Received 22 June 2022, accepted 1 August 2022, date of publication 8 August 2022, date of current version 16 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197585

SURVEY

QoE-Driven IoT Architecture: A Comprehensive Review on System and Resource Management

BOONYARITH SAOVAPAKHIRAN[1], WIBHADA NARUEPHIPHAT[1], CHALERMPOL CHARNSRIPINYO[1], SEBNEM BAYDERE[2], (Member, IEEE), AND SUAT ÖZDEMIR[3], (Member, IEEE)

[1]National Electronics and Computer Technology Center, NSTDA, Pathumthani 12120, Thailand
[2]Department of Computer Engineering, Yeditepe University, 34755 Istanbul, Turkey
[3]Department of Computer Engineering, Hacettepe University, 06800 Ankara, Turkey

Corresponding author: Boonyarith Saovapakhiran (boonyarith.sao@nectec.or.th)

This work was supported in part by the National Science and Technology Development Agency (NSTDA), Ministry of Higher Education, Science, Research and Innovation, Thailand, under Project P1952667; and in part by The Scientific and Technological Research Council of Turkey (TUBITAK), Turkey, under Project 119N049.

ABSTRACT Internet of Things (IoT) services have grown substantially in recent years. Consequently, IoT service providers (SPs) are emerging in the market and competing to offer their services. Many IoT applications utilize these services in an integrated manner with different Quality-of-Service (QoS) requirements. Thus, the provisioning of end-to-end QoS is getting more indispensable for IoT platforms. However, provisioning the system by using only QoS metrics without considering user experiences is not sufficient. Recently, Quality of Experience (QoE) model has become a promising approach to quantify actual user experiences of services. A holistic design approach that considers constraints of various QoS/QoE metrics together is needed to satisfy requirements of these applications and services. Besides, IoT services may operate in environments with limited resources. Therefore, effective management of services and system resources is essential for QoS/QoE support. This paper provides a comprehensive survey for the state-of-the-art studies on IoT services with QoS/QoE perspective. Our contributions are threefold: 1) QoE-driven architecture is demonstrated by classifying vital components according to QoE-related functions in prior studies; 2) QoE metrics and QoE optimization objectives are classified by corresponding system and resource control problems in the architecture; and 3) QoE-aware resource management e.g., QoE-aware offloading, placement and data caching policies with recent Machine Learning approaches are extensively reviewed.

INDEX TERMS Internet of Things, quality of service, quality of experience, IoT services, IoT applications, QoS for IoT services, QoS metrics, QoE metrics, IoT architecture.

I. INTRODUCTION

Quality of Services (QoS) plays a vital role in maintaining services with acceptable criteria for IoT systems. Most prior studies attempted to exploit QoS metrics for various purposes of controls to improve user experiences e.g., optimizing network delay, network quality, and energy consumption. However, recent studies have shown that designing control policies by using QoS metrics alone is not sufficient to reflect actual user experiences. QoS metrics can be seen as system factors that have impact on user experiences. In addition, user experiences can be affected by human factors [1]: user profiles, tendency, history, characteristics, gender and age. Context factors such as location, places and application-related factors (e.g., live streaming) are other non-system factors. Therefore, optimizing system based on QoS metrics only is not sufficient. Many recent works focused on studies of Quality of Experience (QoE) instead. Although there is no exact definition for QoE, the definition given by Le Callet *et al.* in [2] and [3] are widely accepted by the research community: "QoE is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.

of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state.''

Recently, emerging IoT architectures with multi-layers, e.g., Mobile Edge Computing (MEC), Fog Computing and Cloud Computing, have been proposed to improve user experiences. By placing services in computational resources nearby users, significant amount of service delays can be reduced. Managing services and resources on these emerging networks have been investigated by a number of studies. Still, QoE is not often incorporated into the architectures and resource control policies. Therefore, further investigations on QoE-driven IoT architectures and resource control policies in these emerging architectures are needed.

Developing QoE-aware services for IoT applications requires various perspectives of system design such as QoE-aware system architecture, models of QoE metrics, QoE-related underlying technologies, QoE-aware resource management and control. These topics are still actively studied in the literature. For this reason, this paper explores the QoE-driven designs and the state of the are studies on the aforementioned dimensions in a comprehensive manner.

The paper is organized as follows: First, our motivation and the categorization of the data sources are explained in Section II. Next, current designs of IoT architectures with QoS support are given in Section III. QoE-driven IoT architectures and QoS/QoE metrics used in IoT applications are discussed in Section IV and Section V respectively. Various aspects of resource management issues in IoT systems are covered in Section VI. In Section VII QoS/QoE correlation models are provided. Section VIII contains a discussion of the findings and future research directions. Finally, concluding remarks are given in Section IX.

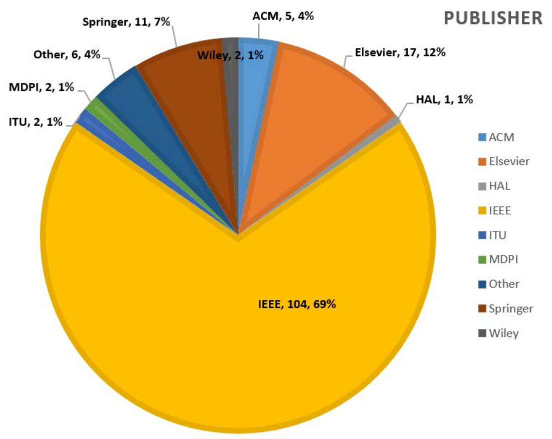

FIGURE 1. Publication sources classified by publishers.

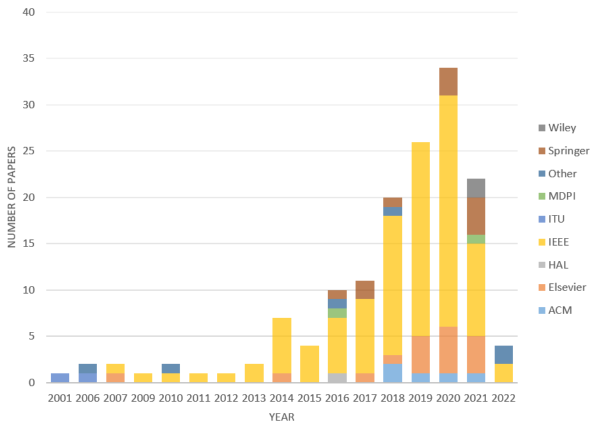

FIGURE 2. Publication sources classified by year.

II. MOTIVATION

There is an increasing interest in incorporating user experience into the system design. As QoE can be varied for each application as well as each user due to different perceptions affected by various factors e.g., user preferential, user location and types of applications, it is essential to define measurable QoE metrics that are suitable for a particular application/user. Although QoE-related topics have been studied in numerous works, these works are specific to certain application areas, solely focusing on tiers, systems, problems and methods related to those applications. Lack of understanding related to the underlying critical components in IoT systems for supporting QoE in the current literature inspires us to provide this comprehensive this study. In this paper, we provide a general guideline and mapping of how QoE can be derived and controlled for a multi-tier system for various applications using recent methods and solutions.

When QoE is considered in the design of IoT architecture, QoE-related vital components are required to be investigated first. Some prior works focused on mapping QoE components into high-level system architectures. For instance, the authors in [4] provided a mapping of QoE monitoring and management onto Software Defined Network (SDN) or Network Function Virtualization (NFV) based architectures. Although this work covers some QoE components, vital elements of QoE-Driven architectures have not yet been extensively studied. In this paper, we present a comprehensive study of a QoE-Driven structure for IoT systems by categorizing the structure into 4 major parts in regard to system architecture: QoE Cause Factors, QoE Measurement and Indicators, QoE Prediction, QoE Optimization and Control. These crucial parts are surveyed and classified from recent works.

We address the QoE-related issues and challenges in various aspects. Our data sources are drawn from well-known publishers such as IEEE, Elsevier, ACM, Springer, Wiley and others. The distribution of the publication sources classified by publishers is shown in Figure 1. As can be seen, IEEE provides the main source with 69%. The distibution of the remaining publications by publishers are 12% Elsevier, 7% Springer and 12% others. In addition, the classification of the publications by year is presented in Figure 2. 70% of the cited works are published during the period of 2017-2022.

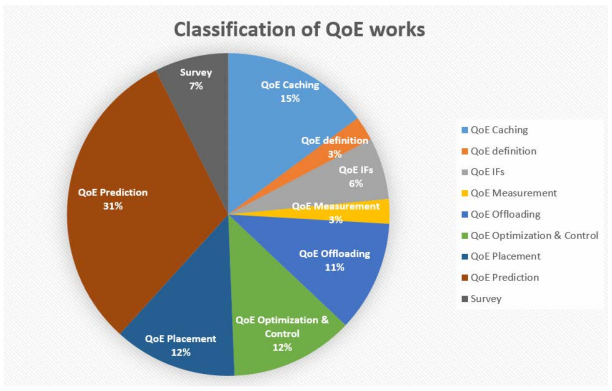

FIGURE 3. **Classification of QoE works.**

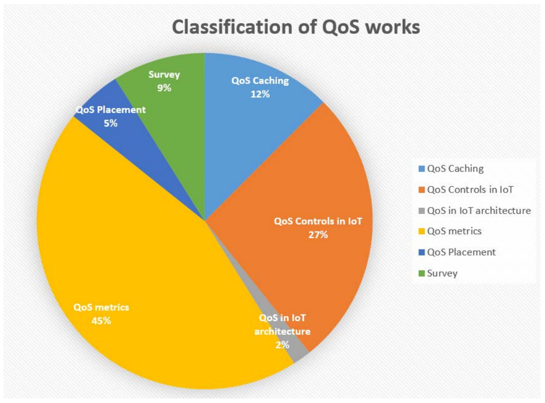

FIGURE 4. **Classification of QoS works.**

The number of publications by type are 82 journals, 61 conferences, 5 books, and 2 for preprints.

The classification of QoE related publications is also given in Figure 3. These publications are categorized into the following topics: Survey, QoE definition, QoE Influencing Factors (IFs), QoE Prediction, QoE Optimization and Control, QoE-aware offloading, QoE-aware placement, and QoE-aware caching. The number of papers per topic are 6, 2, 2, 5, 25, 10, 9, 10, and 12 respectively. Similarly, the classification of QoS related studies are given in Figure 4. These studies are also categorized into the topics: Survey, QoS metrics, QoS in IoT architecture, QoS controls in IoT, QoS aware placement, and QoS aware caching. The number of papers for each topic are 5, 25, 1, 15, 3, and 7 respectively.

In QoE-Driven IoT architectures, we consider the following essential components: First, QoE Cause Factors are quality metrics corresponding to user, application, service, network or physical layer. These factors are classified according to associated metrics of each layer for QoE-driven IoT architecture. Next, methods for measuring QoE, which can be subjective, objective or hybrid, have to be implemented for accurate QoE evaluation. Subjective methods are not available in real-time because of complexity of procedures and involvement of users. To estimate QoE in real-time, an objective function of QoE models from correlation with QoS metrics can be used. Also, QoE metrics can be modeled by using different combination of QoE Cause Factors. They are also dependent on application types, so we attempt to provide an insight of QoE metrics by categorizing them based on problem types. Nevertheless, recent works have attempted to apply emerging methods in Machine Learning (ML) for predicting QoE. Thus, QoE prediction by ML-based methods is indispensable to be a crucial part for optimizing QoE in IoT systems. When QoE can be approximated in real-time, it can be used as optimization objectives to solve variety of problems e.g., QoE-aware adaptive rate control in streaming services [5], QoE-aware power controls on MEC/Fog/Cloud [6], QoE-aware network controls on SDN/NFV enabled networks [7]. For this reason, prior works on QoE-aware optimization and control are required to further investigation from many aspects of problems.

In addition, when QoE is incorporated into the architecture, resource management problems need to be tackled. Recent issues in resource management for emerging networks can be classified into offloading problems, application/service placement problems, data caching problems. However, to the best of our knowledge, only a few recent works focused on QoE-aware offloading, placement and data caching policies. So, it is an active area that requires further investigation given that QoE is accurately predicted.

For these reasons, we are motivated to provide a comprehensive review on vital QoE-related components for QoE-Driven architecture. We classify these components in a perspective of high level architecture, and show details for each QoE-related component. These components are named as QoE Cause Factors, QoE Measurement and Indicators, QoE Prediction, and QoE Optimization and Control. Moreover, we focus on how the most recent ML techniques can be applied in each component, which is not often mentioned by prior works. A variety of models of QoE metrics and its optimization objectives are classified by problem types. Then, QoE-aware resource management is elaborated on three main types of problems: offloading, placement and caching problems. Similarly, ML-based algorithms are mainly focused in the QoE-aware resource management too.

Prior works related to ours can be listed as follows: [1], [4], [8], [9], [10], [11], [12], [13], [14], [15], and [16]. In Table 1, a summary of contributions and limitations of these works are provided and our contributions are included.

Recent surveys focused on topics related to QoE model, QoE measurement, QoE architecture based on emerging technologies e.g., MEC/Fog/Cloud, SDN/NFV and networks. Prior QoE-related survey works are [1], [4], [8], [9]. In [4], Skorin-Kapov *et al.* classified QoE models into 4 emerging models: active learning, crowdsourcing, interdisciplinary and data-aware QoE modeling. Also, important QoE monitoring components and infrastructure were investigated, and data-driven techniques for QoE monitoring was illustrated too. High-level architectures by mapping of QoE monitoring component and QoE management onto SDN/NFV enabled networks were presented. Next, a survey of existing QoE definitions, QoE measurement and the role of data in QoE modeling are provided by Fizza *et al.* [8] for IoT applications. Methods for current QoE measurement were investigated for IoT applications: subjective methods and data quality sub-metric methods. The authors categorized the role of data for QoE modeling into accuracy, data completeness, usefulness, and timeliness. According to [1], QoE subjective, objective and hybrid measurement methods are classified and discussed extensively. The authors provided a review of QoE Influencing Factors (IFs), which were divided into three main categories: System IFs, Context IFs and Human IFs. Various forms of subjective assessment such as single sequence test, double sequence test, multiple sequence test, long sequence test were investigated. On the other hand, objective assessment was divided into media-layer, parametric packet-layer, bitstream-layer, parametric planning and hybrid models. A general objective QoE model approach was shown by comparing existing objective quality metrics. Then, hybrid assessment was illustrated and shown to be related to ML prediction model for QoE. Comparison of three assessments in terms of performance was discussed. QoE management and adaptive streaming approaches for SDN/NFV-based networks e.g., Mobile Edge Computing, Fog Computing, Cloud Computing were investigated in [9]. Also, Barakabitze *et al.* discussed extension of works to emerging application: Augmented Reality, Virtual Reality and others. Various areas in optimization control in QoE-aware adaptive streaming over SDN-based networks were investigated e.g., Server and Network-assisted Optimization, QoE-Fairness and QoE-centric Control, QoE-Centric Routing Mechanisms, QoE-Aware SDN/NFV-based Mechanisms using MPTCP and Segment Routing, QoE cross-layer Optimization, QoE optimization in MEC,Fog,Cloud, and QoE for new applications (AR/VR and Gaming).

Nevertheless, prior works [1], [4], [8] did not discuss in details about QoE metrics, QoE optimization objectives, QoE prediction and control methods and QoE-aware resource management techniques. Although Barakabitze *et al.* [9] provided a comprehensive survey on various optimization area, they focused solely on adaptive streaming services over SDN/NFV enabled networks and emerging MEC/Fog/Cloud networks. Other types of QoE-aware optimizations and resource management e.g., QoE-aware offloading, placement and caching were not investigated in that work.

From a resource perspective, QoE-aware resource controls are still not extensively investigated in emerging network e.g., MEC/Fog/Cloud. Prior surveys in resource management can be found in [10], [11], [12], [13], [14], [15], [16]. Many studies of resource management, related to emerging networks, are classified into three main types: computational offloading problems, application or service placement problems and data caching problems. Surveys in offloading problems were discussed in [10], [11], [12], [13]. Reviews of placement policies are also shown in [14], [15]. Various perspectives of data caching problems are provided by [16]. All surveys are summarized in Table 1.

First, let us consider prior surveys on computational offloading problems, which were studied in [10], [11], [12], [13]. Lin *et al.* [10] classified offloading problems by flow types in Edge. Also, offloading optimization techniques were investigated. However, most optimization objectives were based on QoS metrics rather than QoE. Next, Islam *et al.* in [11] summarized offloading problems based on computation models, decision-making model and algorithmic approaches. Still, most optimization objectives were minimizing energy consumption, computational cost, revenues and task failures, which were system-related metrics. QoE was not focused in that work. Additionally, emerging networks such as Edge induce mobility problems toward offloading problems. Thus, mobility-aware offloading problems were studied in prior works, and a review on this problem is shown by Zaman *et al.* in [12]. Most works focused on energy efficiency and latency reduction. Similarly, QoE is barely seen in the review. According to [10], [11], [12], offloading problems are often solved by classic optimization. A further investigation on ML-based offloading policies is provided in [13]. However, optimization objectives are based on QoS metrics, and QoE is required more studies.

Next, a prior survey about placement policies is illustrated. A review of service placements in Fog/Edge Computing is provided by Salaht *et al.* [14]. Their placement objectives were based on QoS e.g., latency, cost, energy. A few number of studies took QoE into their consideration. Most placement methods in that survey were solved by classic optimization. Recently, ML-based approaches have been utilized to solve placement problems. A study by Salaht *et al.* [14] provided a review of ML-based placement policies in current literature. Still, QoS metrics are mainly used by prior studies, but QoE is utilized by only a few works.

For data caching problems, Shuja *et al.* [16] provided a recent survey on ML approaches for data caching in emerging networks. In [16], both classic Machine Learning and emerging Deep Learning techniques were investigated in data caching at Edge networks. Most of optimization objectives are still related to QoS rather than QoE.

In terms of resources, it is obvious that QoE-aware resource management still requires further investigation. Since some previous works focus only on QoS metrics, incorporating QoE metrics into resource management problems is challenging. As a result, we provide a reviews on QoE-aware resource management from the perspective of offloading, placement and data caching policies for emerging networks.

Moreover, current systems are being driven by QoE requirements. However, there is still lack of understanding in QoE-Driven architectures. Thus, the structure of QoE-driven architecture is summarized in this paper. We demonstrate what crucial components are being used for supporting QoE. We provide a review of QoE models that are classified by QoE Cause Factors at different layers. Methodologies for QoE prediction and calculation are discussed in details. Also, QoE-aware optimization and control are demonstrated in the level of system architecture.

TABLE 1. Comparison of QoS/QoE surveys.

Paper	Contributions	Limitation	Year
[1]	QoE subjective, objective and hybrid measurement methods are classified and discussed extensively.	Lack of QoE optimization objective in point of views of architectures and resources. For QoE prediction model, recent ML approaches (e.g., Deep learning) are still not discussed much.	2018
[4]	classifcation of QoE models into 4 emerging models, providing insight of QoE monitoring components, mapping QoE monitoring and management components on high-level SDN/NFV based architecture	Lack of discussion in details for QoE metrics, QoE optimization objectives, and QoE resource management techniques.	2018
[8]	review of QoE definition in existing IoT works, current methods of QoE measurement in IoT, some QoE models and role of data in QoE models	Lack of discussion more on QoE metrics, QoE estimation methods, QoE optimization, and Resource management	2021
[9]	QoE management and adaptive streaming approaches for SDN/NFV-based networks e.g., Mobile Edge Computing, Fog Computing, Cloud Computing. Extension of works to emerging application : Augmented Reality, Virtual Reality and others.	Focus solely on problems on QoE-aware adaptive streaming over SDN-based networks and emerging networks. However, no studies on other types of QoE optimizations resource-related problems e.g., offloading, placement and caching.	2019
[10]	Review of classification of offloading problem, offloading methodologies from various optimization methods	Most offloading problems are considered in QoS view, but not in QoE perspective.	2020
[11]	Focus on offloading problems in MEC, classify problems in perspective of computational model, decision making model and algorithmic methods	No QoE consideration. Most prior works are concerned merely with QoS metrics.	2021
[12]	Reviews of offloading optimization problems when mobility is considered at MEC	Focus on offloading problems with mobility awareness. Most objectives are QoS, but not QoE	2021
[13]	Survey on ML-based optimizing offloading problems in MEC	QoS metrics are mainly considered in the survey without addressing QoE.	2020
[15]	Review on AI-related methods for optimizing placement policies in Fog Computings	Most objectives are related to QoS, but only a few studies on QoE-aware placement policies	2021
[14]	Focus on optimization of placement policies in Fog and Edge Computing.	Most methods are based on classic optimization with QoS objectives, but without QoE objectives nor ML-based methods.	2020
[16]	Review of ML-related caching policies for Edge networks.	Most objectives are QoS (Cache Hit Rate) with minor consideration about QoE.	2021
Our presented work	Comprehensive review : (1) vital elements in QoS/QoE-Driven IoT architectures including classification of QoE cause factors, QoE prediction methodologies, QoE optimization and control in perspective of architecture. (2) Model of QoE metrics from different perspective of QoE-related problems. (3) QoE-aware resource management methods : QoE-aware offloading, placement, data caching policies.	Only few details on recent QoE concept and evaluation for Machine-to-Machine, Machine-to-Human, Human-to-Machine in IoT systems.	2022

In brief, our contributions are listed as the following:

1) A survey of crucial elements in QoE-Driven IoT Architecture including classification of QoE Cause Factors, QoE prediction methodologies, QoE optimization and control in the level of IoT architecture. Our reviews are based on recent ML-based approaches. QoE-Driven IoT platform is basically composed of multiple AI-based functions.
2) Model of QoE metrics classified by QoE prediction, optimization and resource management problems.
3) Reviews of QoE-aware resource management methods: QoE-aware computational offloading policies, QoE-aware placement policies and QoE-aware data caching policies. In addition, most recent approaches e.g., ML-based policies are extensively discussed in this paper.

Unlike others, our work provides structural details of how QoE can be factored, measured, modeled, predicted and manipulated with recent ML approaches for IoT systems. Moreover, QoE-aware resource allocation and management of prior works for multi-tier IoT systems are classified and various aspects of QoE objectives and metrics with ML-based solutions are shown. Next, existing IoT architectures are investigated and QoE-Driven IoT architectures are discussed in more details in section IV.

III. EXISTING DESIGNS OF IoT ARCHITECTURES

In this section, various designs of IoT architectures are discussed. Common designs of Layered IoT architectures are evaluated in Subsection A. Then, in Subsection B, current commercial IoT platforms, mostly Service Oriented Architecture, are discussed to determine shared functions. Lastly, in Subsection C, we discuss the necessary QoS components of IoT architectures.

Understanding the current designs of IoT systems can give us insight about QoS and service management. The summary of the existing IoT architectures are discussed in this section.

A. LAYERED IoT ARCHITECTURES

A number of IoT services are widely deployed and distributed over different geographical areas. Since IoT sensing data can be exchanged across domains, novel applications or automated services from IoT services can be invented. Smart applications are created by using these services e.g., intelligent traffic, environmental monitoring, smart city, smart campus and others. It is undeniable that these services have to be executed on some service platforms. To exploit IoT services, we first discuss about current service IoT architectures. In prior works, basic IoT services were mostly constructed by using layered IoT architecture [17], [18], [19], [20], [21], and [22].

Three common layers are shown in most of previous works namely: perception or sensing layer, network layer, application and/or service layer. IoT sensing devices are implemented in perception layer to collect sensing data back to cloud or to forward control command from the cloud to IoT devices. As IoT data are measured at IoT devices, sensing data is exchanged via wide range of communication technologies (4G, 5G, WiFi, LoRA) and protocols (CoAP, MQTT, IP) at network layer. When data reaches an application layer, it is processed, analyzed, stored and visualized by computing resources on Cloud. Applications generally offer services according to user requirements via application interfaces. Sometimes, the application layer is embedded with a service sublayer to manage and allocate services for applications.

Although the traditional IoT architecture is composed of three layers, slightly different architectures with additional layers are proposed in the literature. In traditional designs, IoT services are developed for particular application domain by using the three layer approach. This approach incurs high operating cost on resource management for service provider because of high heterogeneity of IoT environments. Various types of computing, storage and configurations have to be properly maintained by using different methods. Moreover, data migration and service provision are quite troublesome tasks for administrators. More efforts and resources have to be poured into system development if more demands request for new applications. As a result, many research works have focused on designing an IoT platform, which can handle heterogeneous data and services.

In [17], MicroThings architecture attempted to solve the problems of the three layered basis design. MicroThings proposed a framework for three environments: Information aggregation, Centralized control and Application environment. Information aggregation environment enables ubiquitous access network. Centralized control environment provides unified framework and elastic computing for on-demand and heterogeneous computing for various types of devices. Application environment supports application provisioning, deployment, and development.

It is obvious that heterogeneity in IoT system is one of the major issues. Similarly, Krco *et al.* [18] claimed that IoT service architecture should comply with the following features: capability to handle heterogeneous domains, support wide ranges of user requirements, less complex system, common framework for unifying different applications. According to [18], different working groups had studied and designed IoT a service architecture. IoT-I group focused on analysis of various IoT architecture. IoT-A group focused on design of an IoT framework for common service architecture. The group provided Architecture Reference Model (ARM) and a set of their best practices utilized for building IoT architecture. IoT-A group worked in parallel with FIWARE project [23], which attempted to develop the next generation IoT architecture. ETSI M2M and oneM2M groups attempted to define a standard for IoT architectures, which was proposed to have two domains: device and gateway domain, and network domain. Each domain has service layer capabilities embedded for performing M2M functions. In short, common IoT architecture and framework are quite crucial for tackling the heterogeneity of IoT natures.

In addition, Lv *et al.* [19] proposed a general IoT architecture. Service architectures could be designed by either top-down or bottom-up approaches. In the top-down approach, services and applications are used as the starting point, and IoT devices are organized according to service requests. In contrast, the bottom-up approach focuses on connecting things with one another and expanding their connectivity and services through applications. In that work, the bottom-up approach was chosen and 7-layer IoT architecture was proposed. Its layers are composed of sensing layer, intelligent device layer, physical information layer, logical information layer, IoT basic service layer, service middle layer, and application layer. The sensing layer deals with sensor, actuator and network at physical nodes while the intelligent device layer handles terminal and local services at physical nodes. The physical information layer is just an abstraction layer for physical IoT objects while the logical information layer provides an abstraction for virtual IoT objects. The main layer for service abstraction is the IoT Basic Service layer, which is utilized to identify Things, collect data from Things, and control the behavior of Things. The service middle layer supports providing service descriptions, flows, and interactions. The application layer has functions similar to other architectures. As mentioned earlier, two service layers are emerging in their design because wide ranges of IoT services have to be managed at some points of the architecture.

Instead of the service middle layer, Lee [20] proposed a starting point for developing IoT services at Service Management layer. This layer was designed for supporting activities and procedures in organization by enhancing application or solutions in domains of lower levels. Its five layer architecture composes of service management layer, application/solution layer, processing layer, network layer, and perception layer. The IoT architecture is quite similar to the approach in [24], except that there is the processing layer (data management layer). The processing layer (middleware layer) functions are to cleanse, store, analyze, and process data from the network layer. Sample platforms in the processing layer are database management, data analytics, and Cloud Computing for massive data storage, high-speed broadband networks and fast processing speed to enable real-time decision making. The processing layer is similar to the Service Sublayer of the application layer in [24] or the Service middle layer in [19]. Both seem to be designed for data management, data analysis and decision making.

Similarly, Zhong *et al.* [21] proposed 4 layer architectures : perception layer, network layer with two sub-layers, application support layer, and application presentation layer. The application support layer implements Cloud Computing, middleware service, and data analytics. Its function is similar to the processing layer in [20], the Service Sublayer in [24], the Service middle layer in [19]. The application presentation layer manages visualization and interaction in each domain application such as smart building, logistics, and intelligence traffic system. On the other hand, the network layer is divided into two sub-layers : access network (4G,5G,WiFi,Cellular) and core network (Internet, dedicated network). Thus, the service layer is quite mandatory for common IoT service architecture. It should be in the middle between the network layer and the application layer, or locates above the application layer for managing a big picture of all services.

In [22], Sarkar *et al.* proposed Distributed Internet-like Architecture for Things (DIAT), which tackled issues like heterogeneity of IoT devices and also privacy and security. In their architecture, they proposed a new intermediate layer between the application layer and the perception layer. This layer is called IoT Daemon, which is composed of three sub-layers : Virtual Object Layer (VOL), Composite Virtual Object Layer (CVOL), and Service Layer (SL). VOL represented physical objects or entities as virtual objects, which had semantic description for modeling physical IoT objects. CVOL composed of multiple virtual objects such that a certain task could be accomplished by a service from this composite objects. CVOL needs a discovery mechanism for determining optimized composite objects. SL receives a service request from the application layer or creates one on its own for service creation. SL job is to divide each request into sub-tasks that will be processed by CVOL. Still, this IoT Daemon is just another name of the Service layer of other works, [19], [20], [21], [24], except that functional details of this layer are totally different.

In conclusion, various prior studies suggested that a common service IoT architecture was a layered architecture with at least four layers (Perception, Network, Service and Application) as shown in Figure 5 should suffice. In the Service layer, its responsibility should cover the following tasks : service identification and mapping, service selection, service composition, service provisioning, service placement, service access, and service classification. These functions are used to cope with well-known issues in IoT environments e.g., heterogeneity, highly dynamics, privacy and security, management cost.

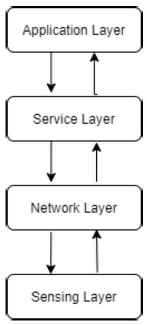

FIGURE 5. General layered IoT architecture.

B. SERVICE ORIENTED ARCHITECTURES

Most of commercial IoT platforms are widely deployed and serviced by big-name companies such as Intel, Baidu, Tencent, Amazon, Google, Oracle, Samsung and Microsoft. In contrast, open IoT platforms are public to various communities such as FIWARE, OpenMTC, SiteWhere, Webinos [25] and Eclipse IoT [26]. Each platform is highly different from others. It is not clear how to group all of them into one unified reference architecture. Some of them are proposed by using layered approach such as Intel IoT platform [27]. Most designs are based on Service Oriented Architecture (SOA) or microservices such as Microsoft Azure IoT [28] and AWS IoT [29].

Thus, Guth *et al.* [25] proposed a reference IoT architecture based on six common component types in IoT platforms. Those six components are Sensor, Actuator, Device, Gateway, IoT Middleware, and Application. They showed that existing IoT platform designs (both commercial and open-source) can be mapped onto their reference architecture. IoT Integration Middleware, supporting multiple core feature functions for managing devices and services for applications, is the main component in this architecture. However, Guth *et al.* [25] did not categorize what functions were suggested to be implemented based on the current platform's IoT Integration Middleware (IoTIM). We extend their results with a set of functions that this middleware should provide to fulfill the desired IoT services : namely message broker, device management, application interface, connectivity management, context management, device discovery, security and identity management, and event and analytical processing. This indicates that any IoT architectures should support these functions.

Firstly, components in IoT Integration Middleware (IoTIM) were elaborated for each platform. The platforms are compared based on the service functions they utilize. Components in IoT Integration Middleware for FiWARE, OpenMTC, SiteWhere, Webinos, and AWS platforms are described as follows. In FiWARE architecture, IoTIM has IoT broker, IoT discovery, IoT device management, and data context broker as IoT backend. In OpenMTC, backend components are composed of connectivity, network exposure, core features, application enablement. For SiteWhere, a tenant engine is responsible for device management, communication engine, REST APIs, Integration, Data Storage SPIs and Asset SPIs. Unlike others, Webinos IoTIM is focused on User Authentication, Policy Repository, Policy Enforcement and Web APIs. Components in AWS IoT are Message Broker, Thing Shadows, Thing Registrys, Rule Engines, Security and Identity.

Next, we consider IoTIM on IBM Watson, Microsoft Azure IoT, Samsung SmartThings, Intel IoT, and Eclipse IoT platforms. IBM Watson IoT architecture has analytic, risk management, connect, information management, open standard services, flexible deployment. In Microsoft Azure IoT, core components of IoTIM are IoT Hub, Event processing and insight, Device business logic, connectivity monitoring, application device provisioning and management. The middleware of Samsung SmartThings has an application management system and subscription processing. According to [27], Intel IoT reference architecture comprises Device and configuration management, Data ingestion software, Service orchestration, security management, analytic software. As discussed in [26], Eclipse IoT core features are device management, application enablement, connectivity and message routing, data management and storage, device registry, event management, analytic, and User Interface, while its cross-stack is composed of security and ontology.

According to [25], [26], [27], all of these components can be grouped and categorized by the following functions : Message Broker, Context Management, Device Management, Device Discovery, Connectivity Management, Security and Identity Management, Event and Analytical Processing and Application Interface parts. It can be seen that Device management, Application Interface, Connectivity Management, and Context Management functions often occur in IoT Integration Middleware. It is known that Context and Device management are responsible for handling IoT context and device information. Also, Application Interface is used by other IoT modules, devices, applications to access all current IoT data. IoTIM is at the center of the platform, so most of other IoT components will have to access IoTIM. This connectivity has to be properly managed. Those are the main reasons why these components often occur in IoTIM of various platforms.

On the contrary, Message broker, Device discovery, Security and Identity management, Event and analytical processing sometimes appear in IoTIM. Security and Identity Management require more complexity and resources on the system. It may be a better choice to separate them as external services in most platforms. Event and analytical processing can be seen as an option for developers or administration of IoT platforms. Similarly, Message broker is used to handle IoT devices with specific protocols, so it can be thought of an alternative to connect to wide ranges of IoT protocols. Device discovery is explicitly defined only in FiWARE, AWS IoT, Eclipse IoT. However, this function is rather compulsory, so it may be indirectly implemented in other modules. Thus, complex tasks such as security management, event and analytical processes can be separated from IoTIM for some platforms.

In short, IoT Integration Middleware is the essential component in existing IoT platforms where its must-have functions are the following : message broker, device management, application interface, connectivity management, context management, device discovery, security and identity management, event and analytical processing software modules. However, existing platform solutions do not explicitly provide details on how to handle QoS in their architectures even though QoS management may potentially be integrated inside the aforementioned services.

C. IoT ARCHITECTURE WITH QoS SUPPORT

In the layered IoT architecture, it composes of four common layers : Perception, Network, Service, Application Layer. Each layer has its own QoS metrics in terms of performances [24]. Precisely, QoS metrics in Perception layer can be energy consumption (service life time), mobility information accuracy, location information accuracy, serving coverage, time synchronization, sampling frequency, sampling precision, or resolution e.g., video. Also, QoS at Network layer can be bandwidth, delay, loss rate, jitter, throughput and response time. QoS at Application and Service layer can be service time, service delay, service priority, service accuracy and service load. These metrics locally associate with individual layer functions because of inheritance from performance tuning.

To achieve end-to-end QoS requirement, negotiation among these layers is unavoidable. Interaction between higher and lower layers to exchange resource information is one approach to ensure QoS by properly scheduling and assigning resources. For instance, Duan *et al.* [24] proposed to construct a new vertical layer called QoS Management Facility layer. This layer interacted with QoS broker at Perception layer and Network layer to negotiate and interpret QoS requirements to local parameters. In short, this approach attempts to control multiple QoS across layers such that a negotiated E2E QoS metric is achievable.

Alternatively, Issarny *et al.* [45] considered Service Oriented Approach (SOA) to design IoT services. IoT service is just another Service Oriented Architecture but the reverse is not true. Moreover, it can be seen that some existing platforms are designed by using SOA and middleware approaches. It is more practical to view Things as an IoT service. An application is just a composition from various IoT services.

Previously, IoT Integration Middleware of various platforms are elaborated and based on Service Oriented Approach. It is shown that a number of certain elements are implemented in existing platforms e.g., message broker, device management, connectivity management, context management and others. Nevertheless, QoS related components are not explicitly defined.

To achieve QoS constraints, additional QoS related elements are necessary by these SOA Middleware approaches. In that paper, a number of QoS-related elements are categorized in Table 2. In the table, must-have QoS elements in IoT middleware can be categorized into five groups : QoS

TABLE 2. QoS control for IoT integration middleware.

QoS Elements in IoT Middleware	QoS Strategies	References
QoS Negotiation	QoS negotiation is mediated between Service Providers and Consumers by using QoS broker	[30]
	QoS negotiation framework is proposed to handle QoS in IoT	[31]
	Context-based negotiation strategy and integrate with Web Service Agreement Negotiation Standard	[32]
	Negotiation framework is used in M2M applications to negotiate the required SLA with standard interface	[33]
	Mixed approach between concession and tradeoff strategy to balance utility and success rate for Cloud service negotiation	[34]
QoS Decision	Extend QoS brokering for SOAs by designing a service selection QoS broker that maximize a utility function for service consumers	[35]
	QoS Decision Making Function and QoS Broker	[36]
	Decision Making for QoS in IoT	[37]
QoS Monitoring	Autonomic middleware level for QoS managements in IoT systems	[38]
	QoS Monitoring used in scheduling process for IoT services	[37]
	Process Monitoring Agent, Sensing Quality and Energy Monitoring Agent	[36]
	To enable QoS support, QoS monitoring and prediction are essential for highly adaptation of changes of QoS in component services	[39]
	QoS is highly dynamic changed with time and places, so monitoring and prediction are critical procedures and the work proposed IoTPredict based on collaborative approaches	[40]
QoS Traffic Management	Novel traffic management policy for achieving QoS of Machine Type Communication	[41]
	Survey in Software Defined Networking realm how to provide QoS in network applications, which is essential for dynamic QoS support because SDN is programmable	[42]
	Automated translation from QoS network constrain into SDN configuration	[43]
	Autonomic QoS management mechanism for SDN network	[44]
QoS Scheduler	Message QoS scheduling algorithm is proposed for classifying emergency and non-emergency messages	[37]
	Three-layer QoS scheduling model for SOA IoT applications	[37]

negotiation, QoS Monitoring, QoS Traffic Management, QoS Decision and QoS Scheduler.

QoS negotiation is an initial procedure before the requested service starts. It is utilized to communicate about QoS contract between Service Providers (SPs) and Clients. The requested service from applications will specify a set of QoS targets to SPs. SPs monitor and control their resources to achieve this QoS request. For instance, in [30], a QoS broker was deployed for negotiating QoS between SPs and consumers for Web Service requests in SOA architecture.

Since QoS negotiation is a critical process for completing QoS agreement, several works proposed negotiation framework solutions. According to [31], a dynamic QoS negotiation framework and Markov Decision Strategy negotiation algorithm were proposed in their work to handle QoS for IoT services. The results show that their method improved success rate, completion time and social welfare in negotiation procedures. Moreover, Li and Clarke [32] proposed a context based SLA negotiation for IoT service domain. Their strategy is automated by attempting to resolve the conflicts for multiple rounds of negotiation with different counter offers for selecting best service scores. It is shown to improve success rate and negotiation time. In [33], negotiation framework was proposed and published as a service for achieving QoS target. Their framework is called BETaaS or Building the Environment for The Things as a Service. In [34], the game theoretic approach was used to design a negotiation framework when conflict of QoS occurred between SPs and Clients. Their proposed method was based on mixed strategy on Nash Equilibrium. Their results show that the mixed strategy outperforms both concession and tradeoff strategy in terms of utility and success rate.

Previously, it was seen that the current QoS and resource status in the system are necessary in the QoS negotiation process. It is unavoidable that the system must have knowledge about QoS status before assigning further resources for their clients. QoS monitoring is mainly used for checking QoS status, so that the system can fulfill QoS requests.

For instance, Middleware QoS management for IoT services was proposed in [38] to monitor and detect QoS degradation, so that QoS actions could be executed to improve QoS level to the target. Moreover, in [37], monitoring QoS was utilized in their proposed QoS-aware scheduling for IoT services. In [36], Process Monitoring Agent, Sensing Quality and Energy Monitoring Agent are important components to handle QoS supports for IoT services. According to [39], QoS monitoring and prediction were used to enable QoS support. IoTPredict was proposed in [40] for monitoring and predicting QoS. Their strategy was designed by collaborative approach.

In addition to negotiation and monitoring, another important component is QoS traffic management. As it is named, QoS traffic management is responsible for dealing with different types of traffic over networks to ensure QoS for End-to-End service. IoT networks are highly dynamic because of the tremendous number of devices and traffic flows. As a result, to achieve QoS, the system must have capability to control various types of traffic.

In prior work [41], Al-Shammari *et al.* attempted to solve network resource starvation problem by proposing a traffic management policy. The policy can prevent degradation of networks and enhance QoS performance of Machine Type Communication. The improvement results are verified by simulation. As shown in [42], Software Defined Networking (SDN) technology can overcome many difficulties and problems of controlled network resources. By separating control and data plane, SDN with network programmable features can help managing network resources better than ever. Karakus and Durresi [42] provided survey in SDN technology and described how it could be utilize in QoS provisioning for network applications. Additionally, Seeger *et al.* [43] proposed an automated translation framework for converting QoS network constraints into SDN configuration. Also, an automated QoS mechanism for SDN network was investigated in [44].

Next, QoS decision and Scheduler will be discussed. QoS decision and Scheduler are quite related. QoS decision is used for making a decision based on predefined rules or dynamic algorithms to solve conflicts in resources when QoS deteriorates. On the other hand, the QoS scheduler is responsible for scheduling requests over resources to achieve the QoS target. A scheduling method can be either static or dynamic.

According to [35], Menascé *et al.* extended QoS brokering with a service selection to maximize utility functions. In [36], QoS Decision Making Function and QoS Broker are the main elements for handling QoS of IoT services. On the contrary, Li *et al.* [37] proposed a QoS scheduling algorithm for classifying emergency and non-emergency messages to improve performance. Then, a three-layer QoS scheduling model for SOA IoT applications was discussed in [37].

As shown in prior works, major components for implementing QoS support in IoT services can be categorized into QoS Negotiation, QoS Monitoring, QoS Traffic Management, QoS Decision and Scheduler. These elements are essential for IoT Integration Middleware to achieve QoS support in any IoT architecture. Next, let us consider designs of architecture that are driven by QoE requirements.

IV. QoE-DRIVEN IoT ARCHITECTURE

As shown earlier, QoS is highly critical for maintaining performances of the system. To support QoS, the architecture of the IoT system has to include various components e.g., QoS negotiation, QoS decision, QoS monitoring, QoS traffic management and QoS scheduler. However, QoS alone is not sufficient to deliver a high enough satisfaction for users because their experiences are not measured. QoE term was defined to quantitatively measure the user experiences. Thus, the system can be controlled in such a way that QoE values are achieved and the users are satisfied.

Unlike other works, vital components in QoE-Driven architecture are classified in the view of system layers as follows.

- QoE common architecture
- QoE Cause Factors associate with system layers
- QoE prediction
- QoE optimization and control

The comprehensive review of the overall architecture and QoE-related parts are shown in this section. Discussion about recent ML approaches in each part are also provided.

Indeed, there is no exact definition for the QoE. Nevertheless, in principle, QoE is proposed to measure how users meet their expectation. Each user has different levels of satisfaction, which are hardly extracted as quantitative measures.

Recently, QoE has become a promising approach to deliver high satisfactory experiences for different user expectations. To deliver such an experience, the system has to implement QoE supports for users. To achieve QoE, it means that the system has to be able to control or configurate system parameters on multiple resources from data source to a destination. QoE implied that end-to-end data flow has been controlled such that the experience at the end user is validated and qualified to be at a good level.

Deployment of QoE features onto the IoT system certainly requires a new ecosystem to the current IoT architectures. First, the service provider has to define which indicators of user expectations will be utilized as QoE feedback information. Next, the system has to be capable of measuring user experiences or QoE. Measuring QoE is a challenging task because user satisfaction varies even if the system environment is the same and there is no exact QoE definition. QoE can be subjectively measured from users, but it is hardly monitored in real-time where the system environment is continuously changing. So, real-time QoE measurement will require the system to estimate or predict the current QoE from the environment. When real-time QoE is feasible, the prediction QoE value will be used for optimizing both configurations and parameters in the system environment. Eventually, the QoE controller is responsible for dynamically applying new configurations and parameters to accomplish the target QoE for users.

As the QoE ecosystem is elaborated, it is seen how QoE affects onto the IoT architecture. At least, additional components, e.g., QoE indicator, QoE measurement, QoE estimation and prediction, QoE optimization and QoE system controller, are considered to be crucial in QoE implementation. QoE indicator tells us how the service provider measure the QoE. It can be either direct or indirect measurement. Based on this indicator, QoE estimation and prediction will be deployed for real-time computation, which leads to QoE optimization. QoE optimization has to collect QoE measurement, and attempt to control system configurations and parameters such that QoE target can be accomplished. Then, QoE controller has to be designed for system reconfiguration purposes.

According to prior information, it is not difficult to see that QoE architecture can be depicted as in Figure 6. In Figure 6, we can divide QoE things into 6 parts, which will be elaborated in the following subsections. Subsection IV-A is related to QoE Cause Factors, which can be divided into two main cause factors : User-related and Application-related factors and System-related factors. In Subsection IV-B, various methods for measuring QoE are explored in prior works. Then, QoE model for training and prediction are provided in Subsection IV-C. Some important QoE indicators are discussed in Subsection IV-C. Lastly, related works about QoE optimization and system controller are described in Subsection IV-D.

A. QoE CAUSE FACTORS

According to Figure 6, we first discuss about QoE Influencing Factors (IFs) before QoE Cause Factors, which is defined in this paper for simplicity of explanation. QoE Influencing Factors are indeed factors that can affect user experiences. Because QoE is rather subjective and dependent on users, QoE IFs can be seen as causes of user experiences. The QoE IFs can result in different QoE results under similar environments.

QoE IFs are classified into 4 different perspective as illustrated in [46] : Human-related, System-related, Context-related, and Content-related IFs. Human-related IFs can be intepreted as personal characters of each user, which is highly dependent on human information e.g., characteristics, gender, expectation, memory experiences and others. These factors lead people to various expectations. System-related IFs are QoS-related parameters and configurations of the system from physical to application layers. Context-related IFs are factors that define the current environment of users, usually physical, temporal, social and economic. Content-related IFs are associated with service and application e.g., codec and chunk sizes in multimedia services.

However, when we consider the system in Figure 6, how we could interpret these QoE Influencing Factors into the IoT architecture is still questionable. Previously, it is seen that QoE can be influenced by human factors, system factors, context factors, and content factors. However, when we consider the relationship between these factors and the system, they cannot be directly mapped onto each layer of the system. These QoE IFs can still be mapped onto QoE-related metrics/parameters in each layer, which are redefined as QoE factors connecting to various layers. For our convenience, the term "QoE Cause Factors" is defined for the simplicity of connecting different parts of QoE IFs into the system architecture. To be precise, "QoE Cause Factors" are system factors that directly affect the QoE value or user experiences from the perspective of the system. We define QoE Cause Factors (CFs) based on the system in Figure 6, so that it is feasible to map QoE IFs into QoE CFs. We consider QoE CFs in Table 3 at each layer according to the system in Figure 6.

Similar to Table 3, the authors in [47] and [48] investigated relationships between QoE IFs and the layered-based IoT reference model. They focused on what QoE IFs are influential in the IoT context and how to extend QoE measures to IoT machines. Also, the authors extend the concept of Quality of Data (QoD) and Quality of Information (QoI) for quantifying the quality of experience when IoT machines are involved.

According to [47], critical QoE IFs are classified into technical, user, and context groups. There are two factors in the technical group: the physical and the network factors. In the physical factors, QoE IFs are related to physical resource constraints at devices e.g., computing power, storage, and battery. In the network factors, QoE IFs connect to network QoS metrics. The system utilized subjective factors such as application usage and user actions to infer user satisfaction in the user group. QoD and QoI are associated with the context group and the application factors. QoD was defined by data accuracy, truthfulness, completeness, and up-to-dateness. Moreover, QoI was defined by the amount of beneficial information to users. Lastly, application factors mean ease of use and how information connects via the user interface. These factors are mapped onto the layered IoT reference model as follows. The physical and network factors are mapped onto the physical and network layer, respectively. Both of them also are mapped onto the virtualization layer. QoD and QoI are linked to the service layer. User subjective and application factors are related to the application layer.

Still, there is no reference IoT model for evaluating the QoE; leveraging the QoE evaluation and control framework is important. Floris *et al.* in [48] proposed the layered QoE management framework for integrating with the layered IoT reference model and evaluating the QoE. There are five layers: the physical device layer, the network layer, the virtualization layer, the combination layer, and the application layer. The physical device layer focuses on QoD acquired from devices; It is then used in the virtualization layer, such as the data accuracy of GPS signal, video quality data (resolution, coding). The network layer acquires network QoS metrics, e.g., delay, bandwidth, and jitter to measure the performance and forward the QoS metrics to the virtualization layer. The virtualization layer determines QoD for each virtualized object (VO) from related QoS metrics. Then, the combination layer determine the overall QoE from the QoD and QoE of multiple VOs. In the application layer, QoE context IFs such as technical factors (device specifications), social factors (user tendency), business factors (operating cost), and environment factors (places) are considered here.

According to [47] and [48], the authors investigated how the QoE can be evaluated in various layers based on the novel concept of QoD and QoI. Unlike [47], [48], we attempt to create the map between each layer's QoE IFs and QoE-related metrics. So, we can gain insight into how to measure, predict, and evaluate the QoE in the system by using associate metrics as input for the QoE model.

In Table 3, QoE IFs are correlated with measured metrics, technologies and control parameters at each layer. QoE IFs means that there are factors having high impact for user expectations and experiences. To measure these IFs, the column metrics demonstrate what we can measure in each system layer. Then, emerging technologies sometimes are associated with how the system can be controlled. Finally, IF-related control parameters are listed at each layer to

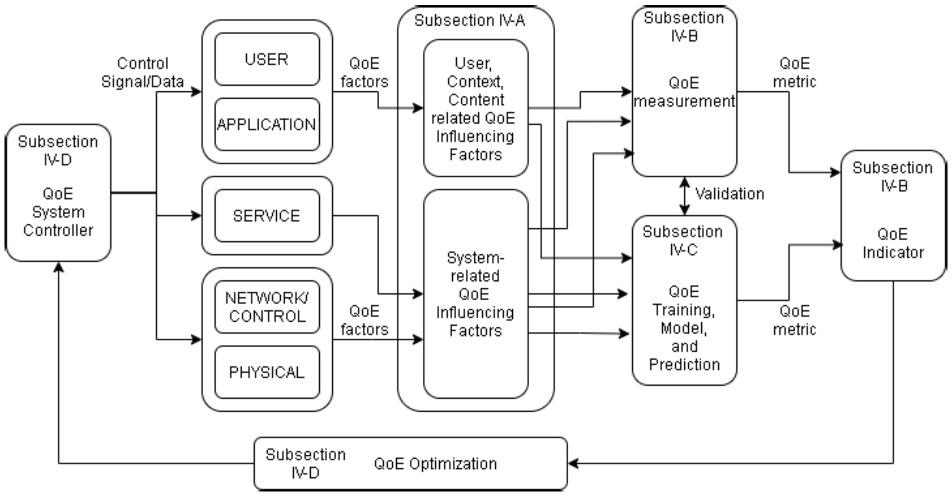

FIGURE 6. IoT architecture with QoE supports.

demonstrate what the system expects to control to achieve the QoE.

First, let us consider User level. User-related and Context-related QoE IFs are associated with user engagement metrics at this layer. User and Context related IFs can be gender, age, social, characteristics, education, user location, background, usage history, living environments, job, personal interest and others. These factors can be measured by using User Engagement metrics such as records of user profile and background, number of downloads, average visit time, screen views per visits, retention rate, user event tracking (e.g., search history), and others [6], [49], [50], [51], [51], [52], [53], [54], [55]. All of these measurements can be utilized by Machine Learning approaches to tailor quality of experience at User level, which is dependent on marketing strategies.

At the Application level, IFs are associated with contents of applications. For instance, video-related IFs are video codec, resolution, video types and others as stated in [46]. Video-related metrics from a system perspective are video distortion rate, average data rate or Video Quality Metrics (VQM) [56]. Examples of configurable parameters for videos are video codec, frame rate and interval, compression ratio, buffer size, chunk size, resolution [5], [46], [56], [57], [58]. For voice-related applications, voice codec, call setup time, call blocking ratio are examples for their IFs. Next, voice-related control parameters are voice codec, call setup mechanism and others. Similarly, voice-related metrics are voice signal impairment, call setup success rate, blocking probability, average call setup time, Mean Opinion Score (MOS), Perceptual Evaluation of Speech Quality (PESQ) [59]. For web-related applications, download and fetch time can be seen as both IFs and performance metrics that system can measure. Since both download and fetch time are related to Network and Physical Layers, the system can control related mechanisms or protocols (e.g. HTTP) for surfing web sites.

At the Service level, service management can be classified into three parts : Device-related IFs, Data-related IFs and System-related IFs. For Device-Related IFs, device registration, device discovery and device management can affect user expectations. Their influences can be quantified by some of the following metrics e.g., registration failure rate, registration time, discovery time, number of errors in discovery process, and ease of usage. Registration, discovery, subscription and notification mechanisms can be designed such that they are appropriated with the system. Next, data semantic, data repository and management, data subscription and notification, transaction management, service charging and accounting are Data-related IFs. For instance, notification failure rate, number of subscriptions, notification success rate, notification time, transaction statistics and service costs can be used to measure what experiences users are encountering. To improve service quality, service optimization can be used to alleviate some of these problems. Lastly, examples of System-Related IFs in Service level are application management, group management, communication (connectivity) management and others. Their metrics are ease of usage, number of connectivity failures, number of hand-over connectivity, number of security breaches. Connectivity mechanisms and security choices can be appropriately designed such that the service quality is acceptable.

Since QoE IFs have a certain degree of correlation with QoS metrics at system layers such as network (control) or physical layer. According to [60], Liotou *et al.* provided studies of the QoE model and estimation, and relationships between QoS and QoE, QoE IFs. The authors provided the relationships between Quality IFs and QoS in aspect of network/physical layers, equipments, wireless conditions, service, and other factors. Although, the relationships between IFs and QoS are described, both are still tightly coupled in their illustration. In contrast, this work attempts to determine the relationships among IFs, measured metrics, technologies and control parameters at different layers.

At the Control or Network layer, its major Influent Factor is focused on network quality. QoS metrics at network level are jitter, delay, packet loss rate, data rate and volume, reliability, congestion indicators, busy period, round trip time [7], [50], [61], [62], [63], [64], [65]. Indeed, these metrics are measured in most networks. Technologies such as SDN [64] and NFV [66] are emerging to help system operators to control these metrics more efficiently. Control variables for SDN and NFV are flow rule/table and node/link parameters respectively.

At the Sensing level, channel quality is the main factor affecting user experiences. In short, QoS metrics, typically used in most systems, are bandwidth, signal strength, bit error rate, dropping probability, power consumption [6], [54]. Even though IoT devices can be physically operated by these technologies : WiFi, NB-IoT, LoRA, LTE, 5G, all of them have similar control parameters such as bandwidth control, subcarriers, modulation scheme, coding, transmission power. These parameters can be used to improve the quality of communication channel as well as user experiences.

Previously, impact factors of user experiences at each layer are elaborated in Table 3. However, recent concepts such as QoD, QoI and QoB (Quality of Business) in [47], [48], and [67] have emerged for evaluating QoE in the IoT context. QoD is the quality of sensed data composed of data accuracy, truthfulness, completeness, and up-to-dateness. On the other hand, QoI is referred to the amount of helpful information for a decision or action. Both QoD and QoI are related to content and context-related IFs. Next, QoB can be seen as QoE system-related IFs in the resource plane, e.g., resource cost and constraints. Most of the prior works exclude resource constraints or expenses from their QoE model. Instead, researchers include these costs and constraints into their optimization objective rather than the models. When QoB is extracted into different levels, it is more convenient for QoE provisioning. Lastly, QoS metrics are mostly related to the system-related IFs in the physical and network layers. These novel concepts have been still developing in recent works.

In brief, we consider what QoE IFs should be at each layer of the IoT system. Then, we attempt to attack the problem of quantifying or measuring these IFs with redefined metrics. Some may be related to emerging technologies. Finally, configurable or control parameters are listed to demonstrate that QoE can properly be controlled by these parameters or tuning mechanisms.

B. QoE MEASUREMENT AND INDICATOR

To measure user experiences, QoE has to be quantified such that the running system is aware of what satisfaction level of users currently are. This QoE indicator will be utilized as the feedback information for the system to adjust controllable parameters as discussed in the prior section at each layer. So, the QoE is converging to the value that user experiences are in high satisfaction level.

One of the well-known QoE indicators is Mean Opinion Score (MOS). It is widely used to measure quality in multimedia services [75]. MOS is quantified into five levels of quality experiences from 1 to 5 (bad, poor, fair, good, excellent). Its measurement can be subjective, objective or hybrid manners. In subjective measurement, MOS is measured from users either in experiments or real usage. Many previous studies [75] attempted to determine relationships between MOS and multimedia parameters by conducting several different subjective measurements. The subjective experiments usually require more time to give an accurate result. In contrast, objective MOS is used as the estimation of subjective MOS (automated subjective measurement). The MOS approximation formulas are derived from the MOS subjective experiments, which are standardized and conducted in multimedia applications [75].

Various methods for MOS subjective measurement are used in prior studies [75] such as Absolute Category Rating (ACR), Double-Stimulus Impairment Scale (DSIS), Double-Stimulus Continuous Quality Scale (DSCQS), Subjective Assessment Methodology for Video Quality (SAMVIQ), Degradation Category Rating (DCR) and others. They are commonly used to measure QoE in multimedia applications such as video streaming and voice/speech transmission. Even though, MOS subjective methods are designed to measure and quantify user experiences into different satisfaction levels. They are not practical in the dynamic system where the quality of services is continuously changing over time. To measure QoE in real-time, MOS objective measurement is used instead of subjective one. As shown in [56], [60], they were formulated by using Content-related parameters to calculate MOS score e.g., Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM) and Video Quality Metrics (VQM).

Novel QoE subjective measurement is actively investigating because of more users demanding of higher satisfaction level. Most prior QoE indicators are designed for multimedia applications. Several new studies proposed a new framework for measuring subjective QoE in IoT systems such as [76]. According to [76], Suryanegara *et al.* proposed a new QoE measurement framework based on Absolute Category Rating with Hidden Reference (ACR-HR) scale. Their measurement has two phases : measuring experience before and after IoT implementation. When MOS can be derived from both phases, Differential MOS (DMOS) is computed on basis of ACR-HR scale method. DMOS can be interpreted as the strong and weak points of implementation, so an improvement of service implementation can be found. However, existing QoE subjective measurement is still based on conservative methods derived from prior multimedia applications. More studies on new QoE subjective model will be essential for QoE-Driven IoT platform.

Unlike in the multimedia context, recent studies in the IoT domain [67] proposed a new method to measure the QoE

TABLE 3. QoS cause factors.

IoT Layer	QoE IFs	QoE Cause Factors/Metrics	Technology and Control Parameters
User	User and Context related IFs : gender, age, social, characteristics, education, user location (e.g., home, workplaces), live style, background, usage history, living environment, workplace, job, personal interest, memory experiences	user engagement metrics : user background and profiles, number of downloads, average visit time, screen views per visits, retention rate, number of active users, user event tracking (e.g., search history), user satisfaction metrics (e.g., app rating, touch heatmap, in-app feedback), user perception metrics (app crashes, speed, latency) [49], [50], [51], [52], [53], [54], [6], [55], [51]	marketing strategies and tuning designs for users e.g., machine learning approaches for tailor-made experience for each user
Application (Content-related)	video-related IFs	buffering, quality switches, start-up time, codec, resolution, frame rate, video distortion rate, average data rate, Video Quality Metric [56], [46], [57], [58], [5]	video control parameters : codec, frame rate and interval, compression ratio, buffering [68], chunk size [57], resolution
	voice-related IFs	voice signal impairment, call setup success rate, blocking probability, voice codec, average call setup time, Mean Opinion Score (MOS), Perceptual Evaluation of Speech Quality (PESQ) [59]	voice control parameters : codec, call setup protocol, VoIP protocols
	web-related IFs	download time, fetch time, number of failed downloads, average/maximum download or fetch time, average data rate	web control parameters : web protocols
Service	Device-related : registration, discovery, device management, Data-related : data semantic, data repository and management, data subscription and notification, transaction management, service charging and accounting	registration failure/success rate, registration time, discovery time, number of errors in discovery process, ease of usage notification failure/success rate, number of subscriptions, transaction statistics, service costs, storage costs	registration, discovery, subscription and notification mechanisms and protocol service cost optimization
	System-related : security, application and service management, group management, communication management, location	number of connection failures, number of handover connection, number of security breaches or satisfaction in management of services	connectivity and security choices
Network (Control)	trade-off between network quality and operation cost	jitter, delay, packet loss rate, data rate, data volume, reliability, congestion indicators, busy period, round trip time [50], [61], [62], [63], [64], [65], [7]	Sofware Defined Network (SDN) : flow rules and flow table [69]
			Network Function Virtualization (NFV) : node and link parameters [70]
Sensing	trade-off between channel quality and power usage	bandwidth, signal strength (SNR, SIR, SINR), bit error rate, dropping probability, power consumption [6], [54]	WiFi : channel selection, modulation, coding, transmit power, rate adaptation [71] NB-IoT : bandwidth, subcarriers, modulation, coding, guardtime, transmit power [72] LoRA : bandwidth, spreading factor, transmit power [73] LTE, 5G : bandwidth, coding, modulation, frame period, subcarrier, code rate, channel quality indicator, transmit power [74]

when interaction among machines is involved. Nevertheless, interaction in the IoT context often occurs between machines or machine-to-machine (M2M) communication without user intervention. As a result, personal feedbacks from users are sometimes not feasible. Subjective tests are not even practical. As a result, Minovski *et al.* in [67] introduced a novel term to define the quality of experience in the IoT paradigm: Quality of IoT or QoIoT. This term is utilized to model the QoE from the users' perspective, including both objective or subjective factors and machine experiences or Quality of Machine Experience (QoME). The authors use quality information of data, network, and context to evaluate the overall QoE in the intelligent IoT machine.

From a real-time perspective, it is difficult and quite infeasible to measure actual QoE from users instantly. Because actual QoE is involved with usage feedback from users, estimating and predicting QoE by using objective methods are a better approach. As a result, many prior works proposed QoE-related things by using objective QoE metrics. QoE metrics can be estimated from various factors : network delay and quality, [6], [50], [54], [61], [62], [63], [64], [77], [78], latency [79], energy consumption [80], [81] processing and completion time [82], resource states [81], [83], content-related metrics (e.g., voice, video, image) [84], [85], [86], [87], and user-related metrics [49], [81] and others. For more details, a summary of QoE indicators from the literature is reviewed in QoE Metric Section.

Previous discussion has been focused only on either subjective or objective. However, recent research works in Artificial Intelligences (AI) allow us to exploit novel Machine Learning (ML) techniques for predicting or estimating QoE. According to Figure 6, it will use system and service metric data along with valid QoE measurement to train and predict the QoE model. Related works will be discussed in the next section.

C. QoE PREDICTION

Unlike QoS metrics, QoE measurement is quite subjective and requires a long and tedious procedure. It is not difficult to see that QoS metrics and QoE indicators have some correlations. The relationship between QoS and QoE measurement has been studied by many prior works such as [60], [88]. QoE estimation with various objective models were provided, so that complicated subjective measurement could be avoided. In [88], QoE metrics are focused on video quality assessment, and one QoE metric can be correlated to many QoS metrics. It can be seen that many works attempted to solve real-time QoE prediction based on relationships between QoS and QoE metrics.

QoE prediction has become necessary mechanism for controlling system to achieve higher satisfaction level. As depicted in Figure 6, QoE prediction is rather new for IoT architecture. It is required feedback information from system, service, application and user metrics for predicting a new QoE. It can be thought of as an AI-related part because quantifying and predicting QoE is quite subjective, and various ML techniques are good at predicting things.

These ML techniques help the system predicting instant QoE values, and QoE can be feasibly controlled in real-time. Classical ML techniques, e.g., Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF) and Neural Network (NN), have been studied and proposed to predict QoE value for various systems. Example of these works are illustrated in [50], [51], [61], [62], [63], [77], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98].

In [50], Mushtaq *et al.* studied system and service factors having performance impact on QoE. Also, the correlation between QoE and QoS metrics is investigated. The authors provided MOS calculation methods from QoE data sets, which are based on classical Machine Learning approaches. They conducted a comparison among ML techniques such as Naive Bayes, SVM, KNN, DT, RF and NN. They showed that DT gave the best MOS estimation.

Abar *et al.* [89] evaluated 4 basic supervise learning techniques e.g., Decision Tree, Neural Network, K Nearest Neighbors and Random Forest for QoE prediction. These algorithms were utilized to predict QoE in real-time based on full service parametric such as SSIM, VQM and video application metrics such as frame rate, data rate, resolution. Also, the authors use Degradation Category Rating (DCR) as subjective measurement for building the training model and validation. Their simulation result on SDN-based network shows that the best ML method for predicting QoE is the Random Forest algorithm.

According to [61], QoE prediction was performed by using network information (e.g., Round Trip Time (RTT), jitter, bandwidth, delay) and objective parameters (e.g., VQM, PSNR, SSIM). The MOS estimation is conducted in real-time in SDN-based network. First, they collected MOS and network parameters as data sets. Then, these data sets were utilized for training Machine Learning Regression models. The authors also proposed an adaptive enhancing QoE mechanism that monitors and predicts the current value of QoE, then it modifies network configurations to control QoE.

According to [62], Menkovski *et al.* provided a theoretical discussion on how to apply QoS parameters (e.g., delay, jitter) to predict QoE. They built a data set from a collection of system data and subjective test scores from end users. Then, online ML algorithms based on SVM and DT were applied to predict the QoE value.

Evaluation of QoE prediction models from four classic ML methods are conducted by Charonyktakis *et al.* [63]. These four ML methods are based on Artificial Neural Network (ANN), SVM, DT, and Gaussian Naive Bayes. Their experiments were performed on voice applications and utilized QoS metrics (e.g., jitter, delay, packet loss) as input parameters for prediction.

Customized NN and Bayesian techniques for predicting QoE was applied by Vasilev *et al.* [77]. Network QoS parameters such as duration of streams, number of TCP incoming and outgoing packets, average delay, jitter, packet loss rate, packet retransmission, and others were used for estimating QoE.

Three ML-based algorithms such as KNN, RF and DT are applied to predict QoE, MOS scores by [51]. The MOS subjective measurements were used to determine QoE impact factors related to both social context factors and user engagement metrics. Then, these user and context metrics were used in QoE prediction model.

In [91], a Feed-Forward Back-Propagation Neural Network (FFBPNN) was deployed to predict MOS when input parameters are network-related factors (e.g., network capacity, network bandwidth, network path constraints). The authors extended the QoE prediction to maximize MOS by using a multimodal data fusion technique. This technique was the first attempt to create a relationship between external factors (user data) and internal factors (system data). Then, the fused results were used to construct a QoE model. The proposed algorithm was applied to optimize QoE on Multimedia IoT applications (MIoT).

According to [92], [93], [94], the authors proposed a ML regression-based approach for estimating the QoE by using network QoS data which is derived from probing networks via ICMP packets. Moreover, Mustafa *et al.* proposed to apply ML methods such as RF, k-NN and ANN for approximating the QoE from network QoS metrics in [95]. Next, a novel ML-based system is invented in [96], [97] for detecting the live video streams and predicting the QoE from network characteristics LSTM Neural Network is utilized for detecting the live video stream. On the other hand, Random Forest technique is utilized for creating the ML classifier model for inferring the QoE. QoE predictions with ML algorithms e.g., decision tree, k-nearest neighbors, and support vector machine are investigated by Laiche *et al.* [98].

Recently, emerging ML techniques, e.g., Deep Learning (DL), Deep Reinforcement Learning (DRL), Deep Neural Network (DNN), Deep Q-Learning Network (DQN) and others, have been applied to improve the QoE prediction problem. These research works are demonstrated in [52], [53], [58], [68], [78], [90]. Still, novel ML-based QoE prediction methods are being actively investigated.

Lopez-Martin *et al.* [90] studied three types of prediction models : classical ML methods, deep learning methods and deep learning with Gaussian Process (GP) classifier. For deep learning methods, they conducted the experiment on both Convolution Neural Network (CNN) and Random Neural Network (RNN). Their results showed that deep learning with GP classifier gave the best prediction model. Also, they proposed a QoE detection and prediction based on the combination of CNN and RNN model to estimate QoE. They used network information from packets to predict QoE and binary classification for detecting anomalies on video transmission.

A large-scale data set is collected for training QoE model, where data is composed of four types of subjective scores and 89 network parameters by Tao *et al.* [52]. The authors developed Deep Neural Network (DNN) for determining network parameters and QoE relations, then it was applied for predicting QoE scores for mobile video applications.

A hybrid network between DNN and Improved Recurrent Neural Network (IRNN) was invented to evaluate the QoE indicator by Yue *et al.* [53]. Due to non-seqential and sequential factors from users, system, context IFs, the authors attempted to integrate both of these factors into different layers of RNN. Non-sequential information was incorporated into Input Layer of RNN. On the other hand, sequential information was included into Attention Layer of RNN. Their method was applied on Internet video applications, and it was shown to outrun other QoE evaluation methods.

In [58], Lekharu *et al.* proposed a sequential DNN, specifically LTSM-CNN as a QoE prediction model. It is claimed to have capable of longer memory duration than other CNNs. So, the prediction is more accurate than others. LTSM-CNN is applied for video quality estimation of three QoE metrics e.g., perceived video quality, buffering time, and video quality switches. Moreover, Reinforcement Learning (RL) is applied for learning control mechanism to adaptively stream video data to users while maintaining the QoE. RL is implemented on top of Dynamic Adaptive Streaming over HTTP (DASH) framework or Adaptive Bit Rate (ABR) over HTTP. Their results are shown that improvement of video quality occurs while degradation in system parameters is reduced. Similar to [58], Hou and Zhang [78] proposed a Deep Reinforcement Learning (DRL) to adaptively adjust streaming rate such that QoE was optimized. Their ABR algorithm exploited Deep Q-Learning Network (DQN) to improve QoE.

Real-time QoE measurement based on Deep Learning was used over encrypted traffic by Shen *et al.* [68]. Their QoE metrics include start-up delay, rebuffering and video resolution. The authors proposed a new QoE prediction called DeepQoE, which is based on RNN with RTT as input data. It was shown that their estimation was improved when compared to state-of-the-art methods.

In brief, it is shown that many QoE predictions involve with ML techniques, which require training and validation data set, novel Neural Networks, learning algorithms, QoE models, and relationships between QoE and user/service/system metrics. For this reason, QoE prediction stirred a new structure of IoT architecture with QoE supports.

D. QoE OPTIMIZATION AND CONTROL

If QoE prediction can be feasibly performed in real-time, QoE-aware system controller targets to achieve its optimal objective. Control policies in prior works are performed at one or more of these layers : physical layer (e.g., transmission control, power control) [6], [54], [99] network layer (e.g., network cost) [54] or control layer (e.g., SDN-based controller, NFV-based controller) [6], [7], [64], [65], [66], [99], application (content-related) layer (e.g., rate control, video-related parametric control) [5], [57], [58], [64], [66], [78], and user layer [54].

It is shown that QoE-aware control policies are mostly related to delivery contents (e.g., video services). Since video service has a unique characteristics in quality perspective, various video-related parameters (e.g., buffering, start-up delay) and quality metrics (e.g., SSIM, VQM) can be included into control policies. Also, there is a content-related framework, called Dynamic Adaptive Streaming over HTTP (DASH), purposely designed to adaptively control data rate for video streaming services. Prior works such as [5], [57], [58], [66] incorporated video-related parameters and stated into their control policies. In addition, the control layer is often a place where control policies are utilized in SDN-based or NFV-based networks. Since SDN or NFV enabled networks allow us to control end-to-end delay, many previous studies attempted to propose control policies to optimize user experiences by these technologies. Examples of those works are [6], [7], [64], [65], [66], [99]. Still, control of network configuration can be performed in network layer as in [54] if not SDN-enabled network. Some other works in [6], [54], [99] concentrated on control policies over transmission rate and power. Lastly, He *et al.* [54] attempted to take user experiences into QoE consideration.

Recently, most works have studied design control policies by using emerging ML-based techniques. Thus, we focus on recent ML-based control policies that are devised for QoE improvement. A summary of ML-based approaches for QoE optimization and control is demonstrated in Table 5. For simplicity, we divide the literature into content-related policies and network-related policies because some works mixed different control points into one policy.

First, let us consider content-related policies in terms of rate adaptation control and content-related parameters as shown in [5], [57], [58], [78], [99]. Details of these works are elaborated below. In [57], Liu *et al.* proposed a Deep Reinforcement Learning (DRL) for Dynamic Adaptive Streaming HTTP (DASH) to improve video QoE. The reward function of Deep Q-Learning is a chunkwise subjective QoE, and their objective is to maximize this QoE. The authors also proposed to modify NN and learning process for faster ABR algorithm's convergence. In short, the rate adaptation algorithm is designed by using DRL, so QoE is maximized. This work is similar to [58] and [78] as illustrated in QoE Prediction Section. In [58], Lekharu *et al.* used DNN as a learning network while the RL algorithm is designed for rate

TABLE 4. QoE prediction.

Paper	application	input type	QoE prediction input parameters	approach	simulator	year
[50]	video	system	network quality, video parameters and user profiles with QoS-QoE correlation approach	Naive Bayes, SVM, KNN, DT, RF and NN	Network Emulator (NetEm)	2012
[89]	video	content	full parametric such as SSIM, VQM and video application metrics e.g., frame rate, data rate, resolution	DT, NN, KNN, RF	SDN simlation (Mininet)	2017
[61]	video	network + service	network information (e.g., Round Trip Time (RTT), jitter, bandwidth, delay) + objective parameters (e.g., VQM, PSNR, SSIM)	Regression ML	Simulation	2017
[62]	-	network	Network QoS (e.g., jitter, delay)	SVM, DT	N/A	2010
[63]	voice	network	Network QoS (e.g., jitter, delay, packet loss)	ANN, SVM, DT, Gaussian Naive Bayes	MATLAB	2015
[77]	video	network	duration of streams, number of TCP incoming and outgoing packets, average delay, jitter, packet loss rate, packet retransmission	custom NN, Bayes	ns-3 simulation	2018
[51]	video	user + context	social context factors and user engagement metrics	KNN, RF, DT	Numerical Studies	2021
[91]	MIoT	network	network-related factors (e.g., network capacity, network bandwidth, network path constraints)	FFBPNN	Simulation	2018
[92], [93], [94]	video	network	network QoS from ICMP probing	regression approach (extreme gradient Boosting)	Emulated network	2020, 2022, 2022
[95]	video	network	network QoS metrics	RF, k-NN, ANN	Emulate network (Mininet)	2021
[96], [97]	video	network	network flow data e.g., chunk features	LSTM and RF	Real testbed	2021, 2021
[98]	video	user + context + system	user and context data	DT, k-NN, SVM	N/A	2021
[90]	video	network	network information from packets	DL + GP classifier + combination of CNN and RNN	Simulation	2018
[52]	video	user + network	4 subjective scores and 89 network parameters.	DL + DNN	N/A	2019
[53]	video	user + context + system	non-sequential + sequential factor from user, system, context IFs to layers of RNN	DNN + IRNN	Numerical Studies	2020
[58]	video	content	perceived video quality, buffering time, and video quality switches	DRL + DNN (LTSM-CNN)	Simulation (controlled LTE env.)	2020
[78]	video	content	video-related metrics (buffer size, initial playback delay, rebuffering, video quality switching metrics)	DRL + DQN	N/A	2020
[68]	video	content	start-up delay, rebuffering and video resolution	DL + CNN	Simulation	2020

adaptation under DASH framework. In [78], DQN is their learning network, and DRL algorithm is applied for adapting streaming rate too. Deeplive, a new DRL-based algorithm, was proposed to maximize QoE for live video streaming services by Tian *et al.* [5]. The authors focused on QoE metrics such as rebuffering, latency, bit rate switches, frame skipping and resolution. Moreover, they improved DRL learning time by exploiting a quick start method with rate-based algorithm and historical data. They showed that both learning time and QoE were improved significantly. An adaptive transmission control based on Deep Reinforcement Learning for 3D video streaming was proposed by Zhou *et al.* [99]. The authors collected feedback information from video playback as QoE information. Also, LTSM networks were deployed for bandwidth and viewport prediction. Instead, historical information of the blocks was used for Actor-Critic Network to control transmission speed over Mobile Edge Computing (MEC) nodes in SDN-based networks. It was shown that their approach helps improving user experiences.

Next, control policies related to SDN/NFV-based and network-based controllers are discussed in [6], [7], [54], [64], [65] and [66]. Some of them such as [64] and [66] include content-related parameters into their QoE optimization. Also, Moura *et al.* [6] and He *et al.* [54] incorporate physical layer control into their problems. Their details are described in the followings. QFlow was proposed to control reconfigurable flow parameters in SDN-based networks by Bhattacharyya *et al.* [64]. The algorithm is designed by using Reinforcement Learning technique. The authors focused on how to assign each client to proper (priority) queue so that QoE could be improved. Their objective QoE metrics are based on network QoS metrics and video player state (e.g, buffered video time). In high load situation, their algorithm outran well-known solutions in terms of user experiences. In [65], a ML-based algorithm was proposed for computational offloading from IoT nodes to Fog servers in multi-hop IoT networks. Their objective is to improve QoE by minimizing end-to-end delay and maximizing reliability of network path by using adaptive switching in SDN controller. The problem was Multi-objective optimization, and a regression-based KNN algorithm was designed to determine the best set of source and destination nodes. ML approach was used to design a QoE-aware Cognitive Learning Network framework for optimizing QoE on SDN-based networks by Wang and Delaney [7]. In their framework, Aggregation, Training and Prediction modules are added on the current SDN architecture. First, SDN system parameters and application states are feed into the Aggregation module, and only high impact metrics will be selected through Training module. In the Training module, supervised learning e.g., classification and regression will be used to build a QoE prediction model. Then, the prediction value will be evaluated and tested in Prediction module. The authors applied the relationship between QoS metrics and QoE to estimate QoE. In [6], power usage of wireless nodes were optimized by using the RL approach and SDN technology. Since SDN-based wireless network allow auto-configuration for wireless nodes, automated power control and channel selection on these nodes are feasible. For this reason, the RL algorithm is designed to maximize QoE by controlling transmission power and channel selection via

SDN controller. Their QoE prediction was based on SVM, and it retrieved system parameters such as physical rate, medium busy as inputs for MOS estimation. Wang *et al.* [66] proposed to use RL for selecting Initial Video Segment (IVS) of video streaming services. Since IVS parameters affect to QoE performance, a novel framework called Rldish was deployed at Edge of Content Delivery Network (CDN) to choose IVS which was the best for QoE. QoE is observed in real-time at Edge nodes, and it is used as a reward function in RL algorithm. Their QoE metrics are defined as a weight combination of a maximum start-up latency, general latency and buffering time. Then, Rldish was deployed as Virtualized Network Function (VNF) in HTTP cache server for supporting HTTP-based live streaming. The average QoE is significantly improved by this proposed framework. A DRL-based algorithm was proposed to tackle resource allocation problems on cache capacity and transmission rate over constraint networks by He *et al.* [54]. Also, a QoE model was proposed by including network costs and user IFs. Their simulation results showed that QoE was improved for tested IoT system.

In conclusion, QoE optimization and control often take place at content-related and control layers, which are the most crucial part affecting QoE. However, user experiences can be worsen if the system misses to incorporate other important factors in other layers e.g., power control, transmission bandwidth, or resource capacity. As a result, there is still a room for future control policies incorporating all critical QoE IFs into the design.

V. IoT APPLICATIONS AND QoS/QoE METRICS

IoT applications have a wide range of QoS requirements. It is best to know what QoS metrics of existing applications are. So, these metrics can be used for modelling QoE later. First, we consider important QoS metrics for existing IoT applications in Subsection V-A. Given these QoS metrics, it is not suffice for emerging application e.g., multimedia IoT applications, which demand on high computational resources. Next, a summary of QoE metrics in current literature is provided and is classified by optimization problems in Subsection V-B. Table 7 provides insight of how QoE is modeled for various types of problems.

Details in various perspectives of the QoS and QoE metrics associated with each layer are elaborated. Unlike others, a comprehensive review of QoS/QoE metrics is extensively investigated in this section.

A. QoS METRICS AND IoT APPLICATIONS

QoS requirements rely on IoT applications. For example, some of them demand extremely low packet error rate while others focus on delay or throughput. Specifically, video streaming application may require low delay and high throughput for general multimedia applications. In contrast, it requires an extremely low error rate when video streaming is used for medical applications. Even both are video streaming, QoS requirements are vastly different. Determining of QoS metrics is essential for measuring the application performance, which is related to user satisfaction. On the other hand, it also determines the underlying technologies the service providers have to deploy to support their services. When QoS metrics are determined, service providers can consider choices of their technologies.

To determine QoS metrics, applications from various areas are investigated to see which metrics are mostly concerned in IoT domain. We consider various domains of IoT applications such as Multimedia, Industrial IoT, Military, Healthcare and Transportation applications. QoS metrics of these applications are listed as the followings : jitter [100], [101], [102], latency [100], [103], packet loss [100], [101], [104], [105], [106], [107], [108], throughput [101], [102], [103], [104], [106], [107], [108], [109], [110], [111], [112], delay [102], [104], [105], [106], [107], [108], [109], [111], [112], [113], [114], reliability [110], [112], [114], [115], [116], [117], [118], availability [113], [116], [118], scalability [115], energy [107], [109], [118], round-trip-time [101], [116], maintainability [116], and response time [117]. These metrics are often utilized to measure performance of IoT applications. Nevertheless, metrics such as packet Loss, throughput, delay and reliability seem to be used more frequently than others. Specifically, these metrics are defined at network layers.

Moreover, each application seems to focus on different group of QoS metrics. For example, healthcare applications [106], [107], [108], [111] are mainly concerned over QoS like packet loss, delay and throughput because nature of healthcare applications require low delay and packet loss. Loss of little information can highly affect patients or doctors who use the applications. In transportation, it was shown by [102], [112], [117], [118] that reliability, delay and packet loss were used more than others because transportation applications require less delay and more reliable from service providers. Some metrics such as jitter and energy may be concerned in some applications. Jitter was focused on multimedia and industrial information types which was defined as QoS in [100], [104], [113]. Energy was a key performance indicator in industrial applications e.g., [101], [103], [109], [115] because of its costs.

Next, the relationship between application types and QoS metrics is investigated. It is shown in Table 6. According to Table 6, applications in Smart home/building, SmartGrid, Healthcare, Smart Environment, Industrial IoT, Transportation are discussed. These applications have related QoS metrics such as data rate, data volume, packet loss rate, latency, and reliability. In addition, IoT devices can physically communicate via the following technologies such as LTE-M, NB-IoT, LoRA, WIFI, Bluetooth (BLE), Zigbee and RFID. Each has different capabilities in terms of coverage, data rate, power consumption and topology. The choice of physical technology will be directly correlated to range of applications and QoS requirements.

According to Table 6, QoS metrics are highly correlated to the application domain. A certain type of application indirectly obligates QoS metrics for IoT service providers. In smart grid, most applications in sub-domain

TABLE 5. QoE optimization and control.

Paper	Control Place	strategies	QoE model	Controller	Approach	Simulator	Year
[57]	content-related layer (video metrics, rate control)	DRL-based ABR algorithm with modified NN to maximize QoE in video streaming services	video-related metrics (video chunkwise QoE)	ABR	DRL, RL	Simulation	2018
[58]	content-related layer (video metrics, rate control)	RL-based ABR algorithm with DNN for improving QoE with DASH framework in video services	video-related metrics (buffering time, quality switches)	ABR	RL + DNN	Simulation	2020
[78]	content-related layer (rate control)	DRL-based ABR algorithm with DQN to optimize QoE in video streaming service	network quality + video-related metrics (buffer size, playback delay, rebuffering, video quality)	ABR	DRL + DQN	N/A	2020
[5]	content-related layer (video quality metrics, rate control)	DRL-based algorithm, Deeplive, is devised to maximize QoE for live video streaming services under DASH framework	video-related metrics (rebuffering, latency, bit rate switches, frame skipping and resolution)	ABR	DRL	DASH streaming simulator	2019
[99]	physical layer (transmission rate) + control layer (SDN-based)	DRL-based algorithm with LTSM networks for optimizing QoE in SDN-based Mobile Edge Networks	network quality + video-related metrics (playback feedback)	ABR	DRL + LTSM	Simulation	2021
[64]	control layer + content-related layer	QFlow, RL-based algorithm, is proposed to control reconfigurable flow parameters in SDN-based networks for improving QoE in video services.	network quality + video-related metrics (player state)	flow configuration in SDN-based networks	RL	Simulation	2019
[65]	control layer	ML-based algorithm for minimizing end-to-end delay and maximizing path reliability by adaptively switching in SDN controller.	network quality (delay, link failure)	SDN-based controller	Supervised learning (a regression-based KNN algorithm)	Simulation	2020
[7]	control layer	ML-based prediction framework for QoE estimation and QoE optimization on SDN-based networks	network condition + user perception (QoS-QoE correlation model)	SDN-based controller	Supervised Learning (classification and regression)	N/A	2019
[6]	physical layer (power + channel) + control layer (SDN-based control)	RL-based algorithm for power control and channel selection to improve predicted QoE by using SDN-based controller	MOS (network quality + user perception)	SDN-based controller for power and channel selection	RL	Real Testbed	2020
[66]	control layer (NFV-based) + content-related layer (video parameters)	RL-based algorithm for selecting Initial Video Segment to maximize QoE on NFV-based Edge networks	video-related metrics (start-up latency, buffer ratio/time)	NFV-based controller for selecting IVS parameters (video-related parameters)	RL	Real Testbed	2020
[54]	physical (transmission rate) + network (network cost) + user-related (cache)	DRL-based algorithm for tackling allocation problems on transmission rate, network cost and cache capacity over constraint networks	MOS (network cost + user IFs)	transmission rate, network control and cache capacity	DRL	Simulation	2020

accept medium to high latency and medium data rate, whereas it typically requires high reliability around 98-99.9 percent. Because controlling high-voltage devices in electrical system does not require real-time operation, but very high reliability to prevent the system from failures. In smart home/building, the application requires low data rate, latency in minutes but high reliability. This application is not required long range communication, and devices can be readily installed in houses or buildings. In healthcare, packet loss rate and latency are required to be very low while data rate is used from medium to high. The reason is that health status or information from both patients and doctors is critical for diagnostic and emergency response. Moreover, there is no QoS restriction for environment applications except latency in a few seconds and update frequency. Environmental data has no urgently direct impact to human when compared to other domains. Industrial IoT such as factory automation requires super low latency (0.25-10 ms) and packet loss rate (10^{-9}) because every second incurs operation cost. When it comes to the transportation domain, most applications require low latency and high accuracy (low packet loss rate), but not high data rate and volume.

In brief, each layer has its own QoS control methods. QoS can be controlled in various ways for different types of networks. However, for End-to-End QoS, a choice of physical technology and network types, QoS metrics and application requirements have to be consistent, so it is feasible to achieve QoS constraint. To achieve QoS, a set of feasible combination from physical and network technologies have to be managed or scheduled by service providers. Their methods will be dependent on the structure of their systems.

B. QoE METRICS

In a system, QoS metrics can be readily drawn into anyone's mind. However, when we discuss about QoE, it requires an understanding of the underlying structure of QoE models. Existing QoE metrics collected from previous works are demonstrated that QoE can be modeled by a combination of system, service, application and user related metrics. Table 7 is a summary of existing QoE metrics in the literature.

According to Table 7, QoE can be modeled and estimated by incorporating one or more metrics from network-related metrics, system-related metrics, content-related metrics, and user-related metrics.

TABLE 6. Related QoS metrics for application.

Layers	Related QoS Metrics					references
Metrics	Data Rate	Data Volume (Bytes)	Packet Loss Rate	Latency	Reliable	
Smart Home/Building						
- Home/Building Automation	< 100 kbps			min	High	[119]
SmartGrid						
- Smart Metering (meters to utility)	$>$ 100kbps	100B-MB		sec,hr	$>$ 98-99.5	[120],[119]
- Pricing Application (utility to meters)		100		min	$>$ 98	[120]
- Electric Service Prepayment (utility to customers)		50-150		min	$>$ 98	[120]
- Demand Response (utility to customers)		100		min	$>$ 99.5	[120]
- Service Switch Operation (utility to meters)		25		min	$>$ 98	[120]
- Distribution Automation	$>$ 18 kbps	25-1000		sec	$>$ 99.5	[120],[119]
- Outage and Restoration Management (meters to OMS)		25		sec	$>$ 98	[120]
- Distribution Customer Storage (DAC to storage)		25		sec	$>$ 99.5	[120]
- Electric Transportation (utility to customers)		100-255		sec	$>$ 99.5	[120]
- Wide Area protection, control and monitoring		4-200		msec,sec	$>$ 99.9	[120]
HealthCare						
- Blood pressure monitoring	$<$ 10 kbps					[121]
- Blood sugar monitoring						
- Vital sign monitoring						
- Electrocardiogram (ECG)	1-20 kbps		0%	1 s		[121]
- Real-time Robotic Service			0%	$<$ 300 ms		[121]
- Audio	4-25 kbps		3%	150 - 400 ms		[121]
- Video	32-384 kbps		1%	150 - 400 ms		[121]
- High Quality Video	0.64 - 5 Mbps		0%	100 - 300 ms		[106]
- Ultrasound, cardiology, radiology	256 kB					[121]
- Scanned X-ray	1.8 MB					[121]
Environment						
- Environmental Monitoring				15 s (freq10min)		[122]
Industrial						
- Factory Automation		10-300	10^{-9}	0.25 - 10 ms		[123]
Transportation						
- Road Safety Urban		$<$ 500	$10^{-3} - 10^{-5}$	10-100 ms		[123]
- Road Safety Highway		$<$ 500	$10^{-3} - 10^{-5}$	10-100 ms		[123]
- Urban Intersection		1M/car	10^{-5}	$<$ 100 ms		[123]
- Traffic Efficiency		1k	10^{-3}	$<$ 100 ms		[123]
- Intersection Management				500 ms		[124]
- Traffic Light Management				$<$ 5 s		[122]
- Vehicle Tracking				$<$ 10 s		[122]
- Quality Shipment Tracking				$<$ 15 s		[122]
- Road Condition Tracking				$<$ 30 s		[122]

In the level of a network, QoE objective models should include network quality, network delay, network parameters and information, network constraints and data rate. Network quality, e.g., jitter, loss, impairment indicators, are used for QoE models in the following works : [6], [50], [54], [55], [61], [62], [63], [64], [77], [78], [84], [125], [126]. In addition, network delay is used for objectively estimating QoE in [61], [62], [63], [77]. Some works extended their QoE models to include network states e.g., parameters, statistics related to network utilization or even network packets themselves. Example of these works are [52], [77], [83], [90]. Constraints on networks, e.g., capacity, bandwidth, path constraints, can be taken into QoE model creation such as in [91]

Additional important metrics for including into QoE model are system-related as shown in Table 7. QoE can be modeled from multiple system-related states and performance indicators e.g., energy consumption [79], [80], [81], [127], [128], [129], [130], [131], service access rate [82], resource usage or resource gain [82], [83], [125], [132], processing time or completion time [80], [82], [125], hardware outage [131], storage cost [133], caching capacity [133], data sharing gain [79], task drop loss [79], task success rate [129], transmission time (downlink and backhaul) [134], and transmission rate [87].

Nevertheless, some metrics defined in many prior works cross over multiple parts in the system e.g., time metric like latency. Latency is incorporated into QoE models in many works; however, its definition sometimes includes both computational and communication time. It corresponds to both system and network metrics. Prior works that incorporate latency into QoE models can be listed as the following : computation and communication latency [127], application latency [128], overall latency [79], service latency [129], [130], application latency [81] and latency (SINR) in data transmission [133].

Actual experiences are reflected from users via QoE estimation. However, it is highly associated with contents of applications too such as video, image, voice, game and others. Prior works focused on modeling QoE mostly in video applications because of their readiness in QoE subjective measurement. Video-related metrics can be buffering time, video quality switches, start-up delay, rebuffering, resolution, frame rate, player state, buffer ratio or chunk size. These metrics are utilized by QoE models of the following works : [50], [57], [58], [64], [68], [78], [85], [86], [87], [89], [134]. Also, some works such as [89], [135] include standard video parametric or objective indicators (SSIM, VQM, PSNR) into their QoE models. For image-related applications, image sizes can be used into QoE model too, as shown in [84]. For game-related contents, game experience loss is considered for direct user experience to the content in [136].

It is obvious that user-related metrics are still not often included into prior QoE models. User-relate metrics can be user tendency, profile, users' device status, class-based metrics and others. Mushtaq *et al.* [50] and Song *et al.* [49] included user profiles into their QoE model. User engagement and perception metrics are applied in [6], [51], [52], [53], [54], [55], [131]. Incorporating social context factor into QoE was studied by Laiche *et al.* [51]. Context information related to user was used in the QoE model by Ting *et al.* [53], and Hong and Kim [128].

In conclusion, QoE metrics can be modeled from various QoE cause factors : user (user and context), application (content), service, network and physical (system) metrics. The summary of prior works is shown in Table 7. It can be seen that most metrics used in the literature are network quality and energy consumption. Still, user-related metrics are used by a small number of works. Content-related metrics are mostly derived from video applications. Due to the lack of QoE models in other applications e.g., IoT, images, it is an opportunity for researchers to invent new QoE models.

VI. QoS/QoE AND RESOURCE MANAGEMENT

For IoT systems, QoS/QoE-aware recent problems in resource management are discussed in this section. Since IoT systems consist of heterogeneous computing resources scattering around in Mobile devices, MEC, Fog and Cloud. Assignment of computational tasks and allocation of computing and memory of resources are indispensable to achieve QoS/QoE criteria. Problems of resource management in IoT often fall into the following categories : computational offloading problems, application or service placement problems, data caching problems. Recent QoS/QoE-related works will be reviewed in subsequent sections.

The critical aspects of QoE-aware resource management are presented in this section. Unlike most prior works, they are not mainly considered QoE-aware resource management in various problems. In this work, we focus on how the current research in the literature tackles the QoE-related issues with recent ML approaches for emerging IoT systems such as Edge/Fog/Cloud as follows.

- QoE-aware computational offloading
- QoE-aware resource placement
- QoE-aware data caching

A. QoS/QoE AND COMPUTATIONAL OFFLOADING

Computational offloading is a problem of determining the best node, which is commonly located in Edge, for moving tasks from Cloud nodes to these Edge nodes. It is a crucial procedure for improving user experiences as well as QoS/QoE metrics. By offloading tasks from Cloud to MEC, reduction of latency will be significantly lower; thus, users experience faster response time. Offloading problems often relate to determine the best location to move computational workloads so that overall latency of services is minimized.

The problems are extensively investigated in prior literature e.g., [10], [11], [12]. All of them are comprehensive survey papers related to offloading problems. According to [10], the authors studied the classification of offloading problem types based on offloading flows in Edge architectures, computational model (e.g., channel model, computing model, communication model), offloading methodologies (e.g., optimization, MDP, game theory, ML-based methods). Most prior objectives are related to minimization in latency, energy consumption, task dropping, cost and maximization in computational rate and efficiency. All of them are system-related factors, not in QoE perspective.

Similarly, a survey by Islam *et al.* [11] focused on offloading problems at Mobile Edge computing. The authors classified the problems into three strategic models : computational model, decision-making model, and algorithmic methodologies. Prior works in the paper are concerned with minimizing latency, minimizing energy consumption at end devices, optimizing computational costs or revenues and minimizing task failures. Only QoS metrics are taken into optimizing offloading, but QoE is rarely mentioned in the prior works.

Lastly, offloading problems at MEC with mobility-awareness are elaborated in [12]. According to [12], most objectives are related to latency, energy efficiency and execution time. Unlike others, mobility models are taken into consideration in this survey. Also, the authors extracted features of existing offloading schemes such as energy efficiency, latency reduction, execution time, channel states, distances between Edge and User devices, and offloading failure reduction. Still, QoE is rarely seen in that review.

Studies in [10], [11] and [12] demonstrated how offloading problems could be solved by various methodologies. A lot of works attempted to propose offloading policies based on classic optimization techniques. Recent works focused on utilization of ML-based algorithms to offload computational tasks from Cloud to Edge or from Mobile devices to Edge. It is investigated by [13], which classified all prior works into three main ML approaches : supervised learning, unsupervised learning and reinforcement learning. Optimized metrics are still related to only delay, energy, QoS, response time, cost and others, but QoE is not addressed.

For these reasons, QoE-aware offloading problems are investigated further in this paper. It is obvious that most of prior offloading goals are related to QoS factors rather than QoE. Thus, real user experiences are not captured by existing offloading policies. Recent QoE-related offloading schemes are listed in [79], [80], [127], [128], [129], [130], [138], [139], [140], [141]. The summary of these works is shown in Table 8.

Let us consider QoE-aware offloading problems, which are formulated by classic optimization techniques. According to [127], QoE and trade-off between latency and energy consumption at Edge are considered a computation offloading scheduling problem. The authors defined QoE factors by using latency (both computation and communication) and energy consumption at Edge and Mobile devices. The system attempted to allocate tasks on Edge computing servers so that these QoE factors were optimized. The authors formulated the problem as Mixed Integer NLP, which was solved by

TABLE 7. QoE metrics.

Paper	QoE Model	Problem Type	QoE Cause Factors (QCF)	Tier	Year
[50]	network quality + video parameters + user profiles Subjective test for MOS correlation with network quality + video parameters + user profiles	prediction	network + content + user	Cloud	2012
[89]	video-related metrics (SSIM, VQM, Frame rate, Data rata, Resolution)	prediction	content	SDN networks	2017
[61]	delay + network quality (RTT + jitter + BW) + video-related metrics (SSIM, VQM, PSNR)	prediction	network + content	SDN networks	2017
[62]	delay + network quality (jitter)	prediction	network	-	2010
[63]	delay + network quality (jitter, packet loss)	prediction	network	-	2015
[77]	delay + network quality (jitter, packet loss) + network information (stream duration + incoming/outgoing traffic)	prediction	network	SDN networks	2018
[51]	user engagement metrics + social context factors	prediction	user + context	SDN networks	2021
[91]	network parameters (network capacity, bandwidth, path constraints)	prediction	network	MIoT	2018
[90]	network information (from packets)	prediction	network	SDN networks/Edge	2018
[52]	4 subjective scores and 89 network parameters	prediction	user + network	Mobile networks	2019
[53]	user + system + context IFs	prediction	user + system + context	-	2020
[58]	video-related metrics (buffering time, quality switches)	prediction + ABR control	content	Mobile networks	2020
[68]	video-related metrics (start-up delay, rebuffering, resolution)	prediction	content	-	2020
[57]	video-related metrics (video chunkwise QoE)	ABR control	content	Mobile networks	2018
[78]	network quality + video-related metrics (buffer size, playback delay, rebuffering, video quality)	ABR control	network + content	Mobile networks	2020
[64]	network quality + video-related metrics (player state)	network control	network + content	Access networks	2019
[7]	network condition + user perception (QoS-QoE correlation model)	network control	network + user	SDN networks	2019
[66]	video-related metrics (start-up latency, buffer ratio/time)	service control	content	Edge	2020
[54]	network cost + user IFs (predicted MOS)	service control	content	-	2020
[6]	MOS (network quality + user perception) (subjective test for MOS prediction by SVM)	power control	network + user	Access networks	2020
[127]	(computation + communication) latency + energy	offloading	network + system	Edge	2019
[128]	(application) latency + energy + user context (tendency + battery)	offloading	network + system + user	Edge	2016
[80]	energy + completion time	offloading	system	IoT nodes + Fog	2021
[79]	(overall) latency + energy + task drop loss + data sharing gain	offloading	network + system + user	Edge	2019
[129]	(service) latency + energy + task success rate	offloading	network + system + user	Edge	2020
[130]	(service) latency (constraints) + (restricted) energy	offloading	network + system	Edge + IVs	2020
[82]	user expectation (service access rate + required resources + expected data processing time) and resource ratings (Round Trip Time, Resource Availability, Processing Speed)	placement	system + user	Fog	2019
[81]	MD-QoE model : runtime context (location, network strength, battery level) + content usage (frequency of usage, overall ratings, uninstall ratings, time spent on application, never-been-used ratings) + user expectation (accuracy, resource usage efficiency, response time and preferences)	placement	network + system + user	Fog	2020
[125]	user expectation (application ratings = expected access rate + required resource + processing time, and resource ratings = RTT + Resource Availability + Processing Speed.	placement	network + system + user	Fog	2020
[132]	resource gain (QoS-QoE correlation model with sigmoid function)	placement	system	Edge	2020
[84]	logarithmic objective function of photo fetching time of network QoS (link capacity, RTT, packet loss) , dependent on network quality + photo sizes + placement location	placement	network + content + context	Edge	2016
[126]	network impairment (jitter, loss, G-Model)* (MOS objective)	placement	network	Fog	2020
[136]	gaming experience loss	placement	user + content	Cloud	2020
[83]	resource usage + network usage (ITU P.1203 standard model)	placement	network + system	Cloud	2021
[131]	hardware outages + energy + colocation interference (UE metrics)	placement	system + user	Cloud	2020
[134]	transmission delay (downlink + backhaul) + video-related metrics (MOS objective)	caching	system + content	MEC	2020
[55]	network quality + user perception (long term QoE) (QoS-QoE correlation logarithmic model)	caching	network + user	Access Network	2019
[85]	video-related metrics (distortion from insufficient bit rate) (QoS-QoE correlation logarithmic model [137])	caching	content	MEC	2020
[133]	latency (SINR) + storage cost + caching capacity	caching	network + system + context	Edge	2019
[49]	class-based user interest model	caching	user	Access Network	2021
[135]	video quality level (abstract)	caching	system + content	MEC	2019
[87]	data rate + video-related parameters (QoS-QoE correlation model)	caching	system + content	MEC	2020
[86]	video-related metrics	caching	content	MEC	2020

Reformulated Linearized Technique (RLT) and branch and bound method. It is shown that overall QoE, energy consumption and latency performance are improved. Unlike others, Hong and Kim [128] studied computational offloading problems from mobile to cloud to optimize trade-off between energy consumption and application latency. In addition, user context such as user tendency and battery information was included into the problem. Dynamic Programming (DP) was used to maximize QoE, which was calculated by parameters on three domains : context, human and technological domains. Battery level, application characteristics, amount of offloaded data are parameters in context domain. User tendency and per-slot energy consumption and latency were used in human and technological domain respectively.

Moreover, the problem of offloading task from IoT devices to Fog nodes was considered for optimizing the trade-off between energy consumption and task completion time by [80]. The authors' objective was to minimize the overhead of energy and completion time in task offloading processes. They assumed that the weighted sum of energy consumption and task completion time was perceived as QoE by users. A heuristic algorithm was proposed and based on the Genertic Algorithm and Particle Swarm Optimization for solving mixed integer nonlinear programming. In [138], Li *et al.* proposed Queec, a QoE-Aware Task offloading algorithm, to offload computing tasks from resource contraint nodes to Edge. Their algorithm was designed by using Mixed Integer Linear Programming (MILP). Also, as shown in [139], Pham *et al.* applied Branch and Bound techniques to determine the offloading algorithm maximizing the QoE utility function. Greedy-based heuristic algorithm is used in [140] to improve the QoE by minimizing the overall completion time of the system.

Recently, emerging ML approaches have been applied on various works as well as offloading problems. We consider QoE-related offloading policies, which are based on recent ML techniques. In [79], a RL-based computational offloading algorithm was proposed to select proper MEC devices, so the quality of experiences in terms of computational delay, energy consumption and task drop loss were optimized. This computational offloading technique was used with IoT wireless devices, which had Energy Harvest (EH) functions. The authors attempted to optimize both performance and energy consumption. To accelerate convergence time, Fast Deep Q-Network was utilized together with hotbooting Q-Learning algorithm. Their simulation results showed that all quality parameters are improved In [129], Lu *et al.* created a novel QoE model incorporating the following metrics : service latency, energy consumption and task success rate at Edge Computing nodes. Deep Reinforcement Learning was utilized for offloading computational workload to these Edge nodes, such that user satisfaction is higher. In their works, Double Q-Learning with Dueling Network was used instead of typical critic network in Actor-Critic deep network. For faster algorithm convergence, the authors proposed Double-dueling-deterministic policy gradient instead of deep deterministic policy gradient together with this new QoE model. Their results showed an improvement in QoE calculation. In [130], QoE-aware task offloading from Intelligent Vehicles (IVs) to Edge nodes was investigated. The authors assumed that IVs had limited caching spaces and computing capabilities, and Edge nodes also had more caching resources that could offloading from IVs. A new objective QoE model was proposed by using restricted energy consumption at IVs with service latency constraints. An improved DRL-based algorithm, called RA-DDPG, was proposed to seek for an optimal offloading solution saving more power. This DRL-based algorithm was modified with new techniques e.g., Prioritized Experience Replay (PER) and Stochastic Weight Averaging (SWA). In [141], Dai *et al.* proposed DRL-based a new offloading method from vehicles to Base Station with UAV assisted Edge nodes, so the QoE function (weighted sum of MOS) was maximized in IoV networks.

QoE-aware offloading policies are still an unaddressed area in the current literature. Many prior works focused on how to minimize latency and energy consumption by exploiting offloading schemes. Only a few existing works incorporate QoE into their offloading policies. Still, current works lack concrete QoE-related evidence because of time consuming processes in QoE measurement.

B. QoS/QoE AND PLACEMENT POLICIES

Placement problems of applications, containers or services over resources are widely investigated in IoT system. Many works attempted to propose placement policies such that an allocation of computational services over heterogeneous resources results in the best outcome. Various types of placement problems have been studied e.g., application placement, [81], [82], [125], container placement, [83], virtual machine placement, [131], [136], user placement, [132], service instance placement, [84], [126], [142].

Placement policy helps users receiving faster service response and better quality (e.g., video) because services are placed and run on either nearby or appropriate-processing power nodes. Thus, these policies result in higher user experiences or QoE. In addition, these placement policies are commonly deployed on Edge nodes [84], [132], [143], or Fog nodes [81], [82], [125], [126], [142], or Cloud infrastructure [83], [131], [136].

Still, most of the prior works considered only QoS for placement objectives. As shown in [14], Salaht *et al.* provided a comprehensive survey for service placements in Fog and Edge Computing. Prior placement policies focused on latency, resource utilization, cost and energy consumption as main objective functions. Only a few works considered incorporating QoE into their placement problems.

Moreover, as demonstrated in [14], many works proposed placement policies by using common optimization techniques e.g., Integer Programming, Constrainted optimization and others. Not many works focused on using Machine Learning techniques in placement problems. Hence, in this paper, we will focus only on QoE-related works, and some

TABLE 8. QoE-aware computational offloading.

Paper	Environment	Offloading strategies	QoE model	optimization objective	Approach	Simulator	Tier	Year
[127]	offloading tasks in IoT wireless networks	offloading scheduling and resource allocation on Edge with trade-off between latency and energy	latency (computation + communication) + energy consumption of Edge and Mobile Devices	minimize QoE-driven cost function (weighted sum of total latency delay + energy consumption)	Mixed Integer NLP solved with RLT-based branch and bound method	Numerical studies	Edge	2019
[128]	offloading tasks from MEC to Cloud	transmission scheduling for offloading from Mobile devices to Cloud	latency + energy + user context (e.g., tendency + battery)	minimize QoE-aware cost function (expectation of total cost)	Dynamic Programming (DP)	Numerical studies	Edge	2016
[80]	offloading tasks from IoT devices to Fog nodes	algorithm for offloading decisions, choosing Fog nodes and allocating computational resources to optimize trade-off between energy and task completion time	energy consumption + task completion time (task offloading overhead)	minimize QoE cost function (weighted sum of energy consumption and task completion time)	Heuristic algorithm by using Mixed Integer NLP + Genetic (GA) and Particle Swarm Opt (PSO) approaches	Numerical studies	IoT nodes and Fog	2021
[138]	offloading tasks from IoT devices to nearby Edge nodes in IoT networks	offloading tasks to Multiple Edge nodes under the QoE constraints to minimize the time used for offloading	QoE vector (execution time, accuracy, error, frame rate, resolution)	minimize the overall offloading latency s.t. QoE constraints	MILP	Real Testbed	MEC	2021
[139]	offloading tasks from mobile devices to MEC nodes	offloading tasks in MEC with mobile resource assistance from vehicles.	mapping latency with utility function	maximize QoE	Branch and Bound	Simulation	MEC	2021
[140]	offloading tasks from IoT devices to Edge in MEC	offloading tasks to improve the QoE completion time and success rate in IoT networks	completion time and success rate of tasks	maximize the completion time	Greedy algorithm	Simulation (Edge-CloudSim)	MEC	2021
[79]	offloading from Mobile Devices to MEC IoT wirless devices + Energy Harvest (EH) for better QoE	offloading data from IoT wireless nodes to MEC by controlling offload rate based on current battery level, prior radio bandwidth and estimated harvested energy.	Quality function of overall delay, energy consumption, task drop loss, data sharing gain	maximize utility of IoT devices	RL + Fast Deep Q-Network	Simulation	Edge	2019
[129]	offloading data from multiple users to Edge	offloading policy for improving QoE based on a new QoE model	service latency + energy consumption + task success rate	optimize QoE function	DRL + Double Q-Learning and Dueling Networks	N/A	Edge	2020
[130]	offloading tasks from Intelligent Vehicles (IVs) to Edge nodes in IoV	offloading data from computational and cache limited IVs to Edge nodes for optimizing QoE satisfaction in perspectives of energy and latency	(restricted) energy consumption and service latency (constraints)	maximize QoE satisfaction	Improved DRL (RA-DDPG) with prioritized experience replay (PER) and stochastic weight averaging (SWA)	Simulation (Pytorch)	Edge + IVs	2020
[141]	offloading tasks from vehicles in IoV networks to assisted UAV Edge nodes	offloading tasks from vehicles to BS and UAV assisted Edge nodes when resources at BS is insufficient	weight sum of MOS function of delay and tx rate	maximize QoE	DRL	Simulation (SUMO) with TensorFlow/Python	MEC	2022

ML-related placement policies because QoS-related and non-ML-related works are extensively studied in the past.

Let us consider QoS-aware placement algorithms that are ML-based policies such as [142], [143], and a comprehensive survey in [14]. In [142], a RL-based placement algorithm was proposed to maximize the value-based utility function for Fog service placement. This RL-based algorithm together with DQN is called FogReinforce. According to [143], Zhang *et al.* considered Edge server placement problems at Mobile Edge Computing. Their proposed placement policy was based on enhanced DDPG algorithm from DRL technique with Convolutional LTSM networks. It attempted to maximize profit for connected vehicles and total cost of reserving MEC server and connection refuses. They exploited the spatio-temporal correlation between vehicle and traffic with Convolution LTSM networks in their design. Nevertheless, there is a recent survey investigating AI-related placement policies on Fog Computing. Similar to [14], most objectives were related to time, cost, network, resource utilization and energy. QoE was explicitly taken into account for only a few studies. Nayeri *et al.* [15] showed that both classic ML and Deep Learning techniques were used to find solutions for Fog service placements.

It is shown that even though placement problems are extensively studied, QoE is not incorporated into prior objective functions. Therefore, real user experiences are not reflected at all. Next, recent works that used QoE into their optimization purposes will be discussed. The QoE-related placement policies are summarized in Table 9. Common optimization approaches have been applied for determining QoE-aware placement policies into the following papers : [81], [82], [84], [125], [126], [132] and [136]. On the other hand, QoE-aware ML-based algorithms have recently been proposed in [83], [131], [144].

To estimate QoE in real-time, the following works consider for QoE calculation by using objective methods. In [82], QoE was based on User IFs or User Expectation (UE) metrics e.g., service access rate, required resources, expected processing time. The QoE metric in [81] is Multidimensional-QoE model : Runtime Context (location, network strength, battery level), Application usage factors (frequency of usage, overall ratings, time spent on application and others), and User

Expectation (accuracy, resource usage efficiency, response time and preferences). Baranwal *et al.* [125] formulated QoE by application ratings from expected access rate, required resource, processing time, and resource ratings from Round Trip Time, Resource Availability, Processing Speed. According to [132], QoE metric was estimated by using QoS-QoE correlation model, or the Sigmoid function mapping QoS to QoE for each user. QoE metric in [84] was modeled from photo sizes, network QoS and placement location as input parameters. MOS objective QoE was estimated from network impairment parameters (e.g., jitter and loss or G-Model) by Tsipis *et al.* [126]. Also, QoE estimation can be application-related e.g., gaming experience losses in [136]. In [83], Carvalho and Macedo utilized QoE based on the standard such as ITU P.1203, then they predicted the QoE by collecting CPU, Memory, Disk, Network information. Lastly, UE-related metric are defined by using the impact of hardware outages, the power required by Data Center and performance perceived by users as shown in [131].

Now let us consider QoE-aware placement policies that were proposed by recent literature in details. In [82], Mahmud *et al.* proposed a QoE-aware placement policy based on Fuzzy logic approach for incoming application placement requests in Fog Computing (FC). First, their policy will prioritize application placement requests based on User Expectation metrics, Rate of Expectation (service access rate, required resources and expected data processing time), by using Fuzzy method. FC instances will be classified by Status metric, or Capacity Class Score, e.g., Round Trip Time, Resource Availability, Processing Speed. Then, the policy can map the prioritized placement requests onto potential FC instances based on UE and Status metrics. A two-phase QoE-aware placement algorithm in FC environments was studied by Nashaat *et al.* [81]. They also proposed a multi-dimensional QoE model for prioritizing application placement requests. In the first phase, the algorithm prioritizes new placement requests by runtime context, application usage e.g., frequency of usage, overall ratings, uninstall ratings, time spent on application, never-been-used ratings, User Expectation metrics e.g., accuracy, resource usage efficiency, response time and preferences. Next, it maps and places the application placement requests on FC instances by considering resource status such as proximity, computing capabilities, expected response time. A light-weight QoE-aware and TOPSIS-based placement policy was proposed for application placement in FC in [125]. The modified TOPSIS has less complexity than common methods solving optimization problems. Their strategy has two parts : prioritizing applications based on user experiences and fog instances based on computing resources respectively. Their QoE model was based on Rating of Application RoA (expected access rate, required resource, processing time) used when prioritizing applications, and Rating of Fog instances RoF (Round Trip Time, Resource Availability, Processing Speed) used when prioritizing Fog instances. Lai *et al.* [132] considered the problem related to user allocation on shared Edge servers with finite resources to maximize QoE. A heuristic algorithm was proposed to maximize QoE by allocating users over distributed Edge servers in Mobile networks. QoE estimation for photo-related service placement on Edge was investigated by Dinh-Xuan *et al.* [84]. Their QoE model was proposed to determine the relationship between photo sizes and loading time. It was modeled by using the logarithmic objective QoE model as a function of photo fetching time (estimated by network QoS : link capacity, RTT, packet loss). The authors suggested that QoE of service placement could be improved by choosing appropriate photo size, controlling network QoS and relocating Edge closer to users. According to [126], a QoE-Aware Rendering Service Allocation (QoERSA) was proposed to optimize a delay-sensitive placement problem at Fog computing. It is used to improve QoE for rendering game service applications and to avoid offloading computational tasks to Cloud, which results in higher delay. A distributed algorithm for optimizing QoE in Virtual Machine Placment problem in Cloud was proposed in [136]. The game theoretic approach in resource competition was used to design a distributed algorithm when QoE was considered. They focused only on VM placement on Cloud side. The mutual satisfaction among different users appears to converge at a certain point by using this algorithm.

Still, most recent works attempted to exploit ML methods for solving QoE-aware placement problems. Example of these works are [83], [131], and [144]. In [83], a container placement problem in Cloud environment was investigated by Carvalho and Macedo. They proposed a Kubernetes scheduler extenstion and a resource scheduling policy to maximize QoE metric, which is defined according to ITU P.1203 standard. They exploited system information such as CPU, memory, disk and network for QoE prediction. Their prediction approach is based on a supervise learning method with regression while LSTM or RNN is used as Neural Network for predicting QoE. The result showed a significant improvement in QoE. A DRL-based placement policy was proposed for selecting server to deploy Virtual Machines in [131]. Caviglione *et al.* formulated a multi-objective optimization by attempting to reduce the impact of hardware outage, power usage and UE-related performance. The placement mechanism was implemented on the Cloud side. According to [144], a new DRL-based algorithm for allocating tasks and resources in MEC networks is proposed to maximize QoE function.

In brief, QoE-aware placement policies in prior studies are elaborated in this section. Even though a large number of placement problems were proposed, only a few among them included QoE metrics into their consideration. In contrast to QoS, a good QoE model can reflect to real user experiences. Most recent works have focused on optimizing QoE-aware placement on resources in Cloud/Fog/Edge by using emerging Deep Learning techniques.

TABLE 9. QoS/QoE and placement policies.

Paper	Environment	Placement Strategy	QoS/QoE Model	QoS/ QoE	ML	Optimization objective	Approaches	Simulator	Tier	Year
[82]	application placement on Fog computing instances	Determine optimal placement of application requests on Fog instances to maximize QoE gain	Rate of Expectation or RoE (service access rate + required resources + expected data processing time) and Capacity Class Score or CCS (Round Trip Time, Resource Availability, Processing Speed)	QoE	No	maximize Rating Gain (RoE * CCS)	Fuzzy logic based method for prioritizing application placement requests by User Expectation metric (RoE) and for classifying Fog instances based on system states (CCS)	iFogSim	Fog	2019
[81]	application placement in Fog Computing	Propose application placement policiy by using Multi-Dimensional QoE (MD-QoE) model onto Fog instances	MD-QoE model : Runtime Context (location, network strength, battery level), Application usage factors (frequency of usage, overall ratings, uninstall ratings, time spent on application, never-been-used ratings) User Expectation (accuracy, resource usage efficiency, response time and preferences)	QoE	No	maximize Quality Rating Gain	Two-step algorithm : (1) prioritizing application requests by runtime, application usage and User Expectations, (2) mapping and placing application on Fog instances based on resource metrics e.g., proximity, computing capabilities, expected response time.	iFogSim	Fog	2020
[125]	application placement in Fog Computing	Propose a light-weighted QoE-aware modified TOPSIS as application placement policy on Fog instances	Rating of Application RoA (expected access rate, required resource, processing time), Rating of Fog instances RoF (Round Trip Time, Resource Availability, Processing Speed)	QoE	No	maximize QoE	Modified TOPSIS : (1) prioritizing application requests by using User Expectaion metrics (2) classifying Fog instances based on computational resources (3) mapping prioritized application onto classified Fog instances by using Modified-TOPSIS method	Simulation	Fog	2020
[132]	placement of user requests in Edge Computing	Propose a method for allocating user requests over shared and resource contraints Edge nodes to maximize overall user expectation	Objective model (QoS-QoE correlation model) with normalization of amount of assigned resources for each user	QoE	No	maximize total QoE of all users	ILP + heuristic	Numerical Studies (CPLEX)	Edge	2020
[84]	Photo-related content placement in Edge Computing	show studies of placement strategies when trade-off between photo size and placement location is considered for improving QoE	logarithmic objective QoE model as a function of photo fetching time (estimated by network QoS : link capacity, RTT, packet loss)	QoE	No	improve this objective model	experiments on QoE improvement method based on resizing photo, relocating photo contents and leveling up network QoS.	Testbed + NetEm	Edge	2016
[126]	Game service placement in Fog Computing	Propose an optimal delay-sensitive placement policy, QoERSA, for rendering game services through Fog renderer and avoiding offloading tasks to only Cloud services	objective model, G-Model, by network impairment parameters : jitter and packet loss	QoE	No	minimize overall cost of communication cost and service deployment cost	distributed approach of FLP (Facility Location Problem)	OMNet++	Fog	2020
[136]	Virtual Machine placement in Cloud Computing	consider VM placement problem for Cloud gaming (high video quality + low delay)	Objective model : Gaming experiences loss	QoE	No	minimize gaming experience loss (QoE)	Distributed algorithm to optimize VM placement by Game theory : resource competition approach	Simulation	Cloud	2020
[142]	Fog service placement on Fog nodes	Propose online RL-based algorithm for Fog service allocation to maximize VoI-based objective	Value of Information (VoI) function	QoS	YES	maximize total VoI utility function to all users	RL + DQN = FogReinforce	Real Testbed	Fog	2021
[143]	Edge server assignment problems for Connected Vehicles (CV) on Edge Computing	Propose a DRL-based algorithm with DNN for learning spatio-temporal correlation between vehicle and traffic	profit gain, cost of reserving servers, cost of refused connection	QoS	YES	maximize profit and cost for reserving MEC server and penalty for denied connections	DDPG algorithm from DRL with two major enhancements + Conv. LTSM networks	Simulation (Pytorch)	MEC	2021
[145]	container placement at Fog	Propose a placement policy for allocating containers in Fog environments	number of allocation success + delay and fog node constraint	QoS	YES	maximize number of satisfied users	DRL	Simulation	Fog	2019
[83]	Container placement in Cloud Computing	propose an extension for Kubernetes resource scheduler/rescheduler for optimizing QoE under Service Level Objective (SLO)	ITU P.1203 standard and predict QoE by using CPU, Mem, Disk, Network information	QoE	YES	maximize QoE	ML-based QoE predictor (supervised learning regression) with NN as (RNN or LSTM)	Simulation (Keras)	Cloud	2021
[131]	Virtual Machine placement in Cloud computing	placement mechanism for selecting server to deploy VMs with multi-objective approach	hardware outages, colocation interference, energy consumption	QoE	YES	minimize effect of HW outage, colocation interference and energy consumption	DRL	Simulation (Python)	Cloud	2020
[144]	Task and resource allocation in MEC	allocating tasks to maximize QoE by using ML approach at MEC networks	required QoS metrics and resources	QoE	YES	maximize QoE	DRL + DNN	Simulation	MEC	2021

C. QoS/QoE AND DATA CACHING

Data caching techniques are used to place frequent access contents, services or applications on computing nodes close to users. Normally, these nodes are close to users or places where services are actually operated. Thus, access time to services or contents are significantly reduced by optimizing caching policies. Recent caching policies were designed to reduce latency and decrease traffic in Edge Computing [133], [134], [146], [147], Mobile Edge Computing (MEC) [85], [86], [87], [135], [148], [149], Fog Radio Access Networks (RANs) [150], [151], [152] and other Access Networks [49], [55], [153].

Since Data caching helps the system reducing latency between IoT devices and Edge/Fog, or Edge/Fog and Cloud, minimizing content retrieval time and maximizing Caching Hit Rate (CHR) will result in significant improvement of QoS/QoE due to much lower access time. However, many works proposed new caching policies with only QoS consideration. Only some works studied how to incorporate a real QoE into the problems.

In QoS-related caching policies, some were designed to optimize revenues [146], network utility and backhaul cost [153], content retrieval time [147], [148], [149], cache hit rate [152], transmission delay [150], and end-to-end latency [151]. Most prior works focused on minimizing content retrieval time and maximizing cache hit rate. These QoS-related caching policies are summarized in Table 10.

Nevertheless, recent caching policies were proposed to incorporate QoE model (user-related or service-related metrics) into their objectives rather than only QoS metrics. For instance, the following works such as [86], [87], [134], [135] proposed to incorporate application-related (video-related) factors into their objective. Specifically, video quality level or video-related parameters were used in their QoE calculation. Some others included user-related factors into the QoE model e.g., [49], [55]. It is well-known that the relationship between QoS and QoE are sometimes used for predicting QoE score. In [55], [85], logarithmic QoE-QoS correlation models [137] were used for estimating QoE values. Also, Chou *et al.* [87] used both QoS-QoE correlation function between accumulated data rate and video-related metrics as their QoE. A prior work e.g., [133] included caching-related parameters into their QoE, which was modeled from storage cost, transmission latency and caching capacity. These QoE-related caching policies are summarized in Table 10.

Many QoS-related caching policies were formulated as optimization problems, and solved by classic optimization techniques e.g., [146], [153]. In [146], Liu *et al.* considered for data caching problem at Edge Computing. They proposed a heuristic algorithm to maximize data caching revenues, which was calculated from profit and cost of data caching subjected to access latency constraints. The formulation was in Integer Programming (IP) form, and a suboptimal algorithm was presented and experimented on real dataset. Joint optimization between load balancing and backhaul saving in [153] was proposed in cache management at Wireless Access Networks. Dai and Yu attempted to maximize backhaul-aware proportional fairness network utility by exploiting both caching placement and association of users to Base Stations. Their objective was a weighted function of network utility and backhaul saving costs. They proposed an iterative algorithm iterating between user association and content placement by using IP formulation. More QoS-related caching policies with classic optimization techniques have been extensively investigated in the literature.

Recently, new caching policies were proposed by exploiting Machine Learning techniques e.g., [147], [148], [149], [150], [151], [152]. Reinforcement Learning (RL) based algorithms were presented in [147], [152]. Deep Reinforcement Learning (DRL) based algorithm was demonstrated in [148], [149], [151] with Double Dueling DQN, [150] with Dueling DQN. In [152], Lu *et al.* proposed a RL-based algorithm for minimizing cache hit loss rate in distributed Edge caching of Fog RANs. Caching contents at Edge nodes such that average transmission delay and Cache Hit Ratio were minimized in Fog RANs was investigated by Xu *et al.* [147]. They proposed a RL-based algorithm to jointly minimize traffic load and retrieval delay cost. A new caching policy was used by Guo *et al.* [150] to minimize average transmission delay and improve Cache Hit Ratio in Fog RANs. The algorithm is based on the RL framework with Dueling DQN. Joint optimization between caching policy, offloading policy and radio resource allocation in Fog RANs was studied in [151]. A new DRL-based caching policy was proposed to minimize average end-to-end delay by using actor-critic DRL with DNN. According to [149], a multi-agent DRL-based algorithm was proposed to minimize content access latency and traffic cost for Content Delivery Networks (CDNs) at Edge. A novel caching policy for minimizing fetching delay and maximizing CHR was designed by using distributed Double Dueling Q-Network (D3QN) and DRL algorithm in [148]. The authors considered for caching microservices at Edge to minimize fetching delay and maximize hit ratio. The problem was formulated by using cache node selection and microservice replacement as Markov Decision Process (MDP). Then, the Distributed Double Dueling Q-Network (D3QN) based algorithm in the DRL approach was proposed to improve fetching time, which was shown in their experiments.

More ML-related caching policies can be found in [16], whose the authors provide a comprehensive survey of caching problems on Edge. The authors provide Supervised Learning, Unsupervised Learning, Reinforcement Learning, Deep Learning and other techniques on their survey. However, most of their objective caching problems are Cache Hit Rate. Only a few prior works concerns over impact factors from QoE.

Previously, QoE is not really taken into account for implementing caching policies on Fog/Edge nodes. Still, recent works took QoE Cause Factors into consideration when designing data caching policies. For instance, QoE-aware caching policies were presented in [49], [55], [85], [86], [87],

[133], [134] and [154]. According to [55], [85], [134], [155], the authors used common optimization techniques for solving caching problems. However, most recent works, e.g., [49], [86], [87], [133] and [154], attempted to deal with these problem by using emerging ML-based algorithms such as Deep Learning.

As discussed, some QoE-aware caching policies were designed by using optimization methods such as [55], [85], [134]. In [134], Wang *et al.* considered UAV deployment and caching placement at Edge networks, which allowed data caching for UAVs. They attempted to maximize QoE by optimizing both UAV deployment and caching placement. The QoE is objective MOS, which is calculated from downlink and backhaul transmission delay with inspired video objective function. A new caching policy was implemented in [55] for pre-caching most frequent used videos for streaming services in Cloud RANs. The problem was divided into two stages : caching stage and delivery stage. Long-term transmission-aware caching problem was formulated in caching stage, but short-term transmission problem was used in delivery stage. A new iterative algorithm was proposed to maximize the weighted sum of users'QoE, which was approximated from long-term estimated QoE. This estimated QoE was based on QoE-QoS correlation Logarithmic model. According to [85], a Dual Lagarian-based caching policy was designed to optimize caching placement and user association of live video streaming services at MEC. Their QoE was based on the QoE-QoS correlation Logarithmic model [137]. The proposed algorithm was used to maximize the weighted sum of this objective QoE in a joint caching placement, video quality decision and user association problem. A heuristic data caching method was proposed in [155] to maximize the overall QoE which was modeled from QoS metrics, data storage capacity and caching cost. The algorithm was designed from Genetic algorithm.

In addition to these works, most recent works attempted to apply Machine Learning approaches for solving QoE-aware data caching problem at MEC. They can be listed in [49], [86], [87], [133] and [154]. In [133], DRL-based data caching at Edge Computing was proposed to reduce storage cost and transmission latency. The algorithm attempted to maximize QoE, which is modeled by three major IFs : storage cost, transmission latency and cache capacity. Their approach was based on Reinforcement Learning with a DNN approximator for estimating Q-value. Also, a modified DRL was applied to reduce convergence time in DRL. A QoE-aware caching problem for short video transmission at Edge in IoV networks was investigated in [49]. A DRL-based caching algorithm was proposed for improving QoE, which was modeled by a class-based user interest model. Similarly, Wu *et al.* [135] focused on caching short videos in mobile networks. The authors considered joint video quality selection and radio bearer control problem, which was constructed as a MDP problem. Their QoE objective is video quality level, which is quite abstract. Deep Reinforcement Learning was used to solve that problem, so long-term video quality profit was maximized and cost of bearer and latency was minimized. According to [87], a joint problem between caching video contents and user association at MEC was studied. A new DRL-based algorithm with Deep Determinisitic Policy Gradient (DDPG) was proposed to reduce backhaul transmission. The QoE value was estimated by using objective method, which was calculated from the QoS-QoE correlation function dependent on accumulated data rate and video-related parameters. In [86], Luo *et al.* considered the data caching and video transcoding selection problem at Edge. A DRL-based algorithm was proposed for optimizing QoE and energy consumption on SDN-based networks. The authors included dynamic of buffer, edge caching, video quality adaptation and transcoding, and transmission into the algorithm design. Their QoE function was related to content-related IFs or video-related factors. In [154], a new DRL-based caching algorithm was proposed to improve QoE. An algorithm was designed by deciding on caching location, caching capacity and caching priority.

In short, most of the prior works investigated data caching problems when QoS was taken into consideration. Only a few numbers of works included QoE into their problem formulation. With emerging ML approaches, ML-based algorithms have been proposed to solved these caching problems. Still, when QoE is considering, little designs of caching policies are based on ML approaches.

VII. RELATIONSHIPS BETWEEN QoS AND QoE

Previously, most QoE metrics can be derived from a combination of various QoS metrics. QoE metrics are defined to measure the actual user experiences. QoS metrics are used to indicate the performance state. Also, the performance state has a direct impact on user experiences. That is the main reason why many QoE metrics are estimated from QoS states. This section will unfold the strong relationship between QoE and QoS metrics. Prior works' correlation models between QoS and QoE are illustrated. Also, recent approaches to estimate QoE, e.g., ML methods, are further discussed.

Determining the relationships and correlation between QoS and QoE metrics can help us improve QoE models. In addition to QoS-QoE correlation models, the relationships of QoS and QoE in Machine Learning aspects are provided in this section.

A. QoS-QoE CORRELATION MODELS

Since QoS and QoE have a strong correlation, a number of prior works utilize their relationship for evaluating user experiences. There are some works such as [55], [84], [85], [87], [126], [134] that focus on determining the QoE estimation model from QoS metrics. Some works attempt to predict the objective model or estimate MOS scores, such as [55], [134]. Predicted the QoE value from QoS metrics is the target output of these works. QoS-QoE Logarithmic model is often used such as [55], [84], [85], [87], [133], [134]. Also, the summary of the correlation between QoS and QoE is illustrated in Table 11.

TABLE 10. QoS/QoE-aware data caching policies.

Paper	Strategy	Formulation	QoS/QoE Model	QoS/ QoE	ML	Optimization objective	Approach	Simulator	Tier	Year
[146]	data caching at Edge	Integer Programming (IP)	revenues, storage cost, transmission cost	QoS	No	maximize data caching revenues subject to data access latency constraints	heuristic	Numerical Studies (CPLEX)	Edge	2020
[153]	caching placement at Wireless Access Network	joint optimal caching placement and user-BS association to optimize the trade-off between load balancing and backhaul saving	a weighted objective of network utility and backhaul savings	QoS	No	maximize backhaul aware proportional fairness network utility	IP and an efficient algorithm iterating between cache-aware user association and association-aware content placement.	Numerical Studies	Access Network	2016
[148]	caching of microservices (containers) at Edge (cache node selection and microservice replacement)	Markov Decision Process (MDP)	fetching time	QoS	Yes	minimize fetching delay and maximize hit ratio.	Distributed Double Dueling Q-Network (D3QN) based algorithm	N/A	MEC	2021
[147]	caching contents at Edge nodes	Integer Non-Linear Programming (INLP)	content retrieval time	QoS	Yes	minimize content retrieval cost (jointly minimize traffic load and retrival delay cost)	RL or Q-Learning based algorithm (online optimal algorithm)	Simulation	Edge	2020
[150]	cooperative data caching at Fog RANs	cooperative caching problem on contraint cache storage + multiple user sharing content + user preferences + mobility + channel fading and conditions	average transmission delay in multi-users Fog RANs	QoS	Yes	minimize average transmission delay and improve Cache Hit Ratio	RL + Dueling DQN	Simulation	Fog RANs	2020
[151]	data caching at Fog RANs	joint optimization between caching strategy, computational offloading policy and radio resource allocation	average end-to-end latency	QoS	Yes	minimize average end-to-end delay	actor-critic RL (DRL) + DNN	Numerical Studies	Fog RANs	2018
[152]	data caching at Fog RANs	hidden Markov Process to characterize traffic model in FRANs	cache hit rate	QoS	Yes	minimize cache hit loss rate	RL	Simulation	Fog RANs	2019
[149]	data caching at Edge	Helper Decision Problem (HDP)	content access latency and traffic cost	QoS	Yes	minimize content access latency + traffic cost in CDN-based caching edge networks	DRL (Multi-Agent DRL)-based solution and they propose a framework called MacoCache	Numerical Studies (real trace data/pytorch)	MEC	2020
[134]	caching placement at Edge with cache-enabling UAV-assisted cellular networks	joint optimization problem (UAV deployment + caching placement)	MOS Objective model : downling and backhaul transmission delay and video-related metrics	QoE	No	maximize QoE	swap matching based UAV deployment algorithm (greedy algorithm)	Numercial studies	Edge	2020
[55]	cache placement of video contents at Base stations of Cloud RANs	Two-stages (1) long-term transmission-aware caching problem in the caching stage, (2) short-term transmission problem in the delivery stage	MOS objective model : logarithmic QoS-QoS correlation model	QoE	No	maximize weighted sum of long-term users'QoE subject to constraints of backhaul capacity, transmission power, storage size	iterative algorithm (classic optimization)	Simulation	cloud RAN	2019
[85]	caching placement of video streaming services at MEC	Integer NLP : joint caching placement, video quality decision, and user association for live streaming video services	MOS objective model : QoS-QoE logarithmic correlation model [137]	QoE	No	maximize weighted sum of QoE	Dual Lagarian Pricing	Simulation (Mobile Edge)	MEC	2020
[155]	data caching at Edge	propose a heuristic algorithm for maximizing the overall QoE for data caching at Edge	QoS + data storage capacity + caching cost	QoE	No	maximize the overall QoE	genetic algorithm	Simulation	Edge	2021
[133]	caching contents at Edge	managing Edge cache to reduce storage cost and transmission latency to improve QoE	QoE model (storage cost (available BW) + transmission latency (depend on SINR) + caching capacity)	QoE	Yes	maximize QoE	DRL + DNN approximator for estimating Q-Value	MATLAB	Edge	2019
[49]	caching placement at Roadside Units (RSU) for short video transmission at IoV networks	QoE-driven RSU cache update	class-based user interest model	QoE	Yes	optimize QoE	DRL	N/A	Access Network	2021
[135]	caching placment at MEC	joint video quality selection and radio bearer control optimization problem is constructed as MDP	video quality level	QoE	Yes	maximize long-term video quality profit + minimize cost of bearer and latency	DRL	Simulation	MEC	2019
[87]	caching contents of video streaming services at MEC	MDP	objective model by QoS-QoE correlation between accumulated data rate and video specific parameters	QoE	Yes	reducing backhaul transmission	DRL + DDPG (Deep Deterministic Policy Gradient) based algorithm	Simulation	MEC	2020
[86]	data caching and video transcoding selection problem at Edge.	Two problems : Constraint DMP solved by Lyponov optimization and MDP problems solved by A3C algorithm	objective model by video-related metrics	QoE	Yes	optimize QoE and energy consumption	DRL (consider buffer dynamics, video quality adaptation, edge caching, video transcoding and transmission)	Simulation	MEC	2020

The relationships between performance parameters and QoS metrics at each layer are shown in Table 11. At the physical layer, research works such as [55], [85], [87], [133], [134] model the QoE metrics from transmission rate ([55], [85], [87]) and transmission delay ([133], [134]). At the network layer, the authors in [84], [126] model the QoE metrics by including the network QoS such as jitter [126], packet loss [84], delay [126], and round trip time [84]. At the application layer, the authors in [55], [83], [85], [87], [134], [136] attempt to quantify the user experience by including content-related QoS parametric such as video or game applications. For video application, the following works [55], [85], [87], [134] and [83] are used in the QoE model. For game application, game parametric is used in [136] for the QoE model. At the user layer, the authors in [128] include user-related data in their QoE model. Still, it can be seen that none of the QoE models in our reviews consider the parameters in the IoT service layer in their QoE model. Because the IoT service layer is composed of functions for managing IoT devices, these functions are not standard and common.

In addition, recent QoE models such as [84], [127], [128], [131], [133], [136] are proposed to include the cost from the resource plane because user experiences are affected from resource constraints. For instance, resource constraints on the storage cost [133], the link capacity [84], the computing resources [131], [136], and energy consumption [127], [128] are used to formulate the QoE model.

According to Table 11, the QoE models and their QoS-QoE correlation functions that are formulated in prior works such as [55], [83], [84], [85], [87], [126], [127], [128], [131], [133], [134], [136] will be illustrated in the following paragraphs.

In [87] and [85], the authors consider a resource assignment problem for multiple user equipment (UE) when users request different video streaming requirements from multiple MEC base stations (BSs) in a cellular network, which its resource is limited. The constraint of resources is bandwidth requirement (from UE to BS) for the l-th enchancement video layers, caching capacity at mobile edge, and backhaul link between BS and core network. Their QoE metric is based on the logarithmic performance function, which is dependent on the property of both video-specific parameters and transmission rate.

$$Q(r_{u,v}, \bar{r}_{u,v}) = \alpha_v ln(\beta_v \frac{\sum_n \sum_l z_{n,v}^{n,l} d_{v,l}}{\bar{r}_{u,v}}) \tag{1}$$

where α_v and β_v are video-specific parameters, $\bar{r}_{u,v}$ is the highest aggregated data rate supported by UE type, $r_{u,v} = \sum_l z_{n,v}^{u,l} d_{v,l}$ is the total allocated data rate for UEs, and $z_{n,v}^{n,l}$ is the control parameter to allocate the user u requesting the l-th enhancement layer of video v from the n-th base station. Their QoE model can be interpreted as the total satisfaction level. Two layers are involved in the definition of the QoE : transmission rate, channel quality at the access network layer, and video parameters at the content layer. The authors showed that the proposed DDPG-based algorithm can improve QoE substantially.

According to [55], Sun *et al.* investigated the problem of video caching placement at base stations (BS) to relieve congestion of the link between BS and the core network. Their objective is to enhance the quality of each user's satisfaction. The QoE model is formulated by using both user subjective and QoS performance metrics for video streaming applications. Its model is based on Weber-Fechner Law, and its function is logarithmic due to the nature of the QoE. As a result, QoE can be computed from the correlation function of QoS in this case. In this paper, the QoE MOS objective function is estimated by the following equation.

$$Q_{k,u,H}^n = a_n ln(b_n \frac{R_{u,H}^n}{\hat{R}_k^n}) \tag{2}$$

where a_n and b_n are video-specific parameters, $R_{u,H}^n$ is the actual data rate of user u in the n-th group, $\hat{R}_k^n$ is the desired data rate of user u in group n, and H is the matrix of channel state information. The QoE represents the estimation of the satisfaction of each user (MOS). The QoE model is correlated with two QoS metrics: transmission rate and video-specific parameters. The authors proposed two states of caching policies that can improve the weighted sum of QoE when constraints on backhaul capacity, transmission power, and storage capacity are also considered.

In [134], the authors considered the offloading traffic problem by caching placement at Edge in unmanned aerial vehicle (UAV) deployment. Multimedia content is commonly distributed from UAVs to the backhaul system. So, the authors formulate the QoE model for maximizing user's satisfaction with the UAV system. In this work, the objective MOS score is estimated by the Logarithmic function of the QoE model. Their objective is to maximize all estimated MOS scores of users in the cell. Each user's MOS score is computed by

$$MOS_{m,k} = C_1 ln(\frac{1}{D_{m,k}}) + C_2 \tag{3}$$

where $MOS_{m,k}$ is the MOS score from UAV m to user k, C_1 and C_2 are system-specific parameters derived from the simulation, and $D_{m,k}$ is the transmission delay from UAV m to user k, which is dependent on the function of backhaul transmission rate from BS to UAV m, signal to noise ratio $SINR$ and the decision of caching content at BS.

The authors applied this QoE model because it is widely used for measuring user experience in multimedia streaming. Their QoE formulation is related to transmission rate and system-specific parameters. The proposed heuristic algorithm is shown to result in the near-optimal solution from their simulation.

Caching contents at Edge to maximize user experiences was investigated in [133]. He, Wang and Xu considered the following influence factors: storage cost, transmission latency, and caching capacity as their QoE model. The model was defined as follows.

$$QoE = -L_a - \xi C_o \tag{4}$$

where L_a is transmission latency which is dependent on SINR and the amount of transmitted data in each chunk, ξ is the weight parameter between latency and storage cost, and C_o is the storage cost which is dependent on storage pricing. Also, it is noted that the transmission latency function is dependent on the popularity of the content. If the cache hit rate is high (popularity is high), then the transmission latency is low.

The authors consider QoE influencing factors related to transmission latency, content popularity (content), and resource constraints (capacity and cost). Also, the authors utilized the RL approach with DQN to optimize user experience according to this QoE model. The experiment results show that QoE is improved by applying the algorithm.

In [84], the caching placement of photo contents in various geographic edge devices is investigated to optimize the user experience or QoE. The authors determine the relationship between photo loading time, photo size, caching locations, and network QoS. They model the QoE by using network QoS metrics such as link capacity, delay, and packet loss. With these QoS metrics, the TCP throughput model is derived. When the photo size is known, the system can utilize the photo size and throughput model to estimate the loading time. Their QoE model is based on this estimated loading time of cached photo contents.

$$QoE(t) = -0.80\, ln(t) + 3.77 \tag{5}$$

Similar to prior works [87] and [85], the QoE models were Logarithmic function where these coefficients were calculated by fitting waiting time of context browsing photos. Instead of using transmission rate, the authors utilized throughput rate to model the QoE function. Since users'satisfaction in photo content services is mainly due to the waiting time before the photo appears. Indeed, QoE is inferred from multiple network QoS metrics reflecting how fast the system can deliver those photo contents. This QoE model was derived from QoS metrics in the network layer. The authors provided the experiment results to demonstrate the relationship between QoS metrics and the QoE model.

In [126], Tsipis *et al.* studied the Fog allocation problems of rendering services for game providers. The problem were formulated as Facility Location Problem. They constructed the QoE model by using gaming network impairment to estimate Mean Opinion Score (MOS) of game users. The network impairment was calculated from network QoS metrics such as the average communication delay and the average jitter. Indeed, the MOS Score (QoE) was approximated by the following equation.

$$\begin{aligned} Q^t(u) = -0.00000587\, N(u)^3 + 0.00139\, N(u)^2 \\ - 0.114\, N(u) + 4.37 \end{aligned} \tag{6}$$

where $N^t(u) = 0.104D(u, v_s) \bar{+} J(\bar{u}, v_s)$, $J(\bar{u}, v_s)$ is the average jitter, and $D(\bar{u}, v_s)$ is the average communication delay of user u at timestep t where v_s is the fog renderer. Unlike the popular logarithmic model, this QoE function is the 3-rd degree polynomial function derived from gaming experience loss by network impairment. It is undeniable that good or bad gaming experiences are mainly dependent on the response time and the quality of response data from fog renderers for users. Both are highly related to the network impairment function $N^t(u)$. The MOS model is mainly approximated from network QoS metrics. Also, the authors proposed the QoE Rendering Service Allocation method and show that it can improve the QoE of users from their simulation results.

According to [136], Han *et al.* investigated the Virtual Machine Placement problem for Cloud gaming services. They proposed a game-theoretic algorithm for optimizing the total cost such that the QoE of users was still under satisfaction level. The authors utilized the delay, gaming resolution, and gaming FPS model to formulate the QoE objective, which was gaming experience loss, similar to [126]. The delay model was calculated by using the initial loading delay because the game was loading into the memory from the files on disks. The gaming resolution model depended on the available bandwidth on the wireless channel, which affects the feasible transmission rate, and the minimum required FPS. The gaming FPS model was dependent on how fast the current capability of resources could render the game. In short, it was dependent on the allocation of renderer resources too. Their QoE model was defined as follows.

$$L_n(a) = \lambda_1 D_n - \lambda_2 F_n(a) - \lambda_3 V_n \tag{7}$$

where D_n is the total initialization delay for loading the game from files, $F_n(a)$ is the gaming FPS that is calculated from a number of allocated resources such as CPU cores, memory, V_n is the actual gaming resolution, R_n is resources that are allocated for user n, and λ_i is the weight parameter for dividing the impact of each factor ($\sum_i \lambda_i = 1$).

In [136], the QoE model was related to application-specific parameters, e.g., resolution, FPS, assigned computing resources, and loading delay. Their game-theoretic approach achieved mutual satisfaction among users in the finite states when the proposed VM placement method was deployed.

Carvalho *et al.* investigated QoE-aware resource scheduling for container-based systems in [83] with the video streaming application. Their QoE metrics were based on the ITU-T P.1203 standard, the subjective Mean Opinion Score. The authors used the network level performance indicators to infer the QoE value. They utilized computing resource data (CPU, disk, memory, file system, network) as the input ; then, they performed the subjective test to verify MOS. Their QoE model was constructed by training the collected input data using the Machine Learning approach. Since the authors considered video streaming as the main service, their QoE model was the MOS subjective score of video referred from the ITU standard. In short, the authors created the QoE model from network and resource usage data. The proposed scheduler significantly improved the average QoE and reduced resource usage, according to their results.

In [131], the authors investigated the Virtual Machine placement problem such that the placement policy attempt to optimize multi-objective such as hardware outage, power consumption and performance. The following equation defines the QoE model at each time step t.

$$QoE^t = \frac{\sum_{i=1}^{M} \rho_i^{L^t,t} \sum_{j=1}^{N} \lambda_{ij}^{L^t,t} \Pi_{r=1,r\neq j}^{N} \theta_{y_{ij}^{L^t,t} y_{ir}^{L^t,t}}}{\sum_{i=1}^{M} \sum_{j=1}^{N} \lambda_{ij}^{L^t,t}} \tag{8}$$

where M is the number of Physical Machine (PM), N is the number of VMs, $\lambda_{ij}^{L^t,t}$ is the sum of used cpu, disk, network of i-th Physical Machine at time step t, $\rho_i^{L^t,t}$ is the reliable factor (impact of churn) of i-th PM and $\theta_{h,k}$ is the interference between two classes of VMs deployed on the same PM.

Their QoE model was mainly derived from the user experience impact of co-location interference of the VM allocation on the same PM. The interference was dependent on the number of allocated resources among different users. The author's QoE model was mainly correlated to performance metrics such as interference level at PMs. The authors proposed a heuristic Deep Learning-based policy to place VMs on physical servers. Their results showed that it could outperform bin-packing heuristic policy.

Additionally, Luo *et al.* studied the offloading scheduling problems at Edge devices to optimize the response time and battery life for end-user devices [127]. Also, the proposed offloading policy took the QoE model into account, resulting in an improvement in both latency and power usage. Latency and energy consumption models were used to construct the QoE model. For latency model, it was composed of the computation latency and communication latency. The energy cost model was composed of the local execution energy cost and offloading energy cost of user devices. The QoE-driven cost was modeled as follows.

$$Q = T_L + \eta E_N \tag{9}$$

where T_L is the total latency, E_N is the total energy consumption, and η is the weighted factor.

Since the purpose of offloading at Edge was to improve the latency and reduce the power usage on mobile devices, the user experience was directly affected by both metrics. The QoE model was functions of QoS metrics such as latency (computation and communication) and energy consumption (mobile and edge). According to the results, the proposed scheme could achieve a better performance in terms of latency and power usage when compared with benchmark policies.

Also, the computation offloading problem from Mobile to Cloud was investigated in [128] by Hong and Kim. In addition to the trade-off between latency and energy consumption, the authors explicitly constructed the QoE model with the additional user context, such as user tendency and user battery level. Their QoE-aware cost function was determined from energy consumption per timeslot, latency cost at slot t, and user context cost.

$$\xi_t(l_t, s_t) = w_{QoE}(1-\alpha B)\frac{E_t(s_t)}{E_L} + (1-w_{QoE})\alpha B\frac{D_t(l_t)}{D_L} \tag{10}$$

where B is the normalized battery level, α is the application characteristics, E_t is the per-slot energy consumption at timeslot t, s_t is the amount of transmit data (bits) at slot t, l_t is the remaining bits to tx, $D_t(l_t)$ is the latency cost, $E_t(s_t)$ is the energy cost at timeslot t, $E_L = B * C_{batt}$ is the battery capacity in Joules, $D_L = T_\tau E_L$ is the normalized factors to make unitless combination of energy and latency, and w_{QoE} is the weight factor between two cost types. In addition to latency and energy usage, battery level and user tendency were included in this QoE model to reflect more accurate user experiences. Their QoE model was related to user context information (battery level, user tendency) and performance (QoS) metrics (latency and energy consumption). The authors proposed the suboptimal policy for offloading scheduling with little extra costs while QoE was significantly improved.

In short, QoE metrics in prior works are highly correlated with performance metrics. Many works utilized QoS metrics as the input parameters for estimating the QoE value. They are also used to approximate MOS scores.

B. PREDICTING QoE FROM QoS METRICS

In the prior section, various QoE metrics are modeled from the objective function of performance or QoS metrics. However, when many QoS metrics are utilized to estimate QoE, the computational complexity of the QoE estimation function becomes too high. As a result, another approach such as ML emerges as a proper solution for modeling the QoE functions. Still, QoS metrics from multiple layers are used as the input data for building the QoE model.

When QoE output data can be measured or collected, supervised learning methods such as Naive Bayes, Support Vector Machine, K-Nearest Neighbor, Decision Tree, Random Forest, Regression, and Neural Network can be utilized for training the model. If QoE output data is not subjectively measured, then QoE objective function can still be used with more dynamic ML learning models such as unsupervised learning, reinforcement learning, and deep learning models.

For supervised learning methods, ML methods for QoE prediction are proposed in many prior works such as Naive Bayes (NB) in [50], [63], [77], Support Vector Machine (SVM) in [50], [63], k-Nearest Neighbor (k-NN) in [50], [51], [89], Decision Tree (DT) in [50], [51], [62], [63], [89], Random Forest (RF) in [50], [51], [89], Neural Network (NN) in [50], [63], [77], [89], [91].

These research works utilized various QoS metrics as their input for creating and evaluating the QoE model. For instance, network QoS metrics are utilized in [50], [52], [61], [62], [63], [77], [90], [91] and [53] as the input metrics for training the QoE mode. Moreover, application-related metrics are used as the input for the QoE model in [58], [61], [78], [89] and [68] for modeling the QoE. User-related metrics are also used as the input data in [51], [52] and [53]. Details for modeling the QoE from QoS metrics are elaborated as follows.

TABLE 11. Correlation between QoS and QoE.

Paper	QoE objective	Layer	QoS & performance metrics	Resources	reference equations	Year
[87],[85]	Total service satisfaction	PHY + APP	transmission rate + video parameters		(1)	2020
[55]	QoE of each user (MOS score)	PHY + APP	transmission rate + video parameters		(2)	2019
[134]	QoE MOS of user k with UAV m	PHY + APP	transmission delay + video parameters		(3)	2020
[133]	QoE is performance and cost	PHY + RES	transmission latency + resource (storage cost)		(4)	2019
[84]	MOS extracted from picture loading time	NET	network QoS (packet loss + RTT + TCP throughput)	resource (link capacity)	(5)	2016
[126]	MOS extracted from network impairment	NET	network (jitter + delay)		(6)	2020
[136]	QoE is quanitfied gaming exp. loss	APP	content (total initialized delay + gaming FPS + gaming resolution)	resources (allocated to user)	(7)	2020
[131]	Discrete time QoE	PHY	system (Interference between 2 classes of VM on the same PM)	resource (number of PM, VM, cpu usage)	(8)	2020
[127]	QoE driven cost (latency + energy)	PHY + NET	physical + network (total latency : commu. + computation latency)	resource (total energy consumption : local + offloading energy)	(9)	2019
[128]	QoE evaluated from performance, content and user context	PHY + USER	physical (per-slot tx delay) + context (battery level, amount of offloading data, total slots) + User tendency	resource (energy consumption (per-slot))	(10)	2016

According to [50], network QoS, video properties, terminal characteristics, and user-profile types as the input metrics for training are used by the authors to train the objective QoE model. ML methods such as NB, SVM, k-NN, DT, RF, and NN are deployed to classify a set of input data into MOS scores. The authors used the QoE subjective measurement or MOS as the output data for training. The accuracy results for DT/RF, 4-NN, NB/SVM, and NNT are 75%, 50%, 55-65%, and 65%, respectively.

In [89], metrics such as full reference parametric (SSIM, VQM) and application-related metrics (resolution, bit rate, frame rate) are used as the input for training the model. The objective ML model is predicted by using DT, NN, k-NN, and RF methods. The subjective method, such as DCR (Degradation Category Rating), is used for measuring the QoE value. The disadvantage is that the collection period of data is around 3 hours.

Letaifa proposed to optimize end-to-end QoE on video streaming in an SDN context [61]. At the network layer, network QoS metrics such as RTT, jitter, bandwidth, and delay are used as the input for creating the QoE model. At the application layer, video parameters such as buffering time, resolution, bit rate, frame rate, VQM, SSIM, PSNR are also the input for training the QoE model. The authors created an ML model by fitting experimental data using the regression approach. The model is then used to estimate MOS when network QoS metrics are known. Training datasets are collected from real MOS scores and network QoS from simulated SDN networks. The predicted MOS and the video quality had a correlation coefficient of around 0.8.

According to [62], Menkovski *et al.* present ML techniques for modeling the QoE, which is derived from network and application QoS parameters. Predicting QoE at run-time by using network and application QoS metrics is studied in this work. The authors proposed to build a QoE model in an online fashion when feedback is a subjective measurement. Online ML methods such as DT are used for training the model. Prediction accuracy is over 90%. The advantage of online training is the improvement of automation and adaptability.

Next, MLQoE (user-centric QoE prediction) for voice/video related applications was proposed in [63]. Three datasets of unidirectional VoIP were used for training the model. Network QoS metrics such as packet loss, delay packet interarrival time, and jitter were the input of the QoE model. The authors created the model using a modular method that applied several ML regression algorithms for training. Nested Cross-Validation was used to select the best algorithm from ANN, SVM, Regression Machine, DT, and Gaussian Naive Bayes classifier. The training procedure was offline. The result outperformed the standard model such as E-model, PESQ, WFL, and IQX. The disadvantage was that the method requires a large-scale dataset to perform well.

In addition, Vasilev *et al.* in [77] focused on user perception of video quality. Three main QoE factors, such as average video bit rate, average video bit rate variation, and re-buffering ratio, were utilized to build a BN model to predict re-buffering events. Also, the authors used a custom NN to verify the prediction performance. They utilized Adaptive Multimedia Streaming Simulator Framework (AMust) in ns-3 to simulate a large-scale dataset. The simulation duration is one month with 69,000 video sessions and 50 associated variables such as context data on network congestion and characteristics, QoS metrics, QoE factors, and hidden QoE variables. The prediction achieved around 97% accuracy of the true distribution. The preparation of the training dataset was tedious, and the model was not dynamic.

Moreover, Laiche *et al.* proposed in [51] to include user-related factors, content-related and system factors for training the QoE model in the video application. They utilized user factors and social context factors such as popularity level (number of likes, dislikes, comments, views) and engagement level (playtime, view duration). The prediction model was developed by using ML algorithms such as k-NN, RF, and DT when quality factors from users were used as the input. The open-source datasets were collected from crowdsourcing and composed of 1,125 sample videos with individual logs. The prediction accuracies for DT, RF, and k-NN were 94%, 82%, and 56 %, respectively. However, this method required engagement from users to create the model.

QoE optimization via Data Fusion (QoEDF) framework was proposed in [91] for MIoT applications. First, the framework mapped user factors and network data with associated MOS for creating the QoE model. Then, the framework attempted to optimize the model continuously. Network QoS (bandwidth, delay, loss, jitter, throughput, and others), application factors (encoding, frame rate, sampling rate), and historical MOS data were utilized as the input for creating the QoE model. Neural Network was designed to predict MOS for real-time QoE optimization. The system monitored the QoE and network characteristics by collecting massive historical multimodal data (network-related data and average MOS scores). The framework optimized QoE with the cycle of collecting data, training, and updating the model.

It is sometimes difficult to perform a large subjective measurement. In this case, Unsupervised Learning and recent Deep Learning methods are used to train the QoE model continuously. For instance, contemporary works such as [52], [53], [58], [68], and [90] are applied Deep Learning techniques for predicting the QoE model.

As shown in [90], Lopez-Martin *et al.* investigated how to predict the QoE from network packet information by using the Deep Learning model. Network packet information and QoS were used as the input for the model. The authors proposed transforming the network flow information as Psuedo-images that allow us to apply CNN. The proposed classifier was based on CNN, RNN, and Gaussian Process (GP) classifier. A dataset was collected from controlled environments with several viewers evaluating video transmission when different network conditions. The accuracy was around 60%-80% for unbalanced labels and 80%-95% for other labels. The problem was that the training dataset was highly unbalanced, and subjective results had noisy results.

In addtion, Tao *et al.* studied a data-driven QoE prediction for mobile video transmission [52]. Network-related factors (89 network parameters) were used for training the QoE model. DNN was utilized for learning the relationship between the network parameters and the subjective test scores. A large dataset of over 80000 pieces and four subjective test scores were used. It can outperform other state-of-the-art methods. The advantage was that the authors used a large network of information to create the model.

Then, a hybrid Neural Network was designed for the QoE evaluation in [53] because subjective methods are expensive, error-prone, and unreliable. The authors used QoS metrics in user, system, and context to construct the QoE model. The Deep Learning approach was used with DNN and improved RNN. Also, real-world datasets from large-scale VoD service providers were used as the training data. The prediction accuracy was around 80%.

In [58], Lekharu *et al.* proposed the ML approach to predict QoE based on network data. Input data, such as network data (network bandwidth, actual bit rate, segment size) and content data (video quality, buffering time, smoothness), were fed into the training model. A DNN was used to predict the video quality model by inferring from video bit rates. This DNN was based on LSTM and CNN (LASH) to estimate the video quality (QoE) and select a a more appropriate bit rate to maximize QoE. Also, the Actor-Critic network was used for training data with a standard HSDPA Norway dataset.

According to [68], Shen *et al.* proposed a DeepQoE algorithm for measuring the QoE in real-time. Network QoS metrics such as RTT in upstream traffic were used as the input for training the model. A Deep Learning approach with a CNN-based classifier was applied to measure the video QoE metrics composed of start-up delay, rebuffering event, and video resolution. The author's utilized real-world datasets from two popular service providers (Youtube & Bilibili) in training. Also, the prediction accuracy was over 90% for identifying video QoE metrics.

Concisely, various QoS metrics were widely used as the input metrics for building the QoE model. When the number of metrics increases, many studies attempt to apply ML methods to create the model. Recently, Deep Learning techniques have been widely studied for QoE Evaluation. Therefore, QoS metrics are tightly coupled with the QoE model.

VIII. FUTURE DIRECTIONS

Driven by QoE, multiple QoE-related problems are intertwined into multi-dimensional problems. First, various IoT applications require different QoE metrics because of content-related factors and user-related factors. Users with the same similarity in characteristics often access to the same application. So, each IoT application can have its own QoE perspective or multi-dimensional QoE metrics of applications. Next, emerging systems e.g., MEC/Fog are multi-tier architectures. Each tier has different resource specification and capabilities as well as QoE aspects for system-related factors. Multi-dimension of QoE metrics in this multi-tier system arises to be a challenging problem. Resource management for these multi-dimensional QoE metrics in a multi-tier system is non-trivial. Moreover, multi-dimensional QoE Cause Factors appearing in this architecture have to be further investigated.

According to multi-dimensional characteristics, we envision an emerging AI-based layers for QoE management with multi-tier QoE metrics and multi-dimensional QoE factors on multi-tier systems. Unlike others, the expected AI-based layers are extensively discussed for opening a path to new research topics.

A. EMERGING AI-BASED LAYERS FOR QoE MANAGEMENT

According to prior sections, recent ML-based techniques were proposed to solve various QoE management problems. These techniques are promising because they can be performed online with no model requirements. While the IoT system is continuously growing larger, more AI-related parts are used for QoE management. Most QoE related problems are multi-dimensional problems that require high computational resources for processing. It is unavoidable to have a separated AI infrastructure for managing QoE.

In Figure 7, a new AI-based layers for QoE management is envisioned to be a part of a larger IoT system. There are

TABLE 12. Relations of QoS metrics and QoE learning models.

Paper	input data (QoS metrics)	ML models	ML methods	traning dataset	prediction accuracy	pros / cons
[50]	network parameters, video characteristics, terminal characteristics, types of users'profiles	Supervised Learning	NB, SVM, k-NN, DT, RF, NN (classify a set of input data into 5 categories (MOS score 1-5) discrete nature)		DT,RF around 75%, 4-NN 50%, NB, SVM around 55-65%, NNT around 65%	
[89]	Full reference parametric (SSIM, VQM) and application metrics (resolution, bit rate, frame rate)	Supervised Learning	DT, Neural Network, k-NN, RF	DCR (Degradation Category Rating) as subjective method (MOS)		data collection period is 3 hours
[61]	network QoS metrics such as RTT, jitter, bandwidth, delay and video parameters such as buffering time, resolution, bit rate, frame rate, VQM, SSIM, PSNR	Supervised Learning	Regression method	collecting a data set, which composes of real MOS score and network QoS, by using web-based application from simulated SDN network.	correlation coefficient between the predicted MOS and the video quality around 0.8	
[62]	network and application QoS metrics	Supervised Learning	online learning algorithm (Decision Tree model : Hoeffding Tree)	data collection is performed at run-time (network probing and customer feedback)	accuracy over 90% datapoint correctly estimated	online training improves automation and apaptability
[63]	network QoS metrics such as packet loss, delay packet inter-arrival time, jitter	Supervised Learning	Nested Cross Validation is used to select the best algorithm from ANN, SVM, Regression Machine, DT, Gaussian Naive Bayes classifier.	subjective opinion data from actual users with three data set with different network statistics (avg. packet loss, avg. jitter)	outperforms to the state-of-the-art model such as E-model, PESQ, WFL, IQX.	Need multiple of large size datasets for MLQoE
[77]	content (average video bit rate, average video bit rate variation, re-buffering ratio)	Supervised Learning	Bayesian Network (BN) model to predict re-buffering ratio, and Custom NN to verfity the BN model	The authors use Adaptive Multimedia Streaming Simulator Framework (AMust) in ns-3 for simulating a large dataset.	around 97% accuracy of the true distribution	long preparation of data, it is hard to change the model dynamically
[51]	user factors : social context factors : (only these two input metrics are used) (1) popularity level : number of likes, dislikes, comments, views, ... (2) engagement level : playtime, view duration prop	Supervised Learning	k-NN, RF, and DT	The dataset is an open source and is collected by using crowdsourcing, which is composed of 1125 sample videos and individual logs.	DT 94%, RF 82%, k-NN 56%	Need engagement from users and do not consider system-related QoS metrics.
[91]	(1) network-related factors : bandwidth, delay, loss, jitter, throughput, ... (2) content-related (media) factors : encoding, frame rate, sampling rate, synchronization, ... (3) historical MOS data (average) to create fusion model (use model later for real-time QoE optimization)	Neural network model to predict MOS (QoE) for real-time QoE optimization	Neural network : FFBPNN-based	monitor QoE and network characteristics, collect hugh historical multimodal data (network-related data and average MOS), MOS data is subjective measurement	estimation error less than 0.1 %	cycle of collecting data, training / updating model, applying model for optimizing QoE
[90]	network-related factors : network packet information	Deep Learning Model	Classifier is based on CNN, RNN, Gaussian Process (GP) classifier.	Small dataset from controlled environments	the accuracy for unbalanced label 60%-80% and for other labels 80%-95%	Small dataset
[52]	network-related factor : some of 89 network parameters	Deep Learning Model	DNN	large dataset over 80000 pieces of data and 4 subjective test scores		a large dataset with a large number of network parameters
[53]	User (user behavior with sequence of actions : pause, seek), system related to user (buffering, error) system related to context (bandwidth, loading time), context (content type, ISP)	Deep Learning Model	Deep learning with hybrid network (DNN and improved RNN)	real-world dataset from a large scale VoD service providers	around 80 %	real-world and large scale dataset
[58]	Network (Network bandwitdh, Actual bit rate, Segment size) , Content (Video Quality, Buffering Time, Smoothness)	Deep Neural Network	DNN based on LSTM and CNN (LASH)	Standard dataset HSDPA Norway dataset		
[68]	Network QoS (RTT of upstream traffic)	Deep Learning Model	CNN based classifier to measure video QoE metrics	real-world datasets from two popular service providers (Youtube & Bilibili)	over 90% for identifying start-up delay, rebuffering event, video resolution	real dataset

four major layers such as Data storage layer, Training layer, Prediction layer and Validation layer that can be implemented for support QoE management.

In multi-tier architecture, QoE Cause Factors are retrieved from each tier, and these factors have to be used in various QoE purposes e.g., QoE prediction, QoE-aware data caching, QoE-aware placment control and others. These QoE factors are originated from either Mobile Devices, Edge nodes, Fog servers and Cloud servers. At multi-tier data storages, QoE-related factors at each tier are monitored and collected for later processing. Next, these data are classified and transferred by appropriate techniques in the Data storage layer for processing. QoE data can be classified by different IoT applications and users into multiple datasets. We can see that QoE management in emerging networks are associated with three dimensions : tier, application and user domain. Then, QoE data in each dataset is passing through the Training Layer for training QoE model. The Training Layer can be used for building QoE model corresponding to the objective domain. It can be seen as a set of Neural Networks e.g., DNN, RNN, CNN, LTSM and other networks in ML literature. Given each completed training model, the Prediction Layer is deployed with prediction service instances that using training models for predicting QoE. At last, the QoE metric associated with each user and application is received and validated by user subjective test in the Validation Layer. The Validation Layer provides QoE subjective measurement and validation methods of training models with dataset for validation. Resulted QoE metrics will be utilized by similar AI-based architecture for managing resource and control IoT systems as prior discussed in IV. For these reasons, this envisioned high-level architecture could be investigated further in the future.

1) DATA STORAGE LAYER

The data storage layer is designed for storing large logging data from performance metrics. Since logging QoE data collected from multiple users/applications are substantial, this layer is designed how we can retrieve data for training and evaluating QoE in real-time. This layer is quite critical because the timing for retrieving data affects the response time from the evaluation. How to retrieve data effectively from the storage will determine the system's overall

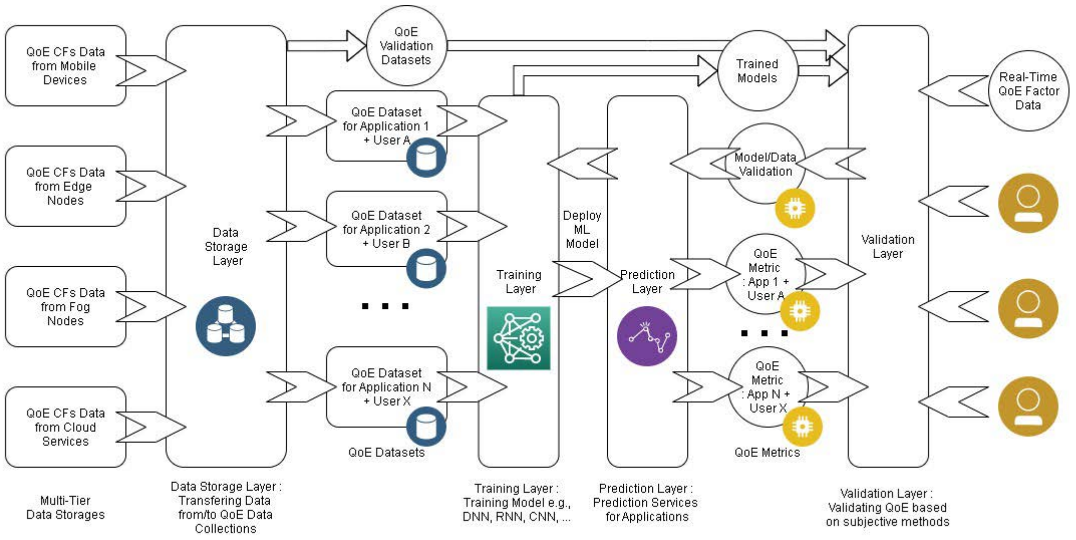
QoE CFs Data from Mobile Devices
QoE CFs Data from Edge Nodes
QoE CFs Data from Fog Nodes
QoE CFs Data from Cloud Services
Multi-Tier Data Storages
Data Storage Layer
Data Storage Layer : Transfering Data from/to QoE Data Collections
QoE Validation Datasets
QoE Dataset for Application 1 + User A
QoE Dataset for Application 2 + User B
QoE Dataset for Application N + User X
QoE Datasets
Training Layer
Training Layer : Training Model e.g., DNN, RNN, CNN, ...
Deploy ML Model
Prediction Layer
Prediction Layer : Prediction Services for Applications
Trained Models
Model/Data Validation
QoE Metric : App 1 + User A
QoE Metric : App N + User X
QoE Metrics
Validation Layer
Validation Layer : Validating QoE based on subjective methods
Real-Time QoE Factor Data

FIGURE 7. Emerging AI-based layers for QoE.

performance. Also, these QoE datasets are composed of large-scale QoS data from various layers, so storing and efficiently transferring data is challenging. Distributed file systems are sometimes necessary for improving computation time and performance. Typically, there are two types of datasets: training and validation datasets. The training dataset is utilized for creating a model. Then, the validation dataset is applied to evaluate the model's accuracy. For instance, the authors in [51] divided the whole dataset into 70% training dataset and 30% validating dataset. Datasets can be drawn from various sources, such as real data collected from the environments and simulated data from the controlled environments. The time for collecting the data depends on the proposed model, whether it requires a large dataset for training or not.

For instance, some datasets were simulated and acquired from controlled environments such as [61] and [77]. In [61], the authors acquired the network data from the simulated SDN network. Similarly, Vasilev *et al.* in [77] collected the statistical data from ns-3 simulated networks, and the collection period was around one month. The datasets were composed of 69000 video sessions with 50 associated variables which were related to the data of network congestion and characteristics, QoS metrics, QoE factors, and variables. In contrast, some works utilized the dataset acquired from the real system, such as [51], [62], [63], and [91]. Menkovski *et al.* in [62] collected data from network probing from the real system. Also, the dataset utilized for creating the model in [51] was open-source and derived from crowdsourcing composed of 1125 sample videos and individual logs. The datasets in [91] were created from monitoring and collecting network statistics of the real system.

Because each application requires different QoE points of view, multimedia applications are focused on video quality perception, while tracking applications are concerned with the accuracy of the location of the tracked objects. On the same network/system, it is feasible that multiple QoE models can be created from different datasets or models for various applications. It is seen that the dataset for each QoE model can be quite large and subtle and demand a long period of aggregation. Moreover, if data is collected from different geographic locations, the decentralization of datasets will occur. How to categorize and manage these datasets in a systematic way in this layer is still questionable and requires further investigation.

The Data Storage Layer is responsible for managing large-scale data storing for training and validating. Sometimes QoE data are collected and distributed all over different geographic areas. How to store and retrieve data for training and validating from either centralized or decentralized storage is quite challenging. Most of the time, data can be either structured or non-structured; storing data efficiently is not trivial.

2) TRAINING LAYER

The Training layer is designed for creating the ML model by ingesting data from the Data Storage layer. It receives corresponding datasets with important statistics of the physical, network, and other QoS-related metrics for training the proper QoE model. Training a particular model requires a significant amount of computing resources. It is a time-consuming process. Also, training the model requires a specific network design for the learning model. QoE models of applications have different learning structure models. By separating the training process from the evaluation process, the system can process the computational workloads in parallel. Still, more computing resources are required. As a result, this layer help saves processing time while the QoE prediction can still be estimated in run-time applications.

In addition, recent works have used various Machine Learning approaches to create the QoE model. As prior discussion, the Supervised Learning approaches such as NB, SVM, k-NN, DT, RF, and NN were widely used in prior works such as [50], [51], [62], [63], [77], [89], [91]. These approaches require a good dataset with QoE subjective measurement. Training datasets are collected from the environments, which can be drawn from the Data Storage Layer. It was shown that datasets were collected from simulated networks such as [51], [61], [62], [77], [89]. In [61], statistics data was monitored and collected from simulated SDN networks. Furthermore, datasets in [77] were drawn from ns-3 simulated networks. In contrast, the authors in [62] collected data from network probing in the real system. Open-source crowdsourcing data from the real system in [51] was used to create the model. Also, some works included subjective measurement procedures to create their datasets, such as MOS subjective measurement in [50], [89] with DCR, and [62].

Deep Learning has recently become the state-of-the-art approach for creating the ML model due to its impressive prediction accuracy. For instance, DeepQoE in [68] was proposed to predict the QoE factors; its prediction accuracy was over 90% to identify the start-up delay, rebuffering event, and video resolution. Their model was CNN based classifier for measuring video QoE metrics. Various models of Deep Learning approaches were proposed in prior works, e.g., DNN in [52], [53], [58], and CNN in [68], [90]. In [58], Lekharu *et al.* used Deep Neural Network (DNN) to predict the video quality model by inferring from video bit rates. Their DNN was based on LSTM and CNN (LASH), and it was used to estimate the QoE (quality of videos) and select the proper bit-rate to maximize QoE. According to [53], the authors deployed DNN with improved RNN to train the model with VoD service providers' real-world datasets. The custom DNN model in [53] can predict the QoE with around 80% of prediction accuracy. The authors in [52] applied the DNN model to learn the relationship between network information and the subjective test scores, and their dataset was composed of more than 80000 pieces of network data. The new classifier was investigated in [90] and based on CNN, RNN, and Gaussian Process classifier. The QoE Deep Learning-based model was just another type of network with some intensive label computation for learning purposes.

In short, different types of QoE models can be designed in the Training Layer. Then, computation resource issues become a concern in achieving good performance. Sufficient resources must be allocated to match the amount of data transferred from the Data Storage Layer and the computation requirements. Once the model training is complete, it must be validated in the Validation Layer. After completing the validation, the model will be deployed into the Prediction Layer so that the system can evaluate QoE in real-time. How to efficiently manage resources in the Training Layer for each model can be further studied. Transferring a massive amount of data from each dataset between the Training Layer and the Validation Layer for learning and evaluating purposes is quite challenging. Additionally, the collected data may be distributed among various geographic locations, and the decentralized model may be necessary for deployment.

3) PREDICTION LAYER

The Prediction Layer is where the deployment of the QoE-designed model implements to predict the target outcome. When the deployment of the QoE model has proceeded, it monitors and retrieves necessary input data from various points of system layers, e.g., user information, context-related data, content parametric, control/network parameters, and physical data. Then, the system processes all of the data with the deployed model, and the target output is computed and released for being used by other modules.

Deploying the model in run-time can be different from the trained model. Several optimization techniques used to simplify the model help the deployment consume fewer resources with faster processing time. Most of the time, computing resources are restricted during high workload utilization. Helpful techniques such as pruning [156], quantization [157], [158], and aggregation can be applied to optimize the ML model. Similarly, as discussed in [159], the computational cost of the Deep Learning model can be improved by reducing the spatial complexity, such as pruning the model parameters, parameter sharing, network quantization, and others.

First, The quantization approach was proposed by [157] to improve the speed of the forward and backward propagation calculation by lowering bits of models from Float Point 32 bits to Integer 8 bits. The technique attempt to reduce the computation cost while retaining the accuracy to be nearly the same. In [158], research at Qualcomm AI Research investigates how the quantization technique can reduce the computational cost and latency in Neural networks. The authors discussed how the AI Model Efficiency Toolkit (AIMET), the library for quantization and compression of the AI model. The tool provided an efficient way of optimizing the Neural Network model. Network pruning can be used to prune the completed trained model or be included in the online training procedure. In practice, pruning the network of the completed trained model typically gives higher accuracy than the other approach, as shown in [160]. Examples of pruning techniques are shown in [156], [161]. Additionally, both pruning and quantization techniques simplify the model while improving the prediction accuracy [161]. Parameter sharing is the clustering approach for grouping a pattern of parameters, so the computational costs decrease.

In the production environment, various approaches are utilized for deploying the model, such as a web service, a batch prediction, and an embedded model. The model is completely trained and validated for web services before its deployment, so all users can access the model as a service via the Application Interface. On the contrary, the batch model is continuously updated and retrained with the updated dataset according to its scheduling update time. In practice, computing resources tend to be limited, so the simplified or embedded model is more suitable in resource-constrained environments.

Given the infrastructure, the intensive computing model requires additional Hardware such as Graphic Processing Units (GPUs) and Tensor Processing Units (TPUs). The model can be coded in various tools, e.g., Keras, Pytorch, and TensorFlow. Also, recent popular approaches [162] for deploying these models are container-based deployment, e.g., Docker or Kubernetes. When the model is deployed onto these systems, the model's scalability for servicing users becomes more elastic than the traditional approaches. It is proper to deploy a medium-to-large scale model when resources on the infrastructure are numerous. For resource-limited environments, simplified and optimized ML models are recommended for deploying as an embedded model.

In brief, the Prediction Layer is focused on the model deployment. The run-time model is not necessarily the same as the complete-trained model. It is dependent on the available resources in the infrastructure. Commonly, the well-trained model can be further optimized and reduced its size by various techniques, e.g., pruning, quantization, parameter tuning, and sharing. Each application may prefer different types of deployments, e.g., as a Web service, batch prediction, or embedded model. Web service is proper when the application usage does not often require updates on the model. When the model has to be quite dynamic, the regular update on the model is preferred as in the batch prediction. If resources are limited, the embedded model is suggested rather than other models. Even though tools or languages used for creating models are different, they can still be deployed on scalable container-based technologies such as Docker or Kubernetes.

4) VALIDATION LAYER

The Validation Layer provides a mean to validate data for the Training Layer and the well-trained model created from the Training Layer. When the model is completely trained, it has to be tested and validated before deployment for run-time prediction. To validate the model requires another dataset for validation purposes. Typically, the whole dataset are divided into multiple datasets for training, validating and testing purposes. Various approaches for splitting datasets, such as holdout set, cross-validation, and others, can be applied to verify whether the accuracy of the model is acceptable or not. Typically, the datasets are divided into the training dataset 70% and the validation dataset 30% as shown in [51]. However, the additional holdout and cross-validation methods can be used to divide datasets to improve the model accuracy and to validate the well-trained model. For instance, holdout and cross-validation are used in [163] to validate their ML model. The additional holdout dataset is normally used for eliminating the overfitting of the model after the validation as the final evaluation.

The Cross-Validation (CV) techniques are commonly used for model evaluation and selection in the ML field. Various Cross-Validation techniques such as k-fold CV [163], [164], Leave-One-Out CV, Leave-One-Group-Out CV, Nested CV [63], and others are used to evaluate the model. Unlike holdout fixed dataset, k-fold Cross Validation applies k-iteration of training with k non-overlapping splitting datasets from the original dataset to reduce the dataset's bias/skew of training. Its performance is the average value from each round performance. The variance of k-fold CV is Leave-One-Out CV, Leave-One-Group-Out CV, and Nested CV. For Leave-One-Out CV, each sample data is pulled out for validation while all other data are used for training. For Leave-One-Group CV, the dataset is categorized according to the data group, and only one group of the data is processed in each fold/iteration of training. For Nested CV, k-fold CV is applied at two places: inner and outer iteration. In the outer iteration, the k-fold CV is used as the original k-fold CV for evaluating the model. In the inner iteration, k-fold CV is used in the training dataset parts for hyperparameter tuning. For instance, the authors in [164] use a 5-fold cross-validation method for their ML model to predict web QoE metrics from network QoS. Nested CV is utilized in [63] discussed in the prior section. This method is also used in model selection to select the best-performing model.

Additionally, input data validation is also critical for training an ML model. If the input data is skewed, biased, and prone to error, the prediction accuracy is likely low. Moreover, the quality of input data is so critical that it can determine the actual accuracy of the model according to [165]. The well-trained model can result in an inaccurate prediction outcome with improper raw input data. In [165], the authors investigated how to validate large input data into an ML pipeline when data has no pattern, is schema-free, and training skew. In [166], the authors proposed the TensorFlow Data Validation (TFDV), developed by Google, for validating the ML model in the ML platform. It was used to detect the anomaly data scheme arriving at the ML pipeline, so systems and staff can quickly solve early anomaly detection.

When QoE is in consideration, the subjective data for each user has to be collected and included in the training and validation procedure. In the previous section, various subjective measurement types can evaluate actual user experiences or QoE metrics. For offline measurement, all subjective test data, e.g., MOS, DMOS, has to be prepared and retrieved from users before training and validating. For online measurement, subjective measurement proceeds continuously until the model's accuracy reaches the target. As a result, monitoring data and filtering the quality of the input data is necessary for this layer. The User Interface development for evaluating user experience is essential to facilitate access.

In summary, the Validation Layer is responsible for three main tasks: ML model validation, data validation, and QoE subjective test/measurement for validation purposes. For the model validation, Cross-Validation is commonly used for managing the whole dataset for training and validating the model. The interaction between the Training Layer and the Validation Layer is quite necessary, and how to interact between those layers is quite challenging. Since the input data for training and validating the model is crucial, recent works focused on controlling the quality of the large-scale dataset in the ML pipeline. When the QoE model is concentrated, subjective score data or metrics related to user experiences are essentially collected and proved to be genuine and integrity.

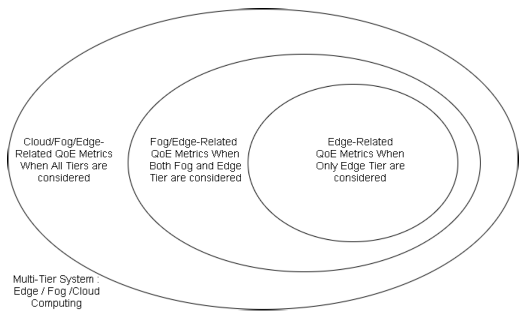
Cloud/Fog/Edge-Related QoE Metrics When All Tiers are considered
Fog/Edge-Related QoE Metrics When Both Fog and Edge Tier are considered
Edge-Related QoE Metrics When Only Edge Tier are considered
Multi-Tier System : Edge / Fog /Cloud Computing

FIGURE 8. QoE domains.

B. QoE DOMAINS FOR MULTI-TIER QoE METRICS AND MANAGEMENT

In multi-tier architecture, QoE metrics and QoE management require a large number of information from different computing nodes, location, and tiers. It is known that acquiring data from multiple nodes locating at different places and tiers can result in deterioration of communication delays. Since each tier has its own QoE Cause Factors, estimating QoE in local domain can faster QoE-aware decisions and improve QoE management in terms of resource management. QoE metric in small domain e.g., Edge tier can be extended for larger domain such as Edge+Fog tier and Edge+Fog+Cloud tier respectively. A set of QoE domains is depicted in Figure 8.

In Figure 8, there are three QoE domains when multi-tier system is considered. Examples of QoE domains are Edge domain, Fog/Edge domain and Cloud/Fog/Edge domain. QoE metrics, estimated by cause factors related to its tier, can be used for local resource management for multiple users and applications. This approach can significantly improve QoE by achieving faster response time for controlling resources to match up with real-time user perception. In our opinion, incorporating three domains for QoE management is a non-trival multi-dimensional problem, which requires further studies.

IX. CONCLUSION

Driven by QoE requirements, many studies focused more on QoE management in IoT system. Still, it is not obvious what critical components are necessary and what resource management techniques are typically used in QoE-Driven architecture. Further studies on these hot topics should be conducted. In this paper, we provide a comprehensive review of vital components in QoE-Driven architecture e.g., QoE Cause Factors, QoE measurement and indicators, QoE Prediction, QoE optimization and control. Prior studies related to these components were reviewed by focusing on recent ML approaches. Then, QoE metrics were modeled from these cause factors. We summarized QoE metrics and QoE optimization objectives for various kind of problems e.g., prediction model, optimization and control, resource management respectively. For resource management, we categorized into three main problems for emerging IoT architectures : QoE-aware offloading problems, QoE-aware placement problems and QoE-aware data caching problems. The reviews are mostly based on ML approaches, which was not considered in prior surveys. Since ML-based approaches have been widely used to predict QoE and solve resource allocation problems, this promising approach requires a novel concept of AI-based layers for managing QoE. In our discussion, these layers, e.g., the Training Layer, the Validation Layer and the Prediction Layer, can be used as the guideline for possible future works, pending further investigation.

ACKNOWLEDGMENT

The authors are grateful for Prof. Dr. S. Serapin from the Professional Authorship Center, National Science and Technology Development Agency, for fruitful discussion in manuscript editing.

REFERENCES

[1] M. Yang, S. Wang, R. N. Calheiros, and F. Yang, "Survey on QoE assessment approach for network service," *IEEE Access*, vol. 6, pp. 48374–48390, 2018.
[2] A. Itu-T 1: Recommendation P. 10/G. 100, *New Appendix I–Definition of Quality of Experience (QoE)*, Telecommunication Standardization Sector of ITU, document 100:2007, 2006.
[3] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M. N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M. C. Larabi, and B. Lawlor, "Qualinet white paper on definitions of quality of experience," *Eur. Netw. Quality Exp. Multimedia Syst. Services (COST Action IC)*, vol. 3, no. 2012, pp. 1–24, 2012.
[4] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for QoE management of multimedia services," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 2s, pp. 1–29, Apr. 2018.
[5] Z. Tian, L. Zhao, L. Nie, P. Chen, and S. Chen, "Deeplive: QoE optimization for live video streaming through deep reinforcement learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 827–831.
[6] H. D. Moura, D. F. Macedo, and M. A. M. Vieira, "Wireless control using reinforcement learning for practical web QoE," *Comput. Commun.*, vol. 154, pp. 331–346, Mar. 2020.
[7] L. Wang and D. T. Delaney, "QoE oriented cognitive network based on machine learning and SDN," in *Proc. IEEE 11th Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2019, pp. 678–681.
[8] K. Fizza, A. Banerjee, K. Mitra, P. P. Jayaraman, R. Ranjan, P. Patel, and D. Georgakopoulos, "QoE in IoT: A vision, survey and future directions," *Discover Internet Things*, vol. 1, no. 1, pp. 1–14, Dec. 2021.
[9] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "QoE management of multimedia streaming services in future networks: A tutorial and survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 526–565, 1st Quart., 2019.
[10] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102781.
[11] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102225.
[12] S. K. Uz Zaman, A. I. Jehangiri, T. Maqsood, Z. Ahmad, A. I. Umar, J. Shuja, E. Alanazi, and W. Alasmary, "Mobility-aware computational offloading in mobile edge networks: A survey," *Cluster Comput.*, vol. 24, pp. 1–22, Dec. 2021.
[13] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.
[14] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–35, May 2021.
[15] Z. M. Nayeri, T. Ghafarian, and B. Javadi, "Application placement in fog computing with AI approach: Taxonomy and a state of the art survey," *J. Netw. Comput. Appl.*, vol. 185, Jul. 2021, Art. no. 103078.

[16] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 181, May 2021, Art. no. 103005.
[17] Y. Shen, T. Zhang, Y. Wang, H. Wang, and X. Jiang, "Microthings: A generic IoT architecture for flexible data aggregation and scalable service cooperation," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 86–93, Sep. 2017.
[18] S. Krco, B. Pokric, and F. Carrez, "Designing IoT architecture(s): A European perspective," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 79–84.
[19] W. Lv, F. Meng, C. Zhang, Y. Lv, N. Cao, and J. Jiang, "A general architecture of IoT system," in *Proc. 7 IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, vol. 1, Jul. 2017, pp. 659–664.
[20] I. Lee, "The Internet of Things for enterprises: An ecosystem, architecture, and IoT service business model," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100078.
[21] C.-L. Zhong, Z. Zhu, and R.-G. Huang, "Study on the IoT architecture and gateway technology," in *Proc. 14th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci. (DCABES)*, Aug. 2015, pp. 196–199.
[22] C. Sarkar, S. N. A. U. Nambi, R. V. Prasad, A. Rahim, R. Neisse, and G. Baldini, "DIAT: A scalable distributed architecture for IoT," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 230–239, Jun. 2015.
[23] Fiware Foundation. *The Opensource Platform for our Smart Digital Future—FiWARE*. Accessed: Jan. 12, 2020. [Online]. Available: https://www.fiware.org/
[24] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IoT," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber, Phys. Social Comput.*, Oct. 2011, pp. 717–720.
[25] J. Guth, U. Breitenbücher, M. Falkenthal, P. Fremantle, O. Kopp, F. Leymann, and L. Reinfurt, "A detailed analysis of IoT platform architectures: Concepts, similarities, and differences," in *Internet Everything*. Singapore: Springer, 2018, pp. 81–101. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-5861-5_4
[26] Eclipse Foundation. (2016). *The Three Software Stacks Required for IoT Architectures*. Accessed: Nov. 18, 2020. [Online]. Available: https://iot.eclipse.org/
[27] Intel. *The Intel IoT Platform Architecture Specification White Paper Internet of Things (IoT)*. Accessed: Nov. 18, 2020. [Online]. Available: https://www.scribd.com/document/386515116/Iot-Platform-Reference-Architecture-Paper
[28] Microsoft AzureIoT IoT. *Microsoft Azure IoT Reference Architecture*. Accessed: Nov. 18, 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot
[29] J. Guth, U. Breitenbucher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of IoT platform architectures: A field study based on a reference architecture," in *Proc. Cloudification Internet Things (CIoT)*, Nov. 2016, pp. 1–6.
[30] D. A. Menascé, H. Ruan, and H. Gomaa, "QoS management in service-oriented architectures," *Perform. Eval.*, vol. 64, nos. 7–8, pp. 646–663, Aug. 2007.
[31] I. Udoh and G. Kotonya, "A dynamic QoS negotiation framework for IoT services," in *Proc. IEEE Global Conf. Internet Things (GCIoT)*, Dec. 2019, pp. 1–7.
[32] F. Li and S. Clarke, "A context-based strategy for SLA negotiation in the IoT environment," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2019, pp. 208–213.
[33] E. Mingozzi, G. Tanganelli, and C. Vallati, "A framework for QoS negotiation in things-as-a-service oriented architectures," in *Proc. 4th Int. Conf. Wireless Commun., Veh. Technol., Inf. Theory Aerosp. Electron. Syst. (VITAE)*, May 2014, pp. 1–5.
[34] X. Zheng, P. Martin, K. Brohman, and L. D. Xu, "Cloud service negotiation in Internet of Things environment: A mixed approach," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1506–1515, May 2014.
[35] D. A. Menasce and V. Dubey, "Utility-based QoS brokering in service oriented architectures," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2007, pp. 422–430.
[36] OpenIoT Consortium. *D4.6. Quality of Service (QoS) for IoT services*. Accessed: Nov. 29, 2020. [Online]. Available: https://cordis.europa.eu/docs/projects/cnect/5/287305/080/deliverables/001-OpenIoTD46 Draft.pdf
[37] L. Li, S. Li, and S. Zhao, "QoS-aware scheduling of services-oriented Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1497–1505, May 2014.
[38] Y. Banouar, S. Reddad, C. Diop, C. Chassot, and A. Zyane, "Monitoring solution for autonomic middleware-level QoS management within IoT systems," in *Proc. IEEE/ACS 12th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2015, pp. 1–8.
[39] G. White, A. Palade, and S. Clarke, "Qos prediction for reliable service composition in IoT," in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2017, pp. 149–160.
[40] G. White, A. Palade, C. Cabrera, and S. Clarke, "IoTPredict: Collaborative QoS prediction in IoT," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2018, pp. 1–10.
[41] B. K. J. Al-Shammari, N. Al-Aboody, and H. S. Al-Raweshidy, "IoT traffic management and integration in the QoS supported network," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 352–370, Feb. 2017.
[42] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, Feb. 2017.
[43] J. Seeger, A. Broring, M.-O. Pahl, and E. Sakic, "Rule-based translation of application-level QoS constraints into SDN configurations for the IoT," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2019, pp. 432–437.
[44] W. Wang, Q. Qi, X. Gong, Y. Hu, and X. Que, "Autonomic QoS management mechanism in software defined network," *China Commun.*, vol. 11, no. 7, pp. 13–23, Jul. 2014.
[45] V. Issarny, G. Bouloukakis, N. Georgantas, and B. Billet, "Revisiting service-oriented architecture for the IoT: A middleware perspective," in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2016, pp. 3–17.
[46] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of experience for streaming services: Measurements, challenges and insights," *IEEE Access*, vol. 8, pp. 13341–13361, 2020.
[47] D. Pal, V. Vanijja, and V. Varadarajan, "Quality provisioning in the Internet of Things era: Current state and future directions," in *Proc. 10th Int. Conf. Adv. Inf. Technol. (IAIT)*, 2018, pp. 1–7.
[48] A. Floris and L. Atzori, "Managing the quality of experience in the multimedia Internet of Things: A layered-based approach," *Sensors*, vol. 16, no. 12, p. 2057, 2016.
[49] C. Song, W. Xu, T. Wu, S. Yu, P. Zeng, and N. Zhang, "QoE-driven edge caching in vehicle networks based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5286–5295, Jun. 2021.
[50] M. S. Mushtaq, B. Augustin, and A. Mellouk, "Empirical study based on machine learning approach to assess the QoS/QoE correlation," in *Proc. 17th Eur. Conf. Netw. Opt. Commun.*, Jun. 2012, pp. 1–7.
[51] F. Laiche, A. B. Letaifa, I. Elloumi, and T. Aguili, "When machine learning algorithms meet user engagement parameters to predict video QoE," *Wireless Pers. Commun.*, vol. 116, no. 3, pp. 2723–2741, Feb. 2021.
[52] X. Tao, Y. Duan, M. Xu, Z. Meng, and J. Lu, "Learning QoE of mobile video transmission with deep neural network: A data-driven approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1337–1348, Jun. 2019.
[53] T. Yue, H. Wang, S. Cheng, and J. Shao, "Deep learning based QoE evaluation for internet video," *Neurocomputing*, vol. 386, pp. 179–190, Apr. 2020.
[54] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 3, pp. 781–796, Jul./Sep. 2020.
[55] R. Sun, Y. Wang, N. Cheng, L. Lyu, S. Zhang, H. Zhou, and X. Shen, "QoE-driven transmission-aware cache placement and cooperative beamforming design in cloud-RANs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 636–650, Jan. 2019.
[56] Y. Wang, *Survey of Objective Video Quality Measurements*, vol. 1748. Hopkinton, MA, USA: EMC Corporation, 2006, p. 39.
[57] J. Liu, X. Tao, and J. Lu, "QoE-oriented rate adaptation for DASH with enhanced deep Q-learning," *IEEE Access*, vol. 7, pp. 8454–8469, 2019.
[58] A. Lekharu, K. Y. Moulii, A. Sur, and A. Sarkar, "Deep learning based prediction model for adaptive video streaming," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 152–159.
[59] *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for end-to-end Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, REC. ITU-TP 862, 2001.
[60] E. Liotou, D. Tsolkas, and N. Passas, "A roadmap on QoE metrics and models," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.
[61] A. Ben Letaifa, "Adaptive QoE monitoring architecture in SDN networks: Video streaming services case," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1383–1388.
[62] V. Menkovski, G. Exarchakos, and A. Liotta, "Machine learning approach for quality of experience aware networks," in *Proc. Int. Conf. Intell. Netw. Collaborative Syst.*, Nov. 2010, pp. 461–466.

[63] P. Charonyktakis, M. Plakia, I. Tsamardinos, and M. Papadopouli, "On user-centric modular QoE prediction for VoIP based on machine-learning algorithms," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1443–1456, Jun. 2015.

[64] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. P. Mok, and A. Dhamdhere, "QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 251–260.

[65] A. Akbar, M. Ibrar, M. A. Jan, A. K. Bashir, and L. Wang, "SDN-enabled adaptive and reliable communication in IoT-fog environment using machine learning and multiobjective optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3057–3065, Nov. 2020.

[66] H. Wang, K. Wu, J. Wang, and G. Tang, "Rldish: Edge-assisted QoE optimization of HTTP live streaming with reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2020, pp. 706–715.

[67] D. Minovski, C. Ahlund, K. Mitra, and R. Zhohov, "Defining quality of experience for the Internet of Things," *IT Prof.*, vol. 22, no. 5, pp. 62–70, Sep. 2020.

[68] M. Shen, J. Zhang, K. Xu, L. Zhu, J. Liu, and X. Du, "DeepQoE: Real-time measurement of video QoE from encrypted traffic with deep learning," in *Proc. IEEE/ACM 28th Int. Symp. Quality Service (IWQoS)*, Jun. 2020, pp. 1–10.

[69] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable OpenFlow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107346–107379, 2019.

[70] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[71] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "A deep probabilistic control machinery for auto-configuration of WiFi link parameters," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8330–8340, Dec. 2020.

[72] M. Kanj, V. Savaux, and M. Le Guen, "A tutorial on NB-IoT physical layer design," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2408–2446, 4th Quart., 2020.

[73] J. Petäjäjärvi, K. Mikhaylov, R. Yasmin, M. Hämäläinen, and J. Iinatti, "Evaluation of LoRa LPWAN technology for indoor remote health and wellbeing monitoring," *Int. J. Wireless Inf. Netw.*, vol. 24, no. 2, pp. 153–165, Jun. 2017.

[74] H. Mousavi, I. S. Amiri, M. A. Mostafavi, and C. Y. Choon, "LTE physical layer: Performance analysis and evaluation," *Appl. Comput. Informat.*, vol. 15, no. 1, pp. 34–44, Jan. 2019.

[75] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives," *Multimedia Syst.*, vol. 22, no. 2, pp. 213–227, Mar. 2016.

[76] M. Suryanegara, D. A. Prasetyo, F. Andriyanto, and N. Hayati, "A 5-step framework for measuring the quality of experience (QoE) of Internet of Things IoT) services," *IEEE Access*, vol. 7, pp. 175779–175792, 2019.

[77] V. Vasilev, J. Leguay, S. Paris, L. Maggi, and M. Debbah, "Predicting QoE factors with machine learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[78] B. Hou and J. Zhang, "QoE estimation of DASH-based mobile video application using deep reinforcement learning," in *Algorithms and Architectures for Parallel Processing*, M. Qiu, Ed. Cham, Switzerland: Springer, 2020, pp. 633–645.

[79] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[80] O.-K. Shahryari, H. Pedram, V. Khajehvand, and M. D. TakhtFooladi, "Energy and task completion time trade-off for task offloading in fog-enabled IoT networks," *Pervas. Mobile Comput.*, vol. 74, Jul. 2021, Art. no. 101395.

[81] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.

[82] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in fog computing environments," *J. Parallel Distrib. Comput.*, vol. 132, pp. 190–203, Oct. 2019.

[83] M. Carvalho and D. F. Macedo, "QoE-aware container scheduler for co-located cloud environments," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2021, pp. 286–294.

[84] L. Dinh-Xuan, M. Seufert, F. Wamser, and P. Tran-Gia, "QoE aware placement of content in edge networks on the example of a photo album cloud service," in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 443–448.

[85] W.-Y. Chen, P.-Y. Chou, C.-Y. Wang, R.-H. Hwang, and W.-T. Chen, "Live video streaming with joint user association and caching placement in mobile edge computing," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2020, pp. 796–801.

[86] J. Luo, F. R. Yu, Q. Chen, and L. Tang, "Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1577–1592, Mar. 2020.

[87] P.-Y. Chou, W.-Y. Chen, C.-Y. Wang, R.-H. Hwang, and W.-T. Chen, "Deep reinforcement learning for MEC streaming with joint user association and resource management," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.

[88] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 1126–1165, 2nd Quart., 2015.

[89] T. Abar, A. Ben Letaifa, and S. El Asmi, "Machine learning based QoE prediction in SDN networks," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1395–1400.

[90] M. Lopez-Martin, B. Carro, J. Lloret, S. Egea, and A. Sanchez-Esguevillas, "Deep learning model for multimedia quality of experience prediction based on network flow packets," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 110–117, Sep. 2018.

[91] X. Huang, K. Xie, S. Leng, T. Yuan, and M. Ma, "Improving quality of experience in multimedia Internet of Things leveraging machine learning on big data," *Future Gener. Comput. Syst.*, vol. 86, pp. 1413–1423, Sep. 2018.

[92] G. Miranda, D. F. Macedo, and J. M. Marquez-Barja, "A QoE inference method for DASH video using ICMP probing," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2020, pp. 1–5.

[93] G. Miranda, D. F. Macedo, and J. M. Marquez-Barja, "Estimating video on demand QoE from network QoS through ICMP probes," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1890–1902, Jun. 2022.

[94] G. Miranda, E. Municio, J. M. Marquez-Barja, and D. F. Macedo, "Machine learning-based end-to-end QoE monitoring using active network probing," in *Proc. 25th Conf. Innov. Clouds, Internet Netw. (ICIN)*, Mar. 2022, pp. 40–47.

[95] R. Ul Mustafa, D. Moura, and C. E. Rothenberg, "Machine learning approach to estimate video QoE of encrypted DASH traffic in 5G networks," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, Jul. 2021, pp. 586–589.

[96] S. C. Madanapalli, A. Mathai, H. H. Gharakheili, and V. Sivaraman, "ReCLive: Real-time classification and QoE inference of live video streaming services," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Jun. 2021, pp. 1–7.

[97] S. C. Madanapalli, A. Mathai, H. H. Gharakheili, and V. Sivaraman, "Modeling live video streaming: Real-time classification, QoE inference, and field evaluation," 2021, *arXiv:2112.02637*.

[98] F. Laiche, A. Ben Letaifa, and T. Aguili, "QoE-aware traffic monitoring based on user behavior in video streaming services," *Concurrency Comput., Pract. Exp.*, Nov. 2021, Art. no. e6678, doi: 10.1002/cpe.6678.

[99] P. Zhou, Y. Xie, B. Niu, L. Pu, Z. Xu, H. Jiang, and H. Huang, "QoE-aware 3D video streaming via deep reinforcement learning in software defined networking enabled mobile edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 419–433, Jan. 2021.

[100] M. Tabassum, K. M. Tikoicina, and E. Huda, "Comparative analysis of queuing algorithms and QoS effects on the IoT networks traffic," in *Proc. 8th IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, Nov. 2018, pp. 88–92.

[101] P. Orosz, P. Varga, G. Soos, and C. Hegedus, "QoS guarantees for industrial IoT applications over LTE—A feasibility study," in *Proc. IEEE Int. Conf. Ind. Cyber Phys. Syst. (ICPS)*, May 2019, pp. 667–672.

[102] S. Smiri, A. Boushaba, R. Ben Abbou, and A. Zahi, "Geographic and topology based routing protocols in vehicular ad-hoc networks: Performance evaluation and QoS analysis," in *Proc. Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, Apr. 2018, pp. 1–8.

[103] R. Basir, S. B. Qaisar, M. Ali, M. Naeem, K. C. Joshi, and J. Rodriguez, "Latency-aware resource allocation in green fog networks for industrial IoT applications," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[104] C. Desogus, M. Anedda, M. Murroni, D. D. Giusto, and G.-M. Muntean, ''ReMIoT: Reputation-based network selection in multimedia IoT,'' in *Proc. IEEE Broadcast Symp. (BTS)*, Oct. 2019, pp. 1–6.
[105] S. Katre, A. Goswami, P. Mishra, J. Bapat, and D. Das, ''Impact of variable MTU size of voice packet to reduce packet loss in bandwidth constraint military network,'' in *Proc. IEEE 5th Int. Conf. Converg. Technol. (I2CT)*, Mar. 2019, pp. 1–5.
[106] A. Mukhopadhyay, ''QoS based telemedicine technologies for rural healthcare emergencies,'' in *Proc. IEEE Global Humanitarian Technol. Conf. (GHTC)*, Oct. 2017, pp. 1–7.
[107] M. Asif-Ur-Rahman, F. Afsana, M. Mahmud, M. S. Kaiser, M. R. Ahmed, O. Kaiwartya, and A. James-Taylor, ''Toward a heterogeneous mist, fog, and cloud-based framework for the Internet of Healthcare Things,'' *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4049–4062, Jun. 2018.
[108] I. U. Rehman, M. M. Nasralla, A. Ali, and N. Philip, ''Small cell-based ambulance scenario for medical video streaming: A 5G-health use case,'' in *Proc. 15th Int. Conf. Smart Cities, Improving Quality Life Using ICT IoT (HONET-ICT)*, Oct. 2018, pp. 29–32.
[109] M. Z. Hasan and H. Al-Rizzo, ''Optimization of sensor deployment for industrial Internet of Things using a multiswarm algorithm,'' *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10344–10362, Dec. 2019.
[110] S. Katre, A. Goswami, P. Mishra, J. Bapat, and D. Das, ''Improved connectivity using differential priority assignments in military network,'' in *Proc. 5th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Dec. 2018, pp. 381–386.
[111] K. M. Besher, S. Beitelspacher, J. I. Nieto-Hipolito, and M. Z. Ali, ''Sensor initiated healthcare packet priority in congested IoT networks,'' *IEEE Sensors J.*, vol. 21, no. 10, pp. 11704–11711, May 2020.
[112] M. T. Abbas and W.-C. Song, ''Infrastructure-assisted hybrid road-aware routing and QoS provisioning in VANETs,'' in *Proc. 19th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2017, pp. 370–373.
[113] X. Zhang and Q. Zhu, ''Hierarchical-caching based statistical QoS provisioning over 5G multimedia big-data mobile wireless networks,'' in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 480–485.
[114] N. Abbas, F. Yu, and U. Majeed, ''Reliability and end-to-end delay evaluation of outdoor and indoor environments for wireless multimedia sensor networks,'' in *Proc. 2nd IEEE Adv. Inf. Manag., Communicates, Electron. Autom. Control Conf. (IMCEC)*, May 2018, pp. 764–768.
[115] R. S. Auliva, R.-K. Sheu, D. Liang, and W.-J. Wang, ''IIoT Testbed: A DDS-based emulation tool for industrial IoT applications,'' in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Jun. 2018, pp. 1–4.
[116] A. Zhang, H. Sun, and Y. Zhan, ''Service allocation based on QoS evaluation in military organization cloud cooperation,'' *J. Syst. Eng. Electron.*, vol. 27, no. 2, pp. 386–394, Apr. 2016.
[117] Z. Yang and Z. He, ''Application of improved genetic algorithm in vehicle networked cloud data platform,'' in *Proc. Int. Conf. Intell. Transp., Big Data Smart City (ICITBS)*, Jan. 2018, pp. 5–8.
[118] A. H. Sodhro, M. S. Obaidat, Q. H. Abbasi, P. Pace, S. Pirbhulal, A.-U.-H. Yasar, G. Fortino, M. A. Imran, and M. Qaraqe, ''Quality of service optimization in an IoT-driven intelligent transportation system,'' *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 10–17, Dec. 2019.
[119] M. Kuzlu and M. Pipattanasomporn, ''Assessment of communication technologies and network requirements for different smart grid applications,'' in *Proc. IEEE PES Innov. Smart Grid Technol. Conf. (ISGT)*, Feb. 2013, pp. 1–6.
[120] M. Kuzlu, M. Pipattanasomporn, and M. Rahman, ''Communication network requirements for major smart grid applications in HAN, NAN and WAN,'' *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.
[121] L. Skorin-Kapov and M. Matijasevic, ''Analysis of QoS requirements for e-Health services and mapping to evolved packet system QoS classes,'' *Int. J. Telemed. Appl.*, vol. 2010, pp. 1–18, Jan. 2010.
[122] J. Mocnej, A. Pekar, W. K. G. Seah, and I. Zolotova, ''Network traffic characteristics of the IoT application use cases,'' School Eng. Comput. Sci., Victoria Univ. Wellington, Wellington, New Zealand, Tech. Rep., 2018. [Online]. Available: https://books.google.co.th/books?id=yfV_zQEACAAJ
[123] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, ''Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture,'' *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017.
[124] P. Papadimitratos, A. D. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, ''Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation,'' *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 84–95, Nov. 2009.
[125] G. Baranwal, R. Yadav, and D. P. Vidyarthi, ''QoE aware IoT application placement in fog computing using modified-TOPSIS,'' *Mobile Netw. Appl.*, vol. 25, pp. 1–17, Oct. 2020.
[126] A. Tsipis, K. Oikonomou, V. Komianos, and I. Stavrakakis, ''QoE-aware rendering service allocation in fog-assisted cloud gaming environments,'' in *Proc. 5th South-East Eur. Design Autom., Comput. Eng., Comput. Netw. Social Media Conf. (SEEDA-CECNSM)*, Sep. 2020, pp. 1–8.
[127] J. Luo, X. Deng, H. Zhang, and H. Qi, ''QoE-driven computation offloading for edge computing,'' *J. Syst. Archit.*, vol. 97, pp. 34–39, Aug. 2019.
[128] S.-T. Hong and H. Kim, ''QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds,'' in *Proc. 13th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2016, pp. 1–9.
[129] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang, ''Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things,'' *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9255–9265, Oct. 2020.
[130] X. He, H. Lu, Y. Mao, and K. Wang, ''QoE-driven task offloading with deep reinforcement learning in edge intelligent IoV,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.
[131] L. Caviglione, M. Gaggero, M. Paolucci, and R. Ronco, ''Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters,'' *Soft Comput.*, vol. 25, pp. 1–20, Oct. 2020.
[132] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, ''QoE-aware user allocation in edge computing systems with dynamic QoS,'' *Future Gener. Comput. Syst.*, vol. 112, pp. 684–694, Nov. 2020.
[133] X. He, K. Wang, and W. Xu, ''QoE-driven content-centric caching with deep reinforcement learning in edge-enabled IoT,'' *IEEE Comput. Intell. Mag.*, vol. 14, no. 4, pp. 12–20, Nov. 2019.
[134] Y. Wang, C. Feng, T. Zhang, Y. Liu, and A. Nallanathan, ''QoE based network deployment and caching placement for cache-enabling UAV networks,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[135] W. Wu, Y. Gao, T. Zhou, Y. Jia, H. Zhang, T. Wei, and Y. Sun, ''Deep reinforcement learning-based video quality selection and radio bearer control for mobile edge computing supported short video applications,'' *IEEE Access*, vol. 7, pp. 181740–181749, 2019.
[136] Y. Han, D. Guo, W. Cai, X. Wang, and V. Leung, ''Virtual machine placement optimization in mobile cloud gaming through QoE-oriented resource competition,'' *IEEE Trans. Cloud Comput.*, early access, Jun. 12, 2020, doi: 10.1109/TCC.2020.3002023.
[137] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, ''QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks,'' *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, Oct. 2013.
[138] B. Li, W. Dong, G. Guan, J. Zhang, T. Gu, J. Bu, and Y. Gao, ''Queec: QoE-aware edge computing for IoT devices under dynamic workloads,'' *ACM Trans. Sensor Netw.*, vol. 17, no. 3, pp. 1–23, Jun. 2021.
[139] X.-Q. Pham, T. Huynh-The, and D.-S. Kim, ''A QoE-based optimization approach to computation offloading in vehicle-assisted multi-access edge computing,'' in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2021, pp. 131–133.
[140] J. Park and K. Chung, ''Resource prediction-based edge collaboration scheme for improving QoE,'' *Sensors*, vol. 21, no. 24, p. 8500, Dec. 2021.
[141] X. Dai, K. Ota, and M. Dong, ''Deep reinforcement learning based multi-access edge computing schedule for internet of vehicle,'' 2022, *arXiv:2202.08972*.
[142] F. Poltronieri, M. Tortonesi, C. Stefanelli, and N. Suri, ''Reinforcement learning for value-based placement of fog services,'' in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2021, pp. 466–472.
[143] J. Zhang, S. Chen, X. Wang, and Y. Zhu, ''DeepReserve: Dynamic edge server reservation for connected vehicles with deep reinforcement learning,'' in *Proc. INFOCOM IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.
[144] I. Alqerm and J. Pan, ''DeepEdge: A new QoE-based resource allocation framework using deep reinforcement learning for future heterogeneous edge-IoT applications,'' *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 3942–3954, Dec. 2021.
[145] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, ''Intelligent resource allocation in dynamic fog computing environments,'' in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–7.
[146] Y. Liu, Q. He, D. Zheng, X. Xia, F. Chen, and B. Zhang, ''Data caching optimization in the edge computing environment,'' *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2074–2085, Jul. 2020.
[147] X. Xu, C. Feng, S. Shan, T. Zhang, and J. Loo, ''Proactive edge caching in content-centric networks with massive dynamic content requests,'' *IEEE Access*, vol. 8, pp. 59906–59921, 2020.

[148] H. Tian, X. Xu, T. Lin, Y. Cheng, C. Qian, L. Ren, and M. Bilal, "DIMA: Distributed cooperative microservice caching for Internet of Things in edge computing by deep reinforcement learning," *World Wide Web*, pp. 1–24, Aug. 2021.
[149] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proc. INFOCOM IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 2499–2508.
[150] B. Guo, X. Zhang, Q. Sheng, and H. Yang, "Dueling deep-Q-network based delay-aware cache update policy for mobile users in fog radio access networks," *IEEE Access*, vol. 8, pp. 7131–7141, 2020.
[151] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2018.
[152] L. Lu, Y. Jiang, M. Bennis, Z. Ding, F.-C. Zheng, and X. You, "Distributed edge caching via reinforcement learning in fog radio access networks," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–6.
[153] B. Dai and W. Yu, "Joint user association and content placement for cache-enabled wireless access networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 3521–3525.
[154] X. He, K. Wang, H. Lu, W. Xu, and S. Guo, "Edge QoE: Intelligent big data caching via deep reinforcement learning," *IEEE Netw.*, vol. 34, no. 4, pp. 8–13, Jul. 2020.
[155] Y. Liu, Y. Han, A. Zhang, X. Xia, F. Chen, M. Zhang, and Q. He, "QoE-aware data caching optimization with budget in edge computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Sep. 2021, pp. 324–334.
[156] J. Li, B. Zhao, and D. Liu, "DMPP: Differentiable multi-pruner and predictor for neural network pruning," *Neural Netw.*, vol. 147, pp. 103–112, Mar. 2022.
[157] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[158] S. Siddegowda, M. Fournarakis, M. Nagel, T. Blankevoort, C. Patel, and A. Khobare, "Neural network quantization with AI model efficiency toolkit (AIMET)," 2022, *arXiv:2201.08442*.
[159] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digit. Commun. Netw.*, vol. 8, no. 1, pp. 1–17, Feb. 2022.
[160] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.
[161] X. Hu and H. Wen, "Research on model compression for embedded platform through quantization and pruning," *J. Phys., Conf.*, vol. 2078, no. 1, Nov. 2021, Art. no. 012047.
[162] P. Singh, *Deploy Machine Learning Models to Production*. Cham, Switzerland: Springer, 2021.
[163] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," 2018, *arXiv:1811.12808*.
[164] A. Huet, A. Saverimoutou, Z. B. Houidi, H. Shi, S. Cai, J. Xu, B. Mathieu, and D. Rossi, "Deployable models for approximating web QoE metrics from encrypted traffic," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3336–3352, Sep. 2021.
[165] E. Breck, N. Polyzotis, S. Roy, S. Whang, and M. Zinkevich, "Data validation for machine learning," in *Proc. 2nd SysML Conf. MLSys*, 2019, pp. 1–14.
[166] E. Caveness, G. C. P. Suganthan, Z. Peng, N. Polyzotis, S. Roy, and M. Zinkevich, "TensorFlow data validation: Data analysis and validation in continuous ML pipelines," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 2793–2796.

BOONYARITH SAOVAPAKHIRAN received the B.Eng. degree in electrical engineering from Mahidol University, Thailand, in 1998, the M.S. degree in electrical engineering from the University of Southern California, USA, in 2009, and the Ph.D. degree in computer engineering from North Carolina State University, USA, in 2013.

He is currently a Researcher and a member of the Internet Innovation Research Team, National Electronics and Computer Technology Center (NECTEC), Thailand. His research interests include network design and optimization, the Internet of Things, data mining, and artificial intelligence.

WIBHADA NARUEPHIPHAT received the B.S. and M.S. degrees in telecommunication engineering from the Suranaree University of Technology, Nakhon Ratchasima, Thailand, in 2008.

Since 2008, she has been a Research Assistant with the Internet Innovation Team, National Electronics and Computer Technology Center, Pathum Thani, Thailand. Her research interests include network technology and system performance. Her current research works are focused on adaptive sampling rate for vehicle tracking and object detection for smart parking in smart city project.

CHALERMPOL CHARNSRIPINYO received the B.S. degree in computer science from Thammasat University, Thailand, in 1992, the M.S. degree in electrical engineering from George Washington University, USA, in 1998, and the Ph.D. degree in telecommunications from the University of Pittsburgh, USA, in 2003.

He is currently a Senior Research Specialist and the Head of the Internet Innovation Research Team, National Electronics and Computer Technology Center (NECTEC), under the National Science and Technology Development Agency (NSTDA), Thailand. His current research interests include network design and optimization, network security and management, network virtualization, the Internet of Things, cloud, fog and edge computing, and data service platform.

SEBNEM BAYDERE (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from METU, Ankara, Turkey, in 1984 and 1987, respectively, and the Ph.D. degree in computer science from the University College London (UCL), U.K., in 1991.

She is currently a Full Professor with the Department of Computer Engineering, Yeditepe University, Istanbul, Turkey. Her current research interests include QoS architectures, the Internet of Things, fog computing, wireless sensor networks, and distributed algorithms.

SUAT ÖZDEMIR (Member, IEEE) received the M.Sc. degree in computer science from Syracuse University, Syracuse, NY, USA, in August 2001, and the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA, in December 2006.

He is currently with the Department of Computer Engineering, Hacettepe University, Ankara, Turkey. His current research interests include the Internet of Things, data analytics, artificial intelligence, and network security.