can access or steel the private and sensitive data of users. Edge sensors and servers could gather such data from end nodes, and to protect against that, practises offer a privacy-preserving data analysis, or a design of lightweight data privacy-preserving approaches can be used like data aggregation with homomorphic encryption without decryption, pseudo-random permutation or Probabilistic public key encryption [142]. Second, an internal adversary having sufficient access privileges can manipulate the data flow by providing false services and bogus information to other entities. In such dynamic and distributed computing environment like Edge computing, it is crucial to protect users' identity data throughout the management and authentication procedures [143]. Finally, as users normally have a relatively unchanging point of interests (POIs), their location

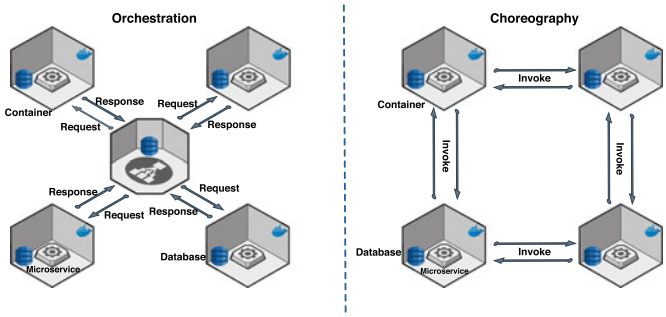

Fig. 1. Orchestration versus Choreography.

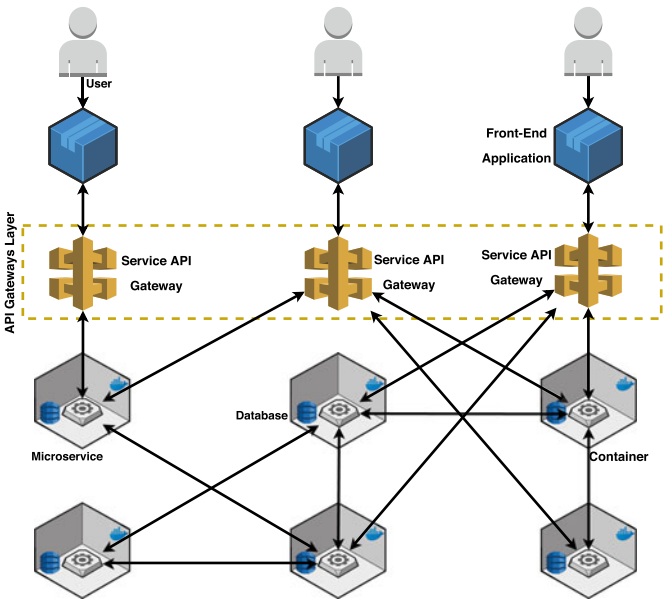

Fig. 2. Microservices framework at edge computing.

data is quite anticipated meaning that they will probably utilise the same edge servers repetitively. In case attackers could gain sufficient control privilege as a legitimate administrator of the edge data centre, that will lead to services manipulation or privileges abuse. In such case, more attention should be paid to protect users' location privacy [144].

4 Proposed Secure Edge-AI Microservices Framework

In this section, we will introduce our view of microservices-based Edge Computing Framework involving IoT nodes, Edge servers, API gateways, Edge containers containing microservices and their management as well as users applications. It was designed based on interacting network and IoT/IIoT systems with the layers of Edge/Fog and Cloud to describe the realistic implementation of recent real-world IoT/IIoT networks. The framework is shown in Figure 4, which includes data privacy and network security services, which their master entity resides on the Edge server while each Edge container could contain a security and privacy client entity. Edge/Fog computing is similar in offering on-premise services, like Cloud services, including Software-as-a-Services (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) [145]. The services are offered near to organisations to manage IoT systems and their

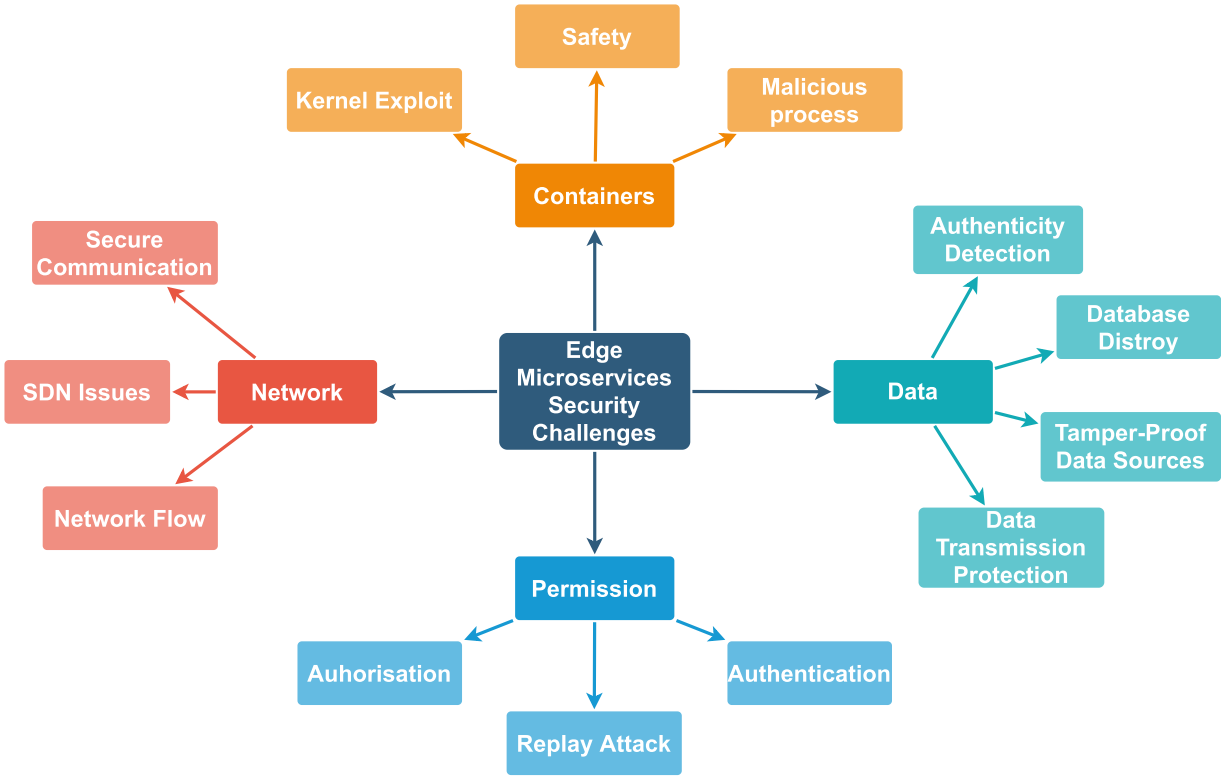

Fig. 3. Microservices security challenges in edge computing.

generated data, allowing analytics and intelligence close to end-users rather than sending a huge amount of data sources to the cloud, which has limitations related to the network bandwidth, security, and latency.

The proposed paradigm includes multiple Containers of Virtual Machines (CVMs) allocated for every AI-based Microservice or cybersecurity solution, like intrusion detection and threat intelligence, at the edge of a network. Rather than utilising the physical and computing resources manually, AI-MAAS could apply an AI orchestration process that automatically configures the computing resources and securely implement MAAS at the edge.

Each Microservice provides part of the overall AI service related to each application. For example, when DL is the aim of the application, the corresponding service API Gateway would communicate using REST API with the containers that hold Microservices related to DL operation like data aggregation and storage, Data cleaning, Data processing, data analysis, modeling, fitting, and visualization microservices.

In the data Aggregation and storage container, data will be gathered from the IoT nodes and stored in an Edge Container that provides a data storage microservice. Then, using REST API, the data would move to another container that provides Data Cleaning microservices and so on to another Edge container till the whole DL process is accomplished and the outcome would be fed back to the application interface via the API gateway which organizes the entire process.

5 Lessons Learnt and Future Directions

Microservices are continuously being deployed on multiple network nodes to perform various services. Edge Computing is a relatively new paradigm, which has been offering extended opportunities for microservices to help with effective resource deployment, management, and interoperability. However, different aspects within IoT-edge microservices deployment still require adequate addressing. This section offers technical insights into some open research challenges and future research directions.

5.1 Edge Lightweight Microservice algorithms

With the increase number in systems deploying IoT nodes, resource constraints such as low processing capabilities, data management, lower battery power, and less memory are rising a serious challenge on how these systems are performing. Therefore, there is a requirement for lightweight algorithms to operate on the IoT edge resource-limited nodes to perform classification, data filtering, and partitioning processes. Additionally, to reduce the consumption of network resources, data aggregation and filtering could be carried out before sending data to the cloud. Moreover, conventional algorithms with the compute-intensive property might not be valid in the context of IoT. The IoT resource limitation needs lightweight security solutions and VM management techniques that can operate on the infrastructure of IoT to orchestrate distributed edge services. The realization of IoT is based on autonomously operating with heterogeneous infrastructure with IoT data is used to control complicated infrastructure. Thus, there is a need for lightweight data classification algorithms to process the data. For all of the above, there is a necessity for Lightweight algorithms enabling computation offloading, seamless connection, security, and interoperability among heterogeneous paradigm.

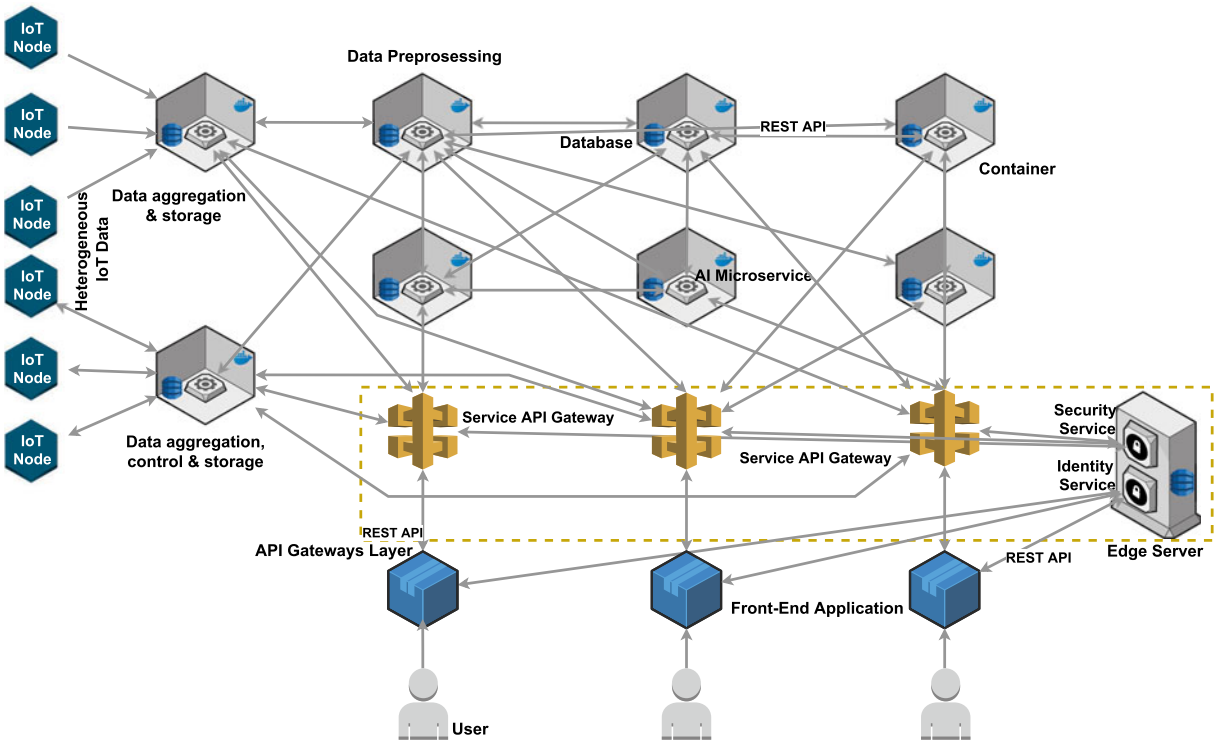

Fig. 4. Microservices framework at edge computing.

With the limited resource allocated on edge nodes, high security is still hard to achieve. IoT devices have resource limitations nature and the common cryptographic protocols may not be appropriate within all IoT networks. In accordance, researches have been proposing different lightweight crypto-secure algorithms. These algorithms require considering IoT constraints when designing and implementing the network. Multiple types of lightweight crypto-secure algorithms were introduced recently which mainly divided into symmetric and asymmetric algorithms. Some examples of such algorithms are Lightweight Block Cipher (LWBC), Lightweight Stream Ciphers (LWSC), and Elliptic curve cryptography (ECC). The factors of the lightweight crypto-secure algorithms primitives can be evaluated through the block size, key size, structures, and number of rounds [146]. In conclusion, the resource scarcity of IoT systems requires developing lightweight algorithms, OSs, and solutions dealing with such limited resources.

5.2 Resource Provision at Edge Nodes

The space of IoT applications and services would increase through leveraging edge computing advantages. However, it's hard to implement DL processes at the edge devices due to their limited specifications and the sheer amount of data each device has. Besides, the training of a deep network required time which plays a substantial role. For that, time-critical and real-time applications could not be able to benefit from DL at the IoT edge. Furthermore, DL models stability is crucial and the recently available data would have an effect on the already trained model. Thus, more investigations are required in this area to load balancing and scheduling mechanisms that optimize resources of edge nodes and their servers.

5.3 Microservices construction and deployment in IoT devices

The services provided by edge and cloud have recently begun to shift from a monolithic setup towards a vector containing many loosely-coupled microservices. The execution of AI algorithms might require a string of software dependencies, and it demands a solution to isolate various AI services within the shared resources. At this stage, the Microservice framework deployed at IoT edge to host AI services, is still quite a new field. There are multiple critical challenges that face it, amongst them is how to flexibly handle the AI deployment and management. Another challenge is how to Achieve live migration of microservices to minimise migration time and the unavailability of AI services when facing user mobility. Moreover, there is the challenge of resource orchestration amongst the distributed edge nodes and the cloud in order to obtain the best possible performance.

5.4 Security Related Challenges

The novel security requirements, expansion, resource constraints, and mobility of IoT networks cannot be coped with the traditional security practices and solutions. There is a need for intelligent methods that would analyse and automate the security reactions within IoT networks closer to the edge. The major kinds of trust and security challenges within IoT network includes: authentication and authorization, identity management, and vulnerability detection. Furthermore, the main challenging threats that affect IoT Networks are API attacks, eavesdropping, flooding, and DDoS Attacks. The best methods to prevent these types are to include intrusion detection, threat intelligence, and digital forensics based on techniques at the network edge.

5.5 Development of AI Algorithms as Microservices

AI algorithms as microservices have changed the way that software is built and deployed. Containers and the algorithm economy have improved the development for running algorithms as microservices, meaning that code is to be written by any programming language, and then smoothly united through an API. As algorithms are running as microservices, code is ensured to be dependency-free, composable, and inter-operable. This research suggests the development of an Artificial Intelligence (AI)-based Micro-Algorithms as Services (MAAS) paradigm at networks edge as a future work. It will be an innovative paradigm, to be called (AI-MAAS), which will allow efficient running and quick deployment of distributed AI micro-algorithms, especially machine and deep learning, for cybersecurity applications at the edge nodes. The AI-MAAS paradigm could be a design of an agile and secure architecture for AI models, such as data protection, privacy-preserving, and intrusion detection algorithms.

6 Conclusion

In this paper, recent developments in various techniques related to deploying AI algorithms and microservices within edge/fog computing platform were explored. We also discuss the communication security and data privacy challenges when presenting microservices at the network edge. This gives a comprehensive view of the modern practices used to perform AI processes closer to front-end devices in a secure way. As a conclusion from the survey, we propose a Containerised microservices-based framework, which provides secure algorithms as microservices to deploy AI algorithms at the IoT networks edge. This will allow the dynamic resource management at the edge, and will enable the balanced deployment of MAAS, as well as will satisfy adjustable resource contains and users' QoS. A future work would be to implement the platform on several testbeds and applying different scenarios where AI models are to be deployed and managed. Finally, accuracy, efficiency, and time metrics are to be considered and measures are to be taken to fulfill the proposed platform.

References

[1] O. Novo, N. Beijar, M. Ocak, J. Kjällman, M. Komu, and T. Kauppinen, "Capillary networks-bridging the cellular and IoT worlds," in *Proc. IEEE 2nd World Forum Internet Things*, 2015, pp. 571–578.

[2] G. Marquez, F. Osses, and H. Astudillo, "Review of architectural patterns and tactics for microservices in academic and industrial literature," *IEEE Latin Amer. Trans.*, vol. 16, no. 9, pp. 2321–2327, Sep. 2018.

[3] C. Pahl, P. Jamshidi, and O. Zimmermann, "Microservices and containers," *Softw. Eng.*, pp. 115–116, 2020.

[4] K. Nova and A. O. M. C. Safari, *Conquering Both Containers and Virtual Machines with Kubernetes*. Philadelphia, PA, USA: O'Reilly, 2019. [Online]. Available: https://books.google.com.au/books?id=hac7zQEACAAJ

[5] H. Sami and A. Mourad, "Dynamic on-demand fog formation offering on-the-fly iot service deployment," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1026–1039, Jun. 2020.

[6] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[7] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[8] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of microservices for IoT using docker and edge computing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 118–123, 2018.

[9] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 196–248, 2020.

[10] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Security and privacy challenges in mobile cloud computing: Survey and way ahead," *J. Netw. Comput. Appl.*, vol. 84, pp. 38–54, 2017.

[11] P. Kumar, R. S. Kunwar, and A. Sachan, "A survey report on: Security & challenges in Internet of Things," in *Proc Nat. Conf. ICT IoT*, 2016, pp. 35–39.

[12] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[13] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, 2013.

[14] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-Scale IoT exploitations," *IEEE Commun. Surv. Tut.*, vol. 21, no. 3, pp. 2702–2733, Jul.–Sep. 2019.

[15] M. A. A.-A. Mostafa and A. M. Khater, "Horizontal offloading mechanism for IoT application in fog computing using microservices case study: Traffic management system," in *Proc. IEEE Jordan Int. Joint Conf. Elect. Eng. Informat. Technol.*, 2019, pp. 640–647.

[16] G. Cherradi, A. E. Bouziri, A. Boulmakoul, and K. Zeitouni, "Real-time microservices based environmental sensors system for hazmat transportation networks monitoring," *Transp. Res. Procedia*, vol. 27, pp. 873–880, 2017.

[17] R. H. Weber, "Internet of things–new security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, no. 1, pp. 23–30, 2010.

[18] M. Sheraz, M. Ahmed, X. Hou, Y. Li, D. Jin, and Z. Han, "Artificial intelligence for wireless caching: Schemes, performance, and challenges," *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 631–661, Oct.–Nov. 2020.

[19] P. Krivic, P. Skocir, M. Kusek, and G. Jezic, "Microservices as agents in IoT systems," in *Proc. KES Int. Symp. Agent Multi-Agent Syst., Technol. Appl.*, 2017, pp. 22–31.

[20] S. K. Datta, M. I. Khan, L. Codeca, B. Denis, J. Härri, and C. Bonnet, "IoT and microservices based testbed for connected car services," in *Proc. IEEE 19th Int. Symp. A World Wireless Mobile Multimedia Netw.*, 2018, pp. 14–19.

[21] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, Newton, MA, USA: O'Reilly, 2015.

[22] H. Vural, M. Koyuncu, and S. Guney, "A systematic literature review on microservices," in *Proc. Int. Conf. Comput. Sci. Appl.*, 2017, pp. 203–217.

[23] T. Hidayat *et al.*, "Smart city service system engineering based on microservices architecture: Case study: Government of tangerang city," in *Proc. Int. Conf. ICT Smart Soc.*, 2017, pp. 1–7.

[24] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? a model and annotation," *J. Syst. Softw.*, vol. 83, no. 10, pp. 1735–1758, 2010.

[25] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1646–1685, Third Quarter 2020.

[26] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," 2020, *arXiv:2002.09668*.

[27] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1722–1760, Jul.–Sep. 2020.

[28] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 38–67, Apr.–Jun. 2019.

[29] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2031–2063, Jul.–Sep. 2020.

[30] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the Internet of Things (IoT) forensics: Challenges, approaches and open issues," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 1191–1221, Apr.–June. 2020.

[31] K. Yeow, A. Gani, R. W. Ahmad, J. J. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric Internet of Things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2017.

[32] N. Slamnik-Kriještorac, H. Kremo, M. Ruffini, and J. M. Marquez-Barja, "Sharing distributed and heterogeneous resources toward end-to-end 5G networks: A comprehensive survey and a taxonomy," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1592–1628, 2020.

[33] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. ur Rasool, and W. Dou, "Complementing IoT services through software defined networking and edge computing: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1761–1804, Jul.–Sep. 2020.

[34] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 869–904, Apr.–June. 2020.

[35] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[36] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[37] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1686–1721, Jul.–Sep. 2020.

[38] D. C. Nguyen *et al.*, "Enabling ai in future wireless networks: A data life cycle perspective," *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 553–595, Oct.–Nov. 2020.

[39] C. Surianarayanan, G. Ganapathy, and R. Pethuru, *Essentials of Microservices Architecture: Paradigms, Applications, and Techniques*. Boca Raton, FL, USA: CRC Press, 2019. [Online]. Available: https://books.google.com.au/books?id=XHatDwAAQBAJ

[40] B. Christudas, "Practical microservices architectural patterns: event-based java microservices with spring boot and spring cloud," 2019. [Online]. Available: https://books.google.com.au/books?id=EUefDwAAQBAJ

[41] A. Benayache, A. Bilami, S. Barkat, P. Lorenz, and H. Taleb, "MsM: A microservice middleware for smart wsn-based IoT application," *J. Netw. Comput. Appl.*, vol. 144, pp. 138–154, 2019.

[42] L. Sun, Y. Li, and R. A. Memon, "An open IoT framework based on microservices architecture," *China Commun.*, vol. 14, no. 2, pp. 154–162, 2017.

[43] J. L. Ribeiro, M. Figueredo, A. Araujo Jr, N. Cacho, and F. Lopes, "A microservice based architecture topology for machine learning deployment," in *Proc. IEEE Int. Smart Cities Conf.*, 2019, pp. 426–431.

[44] A. Power and G. Kotonya, "A microservices architecture for reactive and proactive fault tolerance in IoT systems," in *Proc. IEEE 19th Int. Symp. A World Wireless Mobile Multimedia Netw.*, 2018, pp. 588–599.

[45] A. Kwan, J. Wong, H.-A. Jacobsen, and V. Muthusamy, "Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 80–90.

[46] R. V. O'Connor, P. Elger, and P. M. Clarke, "Continuous software engineering–A microservices architecture perspective," *J. Softw., Evol. Process*, vol. 29, no. 11, 2017, Art. no. e1866.

[47] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "The evolution of distributed systems towards microservices architecture," in *Proc. IEEE 11th Int. Conf. Internet Technol. Secured Trans.*, 2016, pp. 318–325.

[48] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on SOA and microservices," in *Proc. IEEE 4th Int. Conf. Enterprise Syst.*, 2016, pp. 60–67.

[49] M. Ahmadvand and A. Ibrahim, "Requirements reconciliation for scalable and secure microservice (De) composition," in *Proc. IEEE 24th Int. Requirements Eng. Conf. Workshops*, 2016, pp. 68–73.

[50] H. Bloch *et al.*, "A microservice-based architecture approach for the automation of modular process plants," in *Proc. 22nd IEEE Int. Conf. Emerging Technol. Factory Automat.*, 2017, pp. 1–8.

[51] C. Carneiro and T. Schmelmer, "Microservices: The what and the why," in *Microservices From Day One*, Berlin, Germany: Springer, 2016, pp. 3–18.

[52] A. Furda, C. Fidge, O. Zimmermann, W. Kelly, and A. Barros, "Migrating enterprise legacy source code to microservices: On multitenancy, statefulness, and data consistency," *IEEE Softw.*, vol. 35, no. 3, pp. 63–72, May/Jun. 2017.

[53] N. Sam, "Building microservices," Newton, MA, USA: O'Reilly Media, Inc., 2015.

[54] G. A. Oparin, V. G. Bogdanova, A. A. Pashinin, and S. A. Gorsky, "Distributed solvers of applied problems based on microservices and agent networks," in *Proc. IEEE 41st Int. Conv. Informat. Commun. Technol. Electron. Microelectronics*, 2018, pp. 1415–1420.

[55] X. Zhou *et al.*, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, pp. 683–694.

[56] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *J. Syst. Softw.*, vol. 146, pp. 215–232, 2018.

[57] Y. Sun, S. Nanda, and T. Jaeger, "Security-as-a-service for microservices-based cloud applications," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Technol. Sci.*, 2015, pp. 50–57.

[58] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Softw.*, vol. 35, no. 3, pp. 24–35, May/Jun. 2018.

[59] H. Mu noz-Avila, C. Bauckhage, M. Bida, C. B. Congdon, and G. Kendall, "Learning and game AI," Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 6, pp. 33–43, 2013.

[60] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected autonomous vehicles: Challenges posed by adversarial machine learning and the way forward," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 998–1026, Apr.–June. 2020.

[61] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 21, no. 1, pp. 393–430, 2019.

[62] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, Oct.–Nov. 2019.

[63] 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. Tut.*, vol. 19, no. 4, pp. 2432–2455, Oct.–Nov. 2017.

[64] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[65] H. Wang *et al.*, "Architectural design alternatives based on cloud/edge/fog computing for connected vehicles," *IEEE Commun. Surv. Tut.*, vol. 22, no. 4, pp. 2349–2377, Oct.–Nov. 2020.

[66] C. Cicconetti, M. Conti, A. Passarella, and D. Sabella, "Toward distributed computing environments with serverless solutions in edge systems," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 40–46, Mar. 2020.

[67] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3039–3071, Oct.–Nov. 2019.

[68] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015, *arXiv:1511.03575*.

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

[70] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," *IEEE Commun. Surv. Tut.*, vol. 22, no. 4, pp. 2167–2191, Oct.–Nov. 2020.

[71] A. Taherkordi and F. Eliassen, "Data-centric IoT services provisioning in fog-cloud computing systems," in *Proc. IEEE/ACM 2nd Int. Conf. Internet, Things Des. Implementation*, 2017, pp. 317–318.

[72] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76–83, Nov./Dec. 2016.

[73] G. M. Lee, T.-W. Um, and J. K. Choi, "Ai as a microservice (AIMS) over 5G networks," in *Proc. IEEE ITU Kaleidoscope: Mach. Learn. 5G Future*, 2018, pp. 1–7.

[74] O. Voutyras, S. V. Gogouvitis, A. Marinakis, and T. Varvarigou, "Achieving autonomicity in IoT systems via situational-aware, cognitive and social things," in *Proc. 18th Panhellenic Conf. Inform.*, 2014, pp. 1–2.

[75] K. Portelli and C. Anagnostopoulos, "Leveraging edge computing through collaborative machine learning," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud Workshops*, 2017, pp. 164–169.

[76] D. H. Ho, R. Marri, S. Rella, and Y. Lee, "Deeplite: Real-time deep learning framework for neighborhood analysis," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 5673–5678.

[77] A. Mouat, *Using Docker: Developing and Deploying Software with Containers*. Newton, MA, USA: O'Reilly Media, 2015. [Online]. Available: https://books.google.com.au/books?id=wpYpCwAAQBAJ

[78] S. Y. Nikouei, R. Xu, Y. Chen, A. Aved, and E. Blasch, "Decentralized smart surveillance through microservices platform," in *Sensors and Systems for Space Applications XII*, vol. 11017. Bellingham, WA, USA: SSPIE, 2019, Art. no. 110170K.

[79] A. Mouat, *Docker Security*. Newton, MA, USA: O'Reilly Media, Incorporated, 2016.

[80] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers.," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2015, pp. 171–172.

[81] B. I. Ismail *et al.*, "Evaluation of docker as edge computing platform," in *Proc. IEEE Conf. Open Syst.*, 2015, pp. 130–135.

[82] K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis, "Microservice-based IoT for smart buildings," in *Proc. 31st Int. Conf. Adv. Informat. Netw. Appl. Workshops*, 2017, pp. 302–308.

[83] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 616–644, Oct.–Nov. 2020.

[84] M. Abomhara *et al.*, "Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks," *J. Cyber Secur. Mobility*, vol. 4, no. 1, pp. 65–88, 2015.

[85] D. Serpanos, "The cyber-physical systems revolution," *Computer*, vol. 51, no. 3, pp. 70–73, 2018.

[86] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surv. Tut.*, vol. 21, no. 1, pp. 686–728, Oct.–Nov. 2019.

[87] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 1251–1275, 2020.

[88] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surv. Tut.*, vol. 20, no. 1, pp. 601–628, Oct.–Nov. 2018.

[89] P. Kochovski, M. Bajec, R. Sakellariou, and V. Stankovski, "A smart and safe construction application design for fog computing," in *Proc. IEEE World Congr. Serv.*, 2019, pp. 378–379.

[90] X. Zhou, J. Zhang, J. Wan, L. Zhou, Z. Wei, and J. Zhang, "Scheduling-efficient framework for neural network on heterogeneous distributed systems and mobile edge computing systems," *IEEE Access*, vol. 7, pp. 171 853–171 863, 2019.

[91] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *Int. J. Commun. Syst.*, vol. 31, no. 11, 2018, Art. no. e3706.

[92] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded computing for distributed machine learning in wireless edge network," in *Proc. IEEE 90th Veh. Technol. Conf.*, 2019, pp. 1–6.

[93] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 63–71.

[94] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 8743–8747.

[95] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[96] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2012, pp. 1223–1231.

[97] Z. Jiang, T. Chen, and M. Li, "Efficient deep learning inference on edge devices," in *Proc. 12th ACM Int. Conf. Future Energy Syst.*, 2018, pp. 302–308.

[98] Y.-Y. Shih, H.-P. Lin, A.-C. Pang, C.-C. Chuang, and C.-T. Chou, "An NFV-based service framework for IoT applications in edge computing environments," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1419–1434, Dec. 2019.

[99] S. Chaudhry, A. Palade, A. Kazmi, and S. Clarke, "Improved QoS at the edge using serverless computing to deploy virtual network functions," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10673–10683, Oct. 2020.

[100] Y. Zuo, Y. Wu, G. Min, C. Huang, and K. Pei, "An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 548–561, Jun. 2020.

[101] I.-D. Filip, F. Pop, C. Serbanescu, and C. Choi, "Microservices scheduling model over heterogeneous cloud-edge environments as support for IoT applications," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2672–2681, Aug. 2018.

[102] H. Calderón-Gómez *et al.*, "Telemonitoring system for infectious disease prediction in elderly people based on a novel microservice architecture," *IEEE Access*, vol. 8, pp. 118 340–118 354, 2020.

[103] D. Mendes *et al.*, "VITASENIOR-MT: A distributed and scalable cloud-based telehealth solution," in *Proc. IEEE 5th World Forum Internet Things*, 2019, pp. 767–772.

[104] B. Alturki, S. Reiff-Marganiec, C. Perera, and S. De, "Exploring the effectiveness of service decomposition in fog computing architecture for the Internet of Things," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2019.2907405.

[105] D. Kallergis, Z. Garofalaki, G. Katsikogiannis, and C. Douligeris, "CAPODAZ: A containerised authorisation and policy-driven architecture using microservices," *Ad Hoc Netw.*, vol. 104, 2020, Art. no. 102153.

[106] X. Zhao and C. Huang, "Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network," *IEEE Access*, vol. 8, pp. 56 680–56 694, 2020.

[107] E. Harjula *et al.*, "Decentralized IoT edge nanoservice architecture for future gadget-free computing," *IEEE Access*, vol. 7, pp. 119 856–119 872, 2019.

[108] D. Shadija, M. Rezai, and R. Hill, "Microservices: Granularity versus performance," in *Proc. Companion 10th Int. Conf. Utility Cloud Comput.*, 2017, pp. 215–220.

[109] D. Nagothu, R. Xu, S. Y. Nikouei, and Y. Chen, "A microservice-enabled architecture for smart surveillance using blockchain technology," in *Proc. IEEE Int. Smart Cities Conf.*, 2018, pp. 1–4.

[110] M. Villari, M. Fazio, S. Dustdar, O. Rana, D. N. Jha, and R. Ranjan, "Osmosis: The osmotic computing platform for microelements in the cloud, edge, and Internet of Things," *Computer*, vol. 52, no. 8, pp. 14–26, Aug. 2019.

[111] K. Bierzynski, P. Lutskov, and U. Assmann, "Supporting the self-learning of systems at the network edge with microservices," in *Proc. Smart Syst. Integration 13th Int. Conf. Exhib. Integration Issues Miniaturized Syst.*, 2019, pp. 1–8.

[112] L. Rychener, F. Montet, and J. Hennebert, "Architecture proposal for machine learning based industrial process monitoring," *Procedia Comput. Sci.*, vol. 170, pp. 648–655, 2020.

[113] Y. Liu, J. Huang, and N. Lu, "Research on environmental monitoring system based on microservices and data mining," in *Proc. E3S Web Conf.*, 2020, Art. no. 02031.

[114] O. Debauche, *et al.*, "A new edge architecture for AI-IoT services deployment," *Procedia Comput. Sci.*, vol. 175, pp. 10–19, 2020.

[115] S. Ali, M. A. Jarwar, and I. Chong, "Design methodology of microservices to support predictive analytics for IoT applications," *Sensors*, vol. 18, no. 12, 2018, Art. no. 4226.

[116] J. M. Alves, L. M. Honorio, and M. A. Capretz, "ML4IoT: A framework to orchestrate machine learning workflows on Internet of Things data," *IEEE Access*, vol. 7, pp. 152 953–152 967, 2019.

[117] S. Servia-Rodriguez, L. Wang, J. R. Zhao, R. Mortier, and H. Haddadi, "Personal model training under privacy constraints," *Training*, vol. 40, no. 33, pp. 24–38, 2017.

[118] M.-O. Pahl and M. Loipfinger, "Machine learning as a reusable microservice," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2018, pp. 1–7.

[119] S. Shahoud, S. Gunnarsdottir, H. Khalloof, C. Duepmeier, and V. Hagenmeyer, "Facilitating and managing machine learning and data analysis tasks in big data environments using web and microservice technologies," in *Proc. 11th Int. Conf. Manage. Digit. EcoSyst.*, 2019, pp. 80–87.

[120] L. Chen *et al.*, "IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12610–12622, Aug. 2021.
[121] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 328–339.
[122] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 2, pp. 1988–2014, Apr.–June. 2019.
[123] E. De Coninck *et al.*, "Distributed neural networks for Internet of Things: The big-little approach," in *Proc. Int. Internet Things Summit*, 2015, pp. 484–492.
[124] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of microservices-enabled fog applications," *Concurrency Comput. Pract. Experience*, vol. 31, no. 22, 2019, Art. no. e4436.
[125] A. Osman, P. Bruckner, H. Salah, F. H. Fitzek, T. Strufe, and M. Fischer, "Sandnet: Towards high quality of deception in container-based microservice architectures," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.
[126] T. Bui, "Analysis of docker security," 2015, *arXiv:1501.02967*.
[127] Z. Jian and L. Chen, "A defense method against docker escape attack," in *Proc. Int. Conf. Cryptography Secur. Privacy*, 2017, pp. 142–146.
[128] X. Gao, Z. Gu, M. Kayaalp, D. Pendarakis, and H. Wang, "ContainerLeaks: Emerging security threats of information leakages in container clouds," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2017, pp. 237–248.
[129] E. Bacis, S. Mutti, S. Capelli, and S. Paraboschi, "DockerPolicyModules: Mandatory access control for docker containers," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2015, pp. 749–750.
[130] M. Barhamgi, C. Perera, C. Ghedira, and D. Benslimane, "User-centric privacy engineering for the Internet of Things," *IEEE Cloud Comput.*, vol. 5, no. 5, pp. 47–57, Sep./Oct. 2018.
[131] U. Somani, K. Lakhani, and M. Mundra, "Implementing digital signature with RSA encryption algorithm to enhance the data security of cloud in cloud computing," in *Proc. 1st Int. Conf. Parallel Distrib. Grid Comput.*, 2010, pp. 211–216.
[132] A. Agarwal, S. K. Tirumalai, and K. Sriramadhesikan, "Data encryption service," US Patent 10,395,042, Aug. 27 2019.
[133] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, "Design of secure key management and user authentication scheme for fog computing services," *Future Gener. Comput. Syst.*, vol. 91, pp. 475–492, 2019.
[134] M. P. Oo and T. T. Naing, "Access control system for grid security infrastructure," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.-Workshops*, 2007, pp. 299–302.
[135] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A community authorization service for group collaboration," in *Proc. 3rd Int. Workshop Policies Distrib. Syst. Netw.*, 2002, pp. 50–59.
[136] S. Patanjali, B. Truninger, P. Harsh, and T. M. Bohnert, "Cyclops: A micro service based approach for dynamic rating, charging & billing for cloud," in *Proc. 13th Int. Conf. Telecommun.*, 2015, pp. 1–8.
[137] W. Li and C. J. Mitchell, "Analysing the security of Google's implementation of openid connect," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2016, pp. 357–376.
[138] Q. P. Van, H. Tran-Quang, D. Verchere, P. Layec, H.-T. Thieu, and D. Zeghlache, "Demonstration of container-based microservices SDN control platform for open optical networks," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, 2019, pp. 1–3.
[139] P. Porambage *et al.*, "Sec-EdgeAI: AI for edge security versus security for edge AI," in *Proc. 1st 6G Wireless Summit*, 2019, pp. 1–2.
[140] M. Villari, M. Fazio, S. Dustdar, O. Rana, L. Chen, and R. Ranjan, "Software defined membrane: Policy-driven edge and Internet of Things security," *IEEE Cloud Comput.*, vol. 4, no. 4, pp. 92–99, Jul./Aug. 2017.
[141] A. L. Aliyu, P. Bull, and A. Abdallah, "A trust management framework for network applications within an SDN environment," in *Proc. 31st Int. Conf. Adv. Informat. Netw. Appl. Workshops*, 2017, pp. 93–98.
[142] S. K. Pasupuleti, S. Ramalingam, and R. Buyya, "An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing," *J. Netw. Comput. Appl.*, vol. 64, pp. 12–22, 2016.
[143] I. Khalil, A. Khreishah, and M. Azeem, "Consolidated identity management system for secure mobile cloud computing," *Comput. Netw.*, vol. 65, pp. 99–110, 2014.
[144] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Enhancing privacy through caching in location-based services," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1017–1025.
[145] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network ton_IoT datasets," *Sustain. Cities Soc.*, vol. 72, 2021, Art. no. 102994.
[146] M. Rana, Q. Mamun, and R. Islam, "Current lightweight cryptography protocols in smart city IoT networks: A survey," 2020, *arXiv:2010.00852*.

Firas Al-Doghman (Member, IEEE) is with University of New South Wales (UNSW)'s UNSW Canberra, Australia.

Nour Moustafa (Senior Member, IEEE) is with University of New South Wales (UNSW)'s UNSW Canberra, Australia.

Ibrahim Khalil (Senior Member, IEEE) is with RMIT University, Melbourne, Australia.

Nasrin Sohrabi (Student Member, IEEE) is with RMIT University, Melbourne, Australia.

Zahir Tari (Senior Member, IEEE) is with RMIT University, Melbourne, Australia.

Albert Y. Zomaya (Fellow, IEEE) is with the University of Sydney, Sydney, Australia.