An Optimized IoT-Enabled Big Data Analytics Architecture for Edge–Cloud Computing

Muhammad Babar, Mian Ahmad Jan, *Senior Member, IEEE*, Xiangjian He, *Senior Member, IEEE*, Muhammad Usman Tariq, Spyridon Mastorakis, *Member, IEEE*, and Ryan Alturki, *Senior Member, IEEE*

***Abstract*—The awareness of edge computing is attaining eminence and is largely acknowledged with the rise of the Internet of Things (IoT). Edge-enabled solutions offer efficient computing and control at the network edge to resolve the scalability and latency-related concerns. Though, it comes to be challenging for edge computing to tackle diverse applications of IoT as they produce massive heterogeneous data. The IoT-enabled frameworks for Big Data analytics face numerous challenges in their existing structural design, for instance, the high volume of data storage and processing, data heterogeneity, and processing time among others. Moreover, the existing proposals lack effective parallel data loading and robust mechanisms for handling communication overhead. To address these challenges, we propose an optimized IoT-enabled big data analytics architecture for edge–cloud computing using machine learning. In the proposed scheme, an edge intelligence module is introduced to process and store the big data efficiently at the edges of the network with the integration of cloud technology. The proposed scheme is composed of two layers: 1) IoT–edge and 2) cloud processing. The data injection and storage is carried out with an optimized MapReduce parallel algorithm. An optimized yet another resource negotiator (YARN) is used for efficiently managing the cluster. The proposed data design is experimentally simulated with an authentic data set using Apache Spark. The comparative analysis is decorated with the existing proposals and traditional mechanisms. The results justify the efficiency of our proposed work.**

***Index Terms*—Backpropagation (BP) neural network, big data analytics, edge computing, Internet of Things (IoT), machine learning (ML), yet another resource negotiator (YARN).**

Manuscript received 18 December 2021; revised 4 February 2022; accepted 26 February 2022. Date of publication 14 March 2022; date of current version 20 February 2023. This work was supported in part by NIH (NIGMS) under Grant P20GM109090; in part by NSF under Award CNS-2016714, Award CNS-2104700, and Award CBET-2124918; in part by the Nebraska University Collaboration Initiative; and in part by the Nebraska Tobacco Settlement Biomedical Research Development Funds. *(Corresponding authors: Mian Ahmad Jan; Xiangjian He.)*

Muhammad Babar is with the Department of Computer Science, Allama Iqbal Open University, Islamabad 44000, Pakistan.

Mian Ahmad Jan is with the Department of Computer Science, Abdul Wali Khan University Mardan, Mardan 24200, Pakistan (e-mail: mianjan@awkum.edu.pk).

Xiangjian He is with the Global Big Data Technologies Center, School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: xiangjian.he@uts.edu.au).

Muhammad Usman Tariq is with the Department of Business and Computing, Abu Dhabi University, Abu Dhabi, UAE.

Spyridon Mastorakis is with the College of Information Science and Technology, University of Nebraska Omaha, Omaha, NE 68182 USA.

Ryan Alturki is with the Department of Information Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Digital Object Identifier 10.1109/JIOT.2022.3157552

I. Introduction

THE Internet of Things (IoT) consists of software utilities along with hardware. The hardware involves of sensor-embedded devices connected with each other. The software utilities, on the other hand, comprise data storage and analytics programs that provide assistance in presenting information to the users [1]. The idea of edge computing is gaining significant attention with the rise of IoT [2]–[4]. Edge-enabled solutions offer efficient computing and provide the control near the edge of the network to resolve the scalability and latency issues [5], [6]. The IoT applications generate massive data, i.e., big data,[1] which require effective aggregation, intelligent processing, and robust analysis to realize optimum outcomes for decision making [7], [8]. In an IoT-enabled edge computing environment, the data are growing rapidly, and the computation relies heavily on this massive data. Moreover, the fast speed at which the data are generated make it become infeasible to stockpile the data into any explicit server. It is a challenging task to hold a gigantic amount of big data, which continue to raise exponentially at an extraordinary speed [9], [10]. In IoT-enabled edge computing environments, major sources of Big Data are the embedded sensors [11], [12].

With the rise of Big Data in IoT-enabled edge environments, the notion of machine learning (ML) model development is gaining prominence [13]–[15]. Moreover, the federated learning concept is also introduced at the edges of the network [16]. ML-based models deliver enriched performance in the market by utilizing the huge data [13], [17]. In this context, low-cost computing devices, sensors, and storing mechanisms integrated with convincingly speedy wireless equipment provide the foundation for managing the big data [18], [19]. However, an accessible substructure is still required to deal with the massive set of devices. Big Data analytics has the ability of providing massive opportunities to resolve problems related to various applications dealing with plethora of data [7], [20]. The future of this universe lies with ML-enabled IoT paradigms that aim to bring intelligence to the real-world physical objects [21]. Various IoT applications seem to be progressively integrated with ML techniques.

For edge computing, it becomes challenging to manage the diverse IoT applications as they produce massive amount of big data. Edge computing is the future of big data analytics.

[1]Big Data and big data are used interchangeably.

The domain of big data has undertaken a huge revolution over the decade. This revolution is creating a shift in computing infrastructure; where the computing is moving out of the data center to the edge. The architectures of IoT-enabled Big Data analytics face numerous challenges, such as the storage and processing of high volume of data, latency, data heterogeneity, inconsistent data patterns, and so forth, in their existing structural design. Moreover, the existing proposals lack an efficient parallel data ingestion and incur a high communication overhead [22]. The major challenges and problems that need to be undertaken are inadequate data ingestion into the traditional cluster supervision frameworks, e.g., Apache Hadoop/Spark. The customary data ingestion is time wasting, requires more storage, contains commands that are difficult to understand, has no appendage, and has no partial ingestion [22]. Similarly, the processing of traditional big data analytics platforms based on conventional cluster management faces challenges such as difficult scheduling, inefficient load balancing, nonscalability, and responsibility unification.

This research aims to design a specific framework using edge computing to evaluate the huge data proficiently and overcome the data ingestion and computation issues. We adopt the optimization of yet another resource negotiator (YARN) framework to customize for edge computation. The major contributions of this article include the following.

1) A framework for data analytics at the edge including parameter selection for modules affecting the distributed files for edge data availability, efficient preprocessing to prepare the data for quick processing and accurate prediction, and efficient technique to concurrently store the data at the edges.
2) An algorithm for parallel data ingestion, an edge–map-reduce (MR) algorithm for parallel processing at every edge, an edge–yarn algorithm for efficient cluster resource management by optimizing the YARN, and an enhanced weighted backpropagation (BP) algorithm for model parallelism.
3) Simulation of the proposed scheme and its algorithms performed using the opensource Apache Spark 3.0 framework. We preferred the opensource platform to achieve the cross-platform applicability.

The remainder of this article is organized as follows. In Section II, we provide a comprehensive literature review. In Section III, we present our proposed IoT-enabled big data analytics framework. In Section IV, the experimental results and analysis are provided to validate our framework. Finally, we conclude this article in Section V.

II. Literature Review

Superfast network connectivity and novel frameworks of IoT are just some of the things transforming applications around the globe. This sort of transformation does not happen in isolation and overnight. It is easy to lose the benefits of IoT with the presence of Big Data, but connectivity at the network edges offer alternative transmission options to the end users. Nevertheless, there are challenges faced by Big Data and IoT in the edge computing paradigm related to interoperability issues, heterogeneity problems, data preparation and cleaning, data format challenges, normalization of data, data filtration, and loading data into processing frameworks. Researchers have been working to tackle the challenges of big data in complex environments for over a decade now. A summary of existing proposals for big data processing of IoT is provided in Table I. The related works are tabulated along with the key challenges. A specific solution is suggested to process the Big Data in a healthcare domain that contains the separation of dynamic data into subgroups, which are based on the hypothetical data fusion working in Hadoop to improve computational effectiveness [23]. The key challenges identified for the proposed scheme are the use of customary MR mechanism, insufficient data loading, and impalpable structure. The data loading efficiency is ignored in the architecture and only the health data set is evaluated. A scheme is proposed for data management, which is composed of different tiers [24]. Tiers are liable for different steps of data processing. Nevertheless, it is a comprehensive design that consists of four tiers from big data gathering to analysis of big data, it also causes computation delays, and the use of a legacy MR framework slows down the performance. Moreover, the focus is mainly on data aggregation before data ingestion, while data loading competence is overlooked [25].

Hussain *et al.* [26] proposed an IoT framework using Hadoop-based Big Data analytics with different layers from data acquisition to application. The major issue in the design of this architecture is that data loading effectiveness was overlooked. Some researchers proposed a model based on data

TABLE I
State-of-the-Art Models for IoT Data Processing

Proposals	Challenges
[23]	• Used Traditional Cluster management • Data loading efficiency is ignored in the architecture • Only health dataset is evaluated
[24]	• Causes delay in processing • Data accumulation prior to data ingestion is highlighted while data ingestion proficiency is overlook • Classical map-reduce framework used
[25]	• Computation at edge is overlooked during processing
[26]	• The data ingestion proficiency is overlooked and the utilization of MRV1 framework • Limited architecture
[28]	• Conceptual and logical framework • Implementation is not performed
[29]	• It causes additional delay This scheme is also insufficient to load data efficiently to the Hadoop • Conventional cluster resource handling • Only transportation dataset are evaluated
[30]	• Customary and conventional cluster resource handling • It fails to load data efficiently
[31]	• Data loading mechanism exist but delay in loading • Another issue is the utilization of classical map-reduce framework
[32]	• This scheme is also insufficient to load data to the Hadoop • Traditional cluster resource handling is utilized in this scheme
[28]	• Data collection is preferred and data ingestion prior to analysis is overlooked • Only health dataset is evaluated to test the proposed system

analytics that encourages the concept of smart and connected societies (SCCs), which is developing from the notion of smart cities [27]. The SCC model is a theoretical structure that is not yet implemented for the physical world. Likewise, Tönjes *et al.* [28] presented a framework, however, this is also a theoretical prototype. These researchers also overlooked the data loading process in the context of a parallel and distributed domain. There are few designs recommended to tackle the challenges of Big Data analytics within a smart infrastructure [29]–[31], but these solutions utilized the customary cluster management mechanisms and unsatisfactory data ingestion to Apache Hadoop. Furthermore, a design was proposed to explore the data in an accessible and efficient transport environment [32]. However, the proposed work causes additional processing delay. Moreover, the proposed work was only verified using transport data, and data loading competence was ignored during Big Data loading to Hadoop. A proposal was presented that promotes the data aggregation of results but the data loading before analysis was overlooked [25]. These studies conclude with the following challenges that need careful consideration.

1) Effective data loading into the Hadoop distributed file system (HDFS) for edge data availability is missing.
2) Lack of efficient and improved method to load and store the data sets at the edge.
3) Lack of efficient cluster resource management.
4) Lack of efficient processing for edge devices while taking into consideration the parallel and distributed paradigm.

Architectures of Big Data analytics for complex environments e.g., edge–cloud, is a novel initiative to switch the traditional form of services and facilities to a well-designed mode [33]–[37]. This changeover seems to be challenging as there are no predefined instructions available to construct a robust architecture for Big Data computation. IoT offers services using sensors and other objects that supply a huge quantity of data for analysis. Big Data analytics using Hadoop involves several varied phases including data loading and data processing that bring issues and challenges. Although, numerous research works exist to analyze the Big Data efficiently and measure its performance, studies dealing with data loading efficiency are exceedingly rare. The literature suggests that several proposals have been presented to analyze the Big Data in smart city environment using the Hadoop framework, but significant challenges exist that need to be dealt with, e.g., efficient data loading and processing using an effective cluster management technique. Therefore, we discover the need for a resourceful model of Big Data analytics and proposes an improved architecture for this purpose.

III. IoT-Enabled Big Data Analytics for Edge–Cloud

In this section, we design a big data processing scheme using ML models. The ML models are trained using the huge big data produced within the IoT–Edge environment.

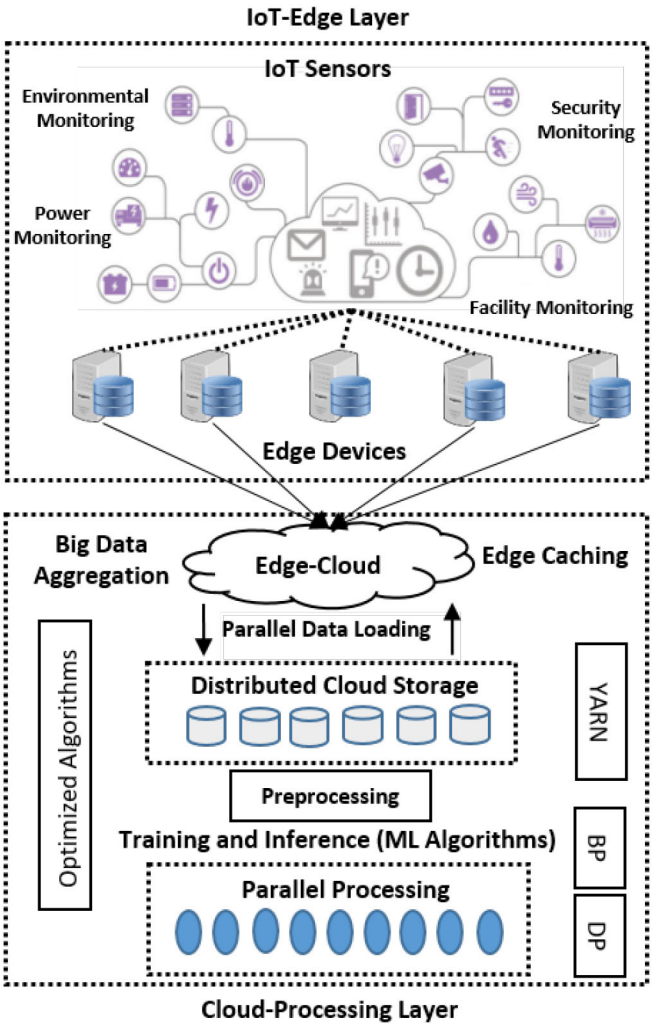

Fig. 1. System model of our IoT-enabled data analytics framework.

A. System Model

The system model of the proposed IoT-enabled Big Data analytics framework is discussed in this section. The model is composed of two different layers: 1) IoT–edge layer and 2) cloud-processing layer as depicted in Fig. 1.

The first layer is comprised of IoT sensors and embedded devices integrated with edge servers. The second layer is composed of cloud and processing units that perform various activities, e.g., big data aggregation, distributed storage of big data, and big data processing with model training. The cloud storage is responsible for holding various sources of data. The data are produced by various IoT sensors such as environmental observing, security monitoring [38], facility monitoring, traffic, power observing, and transportation observing sensors. The generated data are received by edge devices and servers. The edge data are properly gathered by various sophisticated processes known as edge caching. Finally, the edge data are collected by cloud that performs distributed storage. The gathered data are feed into a distributed architecture for parallel processing. It involves the overall data management that includes collection, aggregation, and storage. The data aggregation offers the grouped data with a precise shaping for

Algorithm 1 Data Normalization

BEGIN
Input: = Each value of data block
Max value of corresponding data block
Min value of corresponding data block
Output: = scaled value
Set upper and lower limits (m, n)
// specific range
Identify Max and Min values ($Xmax$, $Xmin$)
For each (data item) do
$Y = \frac{(X-Xmin)}{(Xmax-Xmin)}(*(n-m)+m$
END

Algorithm 2 Filtration Using KF

BEGIN
Instantiation
instantiate transition model (T)
instantiate observation model (O)
instantiate noise co-variance (CN)
instantiate OBS co-variance (CO)
instantiate control input (CI)
Calculating first-hand DataSplit (D)
Search previous DataSplit ($D1$)
Search new DataSplit
Approximating standing DataSplit
DataSplit prediction (DP)
co-variance prediction (CP)
Merging present observation with prediction
Discover present observation (X)
Discover co-variance of observation (S)
Discover optimal gain (K)
Update DataSplit (PD)
Update co-variance
Prediction
Observation
END

further analysis, which is useful for enormous data that lacks valued information. The preferred technique for aggregation is divide-and-conquer that divides the huge data set into smaller chunks and groups the similar data in the form of blocks.

The preprocessing mechanisms, e.g., normalization, filtration, and queuing, are utilized to prepare the data for efficient processing and training. The normalization of data is performed using a min–max normalization method as depicted in Algorithm 1. The filtration is performed using an optimized Kalman filter (KF) and the queuing is carried out using the hybrid M/M/1 queuing model. Algorithm 2 depicts the working of filtration process to speed the actual processing. KF provides only the quality information to be processed. It filters the inferiority information that affects the quality. Moreover, KF is ideal for real-time processing that can easily be integrated with the Apache Spark platform and avoids heterogeneity. The message queue is also utilized to speed up

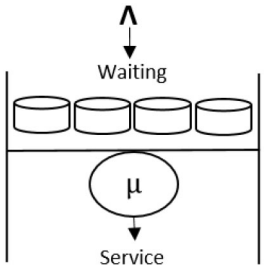

Fig. 2. M/M/1 queuing process.

the data processing. The message queue functions in a particular operational mode to obtain message M at time t and then forwards it to the resultant component. It is controlled by H (a specific handler). This enhances the efficient exploitation of big data for better results. The data are loaded to the processing server with parallel loading using a map-only parallel algorithm. It speeds up the overall computation time. The number of parallel channels is kept balanced by optimizing the block size of processing unit. The big data processing is performed by dividing the big data sets into small chunks of fixed block sizes. The blocks are processed by each node in parallel. The default block size of Hadoop (the rationale of Apache Spark) takes more time to be processed and provides less number of parallelism. The default size is optimized and modified to enhance the data ingestion efficacy. The default block size in the Hadoop latest version is 128 MB. The default size creates too many replicas and communication overhead. Therefore, the default block size is tuned using the block-size parameter.

The processing and computation of big data are performed using the parallel and distributed framework of big data analytics. The proposed architecture is realized using the Apache Spark 3.0 framework. The traditional Spark is customized using an optimized YARN cluster manager. The execution time of the overall big data computation using the proposed design is compared with the state-of-the-art existing related works. The parallel processing is equipped with a machine-learning algorithm of BP neural network. This module includes the training and inference of BP neural network's ML model. The training of the model is performed with a 70:30 ratio of the data set. This module provides the customization of BP algorithm integrated with the proposed architecture. The novelty of the BP network is the model parallelism. The proposed BP network is trained and validated in parallel with several processing nodes. The partial results produced by the nodes are merged using the ensemble learning for better results and improvement.

B. Hybrid M/M/1 Queue

To speed up the data processing for efficient exploitation of big data, we optimized the M/M/1 queuing model. This model performs various operations when it receives the data split D at time t. At this time, a system is thought to be in the steady state. Figs. 2 and 3 demonstrate the data processing and transmission of an M/M/I queue.

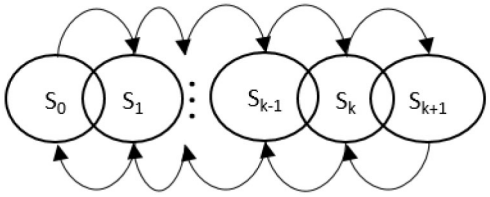

Fig. 3. M/M/1 queue computation.

The two splits, e.g., S_1 and S_2, are balanced during the steady state. Here, $S_2, S_3, \ldots, S_{k-1}, S_k, S_{k+1}$, can be calculated as

$$\Lambda S_0 = \mu S_1 \tag{1}$$

$$\Lambda S_1 = \mu S_2 \tag{2}$$

$$\Lambda S_{k-1} = \mu S_k \tag{3}$$

hence

$$S_n = \left(\Lambda/\mu\right)^k S_k - 1 = \left(\Lambda/\mu\right)^k S_0. \tag{4}$$

As the probability is equal to 1

$$\sum_{j=0}^{\infty} S_j = 1 \tag{5}$$

$$S_0\left(1 + \left(\Lambda/\mu\right)^1 + \left(\Lambda/\mu\right)^2 + \cdots\right) = 1. \tag{6}$$

Summing the series

$$S = 1 - \Lambda/\mu. \tag{7}$$

As $S = 1 - S_0$, thus

$$S = 1 - \left(1 - \Lambda/\mu\right) \tag{8}$$

$$S = \Lambda/\mu \tag{9}$$

$$S_k = \left(\Lambda/\mu\right)^k S_0 \tag{10}$$

$$\left(S\right)_n = \left(1 - S\right) \tag{11}$$

$$S_n = SR^k\left(1 - S\right) \tag{12}$$

$$\text{MS} = \sum_{m=0}^{\infty} m.S_j \tag{13}$$

$$\text{MS} = \sum_{m=0}^{\infty} m.S^n\left(1 - S\right) \tag{14}$$

$$\text{MS} = \left(1 - S\right)S\frac{d}{ds}\sum_{m=0}^{\infty} S^m \tag{15}$$

$$\text{MS} = \left(1 - S\right)S\frac{d}{ds}\left(\tfrac{1}{1-S}\right) = \frac{1}{1-S}. \tag{16}$$

The average is

$$T = \frac{N}{\lambda} = \frac{S}{1-S}\frac{1}{\Lambda} = \frac{1}{\mu - \Lambda}. \tag{17}$$

As we know MQ = M-Tasks

$$\text{MQ} = \left(\frac{S}{1-S}\right) \times S = \frac{S^2}{1-S} = \frac{\Lambda^2}{\left(\mu - \Lambda\right)^{\mu}}. \tag{18}$$

Hence, the waiting time is

$$W_Q = T - \frac{1}{\mu} = \frac{1}{\mu - \Lambda} - \frac{1}{\mu} = \frac{\Lambda}{\left(\mu - \Lambda\right)^{\mu}}. \tag{19}$$

Finally, the average number of tasks in a system becomes

$$M = \frac{S}{1-S} = S + \frac{S^2}{1-S}. \tag{20}$$

C. Map-Only Algorithm for Parallel Data Ingestion

The Apache Hadoop and Apache Spark process the data available inside the Hadoop in an HDFS format. The data ingestion is the key stage for both batch and stream processing. The data can either be loaded in batch or stream form. The data loading is dependent on the kind of processing available inside the parallel and distributed platform. The data must be loaded to the parallel processing platform before processing. The traditional mechanism of data loading is sequential and time consuming. Therefore, the MR programming paradigm is optimized for parallel data loading. The data loading also improves the overall data processing. The data analysis within the big data platforms, e.g., Apache Hadoop and Apache Spark, include the loading of data. The big data analytics platforms carry the processing of data by loading it in the required format. As a result, the data loading efficiency improves the overall data processing. We integrate the Sqoop utility with the map job of the MR paradigm and introduced a map-only algorithm. In addition, the split size and replication factor of the traditional approach are also tuned with the optimized map-only algorithm for parallel data ingestion. The split size is defined using the unit byte. The replication based on the file can also be modified accordingly. First, we need to create a new directory at the root. Next, we need to verify and add a file to the directory. Afterward, a command is executed to change the replication. Similarly, the replication of all files can be changed within a directory. For this purpose, specific and generic arguments can be used to control the operations of the Sqoop tool.

The general command-line parameters of Hadoop must precede any tool-specific parameters. This specific tool of Sqoop utility is utilized in the proposed architecture to import a particular table (source) to HDFS from an RDBMS. The HDFS format for processing is mandatory as the processing is carried out in the HDFS format by Hadoop and Apache Spark in the proposed framework. Every table line is symbolized as a separate (detach) record that can be saved as text files in HDFS or it can also be saved in binary form as Avro or Sequence Files. The –table argument is used to choose the table that is about to be loaded. A specific subset of the columns can also be selected and can be controlled using –columns argument if required. Similarly, specific rows can also be imported using the SQL WHERE clause. SQL WHERE clause is a very useful tool that can enhance the utilization of Sqoop commands. One of the most beneficial qualities of the Sqoop tool is the incremental import. Sqoop offers –append argument that is used for loading in an incremental import manner, which can be utilized to get only newer records or data.

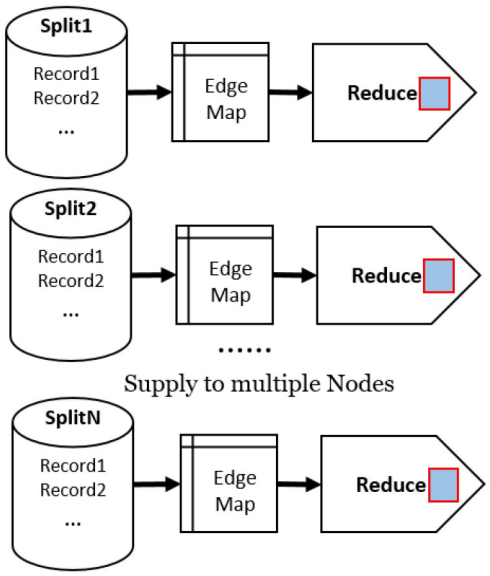

Fig. 4. MapReduce operation for parallel processing.

D. Map-Reduce for Parallel Data Processing

The parallel processing in the edge environment is achieved by the MR mechanism. It provides a parallel mechanism for data processing. Big data are huge and need to be divided into blocks or splits for distributed storage and parallel processing. Therefore, the MR paradigm is preferred. The MR programming paradigm is preferred as it is more scalable, flexible, cost effective, fast, simple, and resilient. The process of mapping and reducing is depicted in Fig. 4. The traditional MapReduce paradigm is optimized for edge data processing by introducing the edge-map and edge-reduce phases. This edge-based MapReduce is introduced to realize the edge-based big data processing. The traditional cluster management is also replaced with optimized edge–YARN. The edge–YARN performs only the resource management inside the cluster along with other operations in the cluster. It is responsible for the overall cluster management. YARN is optimized for the edges of the system. The previous forms of distributed podiums, e.g., customary Hadoop by Apache, utilized the MR for computation and the management of clusters. Job dissemination in the YARN-distributed cluster management framework is performed with optimized and tuned parameters. Earlier platforms, e.g., the previous versions of traditional Hadoop and MR were used for cluster management along with parallel processing. However, they generated higher communication overhead and reduces the overall performance. To overcome these challenges, YARN is favored and optimized as it achieves the separate functionalities for cluster management. We considered three major parameters: 1) the number of replicas; 2) block size; and 3) the level of parallelism. The big data are processed using the parallel and distributed platform. Here, the data are distributed into several blocks (distributions) and each block is inserted in a parallel fashion. It is mandatory to choose the amount of parallelism, i.e., how many nodes (blocks) are required to be inserted for processing and computation. This concept is known as level of parallelism.

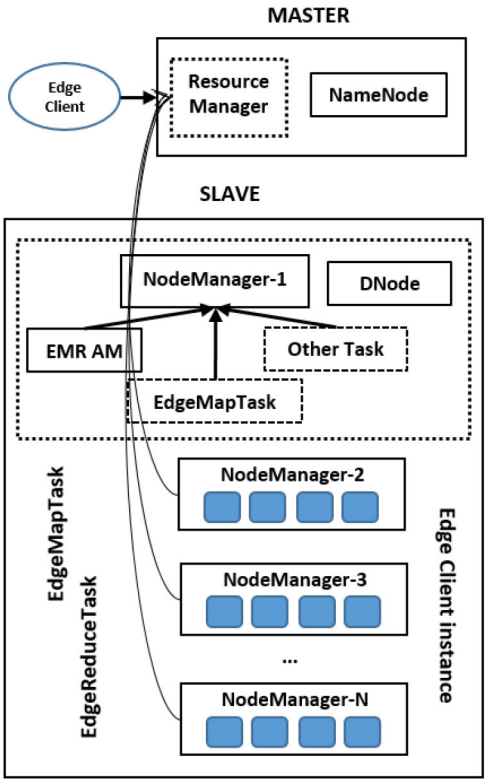

Fig. 5. Tailored YARN for edge.

The YARN framework is furnished with dynamic programming (DP) for job distribution and managing the cluster. DP is used for the recursion function. The enormous data set is repeatedly distributed into various blocks. In this context, DP is preferred as it is an ideal algorithm for recursive problems by breaking the problem into simpler subproblems to produce an optimal solution. YARN operation in an edge computing paradigm is reflected in Fig. 5. The MRAppMaster delivers the implementation of application master (AM) in the YARN-enabled framework. Besides, interweaving is probable among map and reduce stages; hence, the reduce stage may start prior to the mapping stage ends.

The resource manager (RM) allocates a container (CON) when required by the AM. Any specific request is entertained by the CON to exploit the required resources on a particular host. AM requests the node manager (NM) accountable for controlling the host (holding the CON) to take off the dedicated task of an application. This task could be any MR task. NM only assesses the resource usage and destroys any resource that utilizes extra memory than originally assigned. AM pays out its entire existence by negotiating various CONs to initiate request(s) to complete its application. AM is also accountable for restarting the job in new CONs if the previous one fails. It provides the progress back to the user. AM ends the job and relieves its CON on completion of the job. However, RM does not carry out checking of the tasks in an application; it endorses the strength of AM. A demand for a specific resource is simply a request for few CONs to guarantee several resource provisioning.

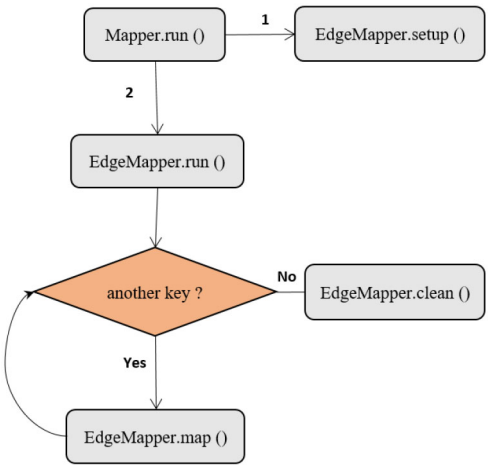

Fig. 6. EdgeMapTask executions.

The execution of EdgeMapTask includes the INIT, EXECUTION, SPILLING, and SHUFFLE phases. The INIT phase of EdgeMapTask is responsible for the overall setup of EdgeMapTask. The EXECUTION step of EdgeMapTask is carried out by a specific method of Mapper class, known as the run method, which is customized according to the required edge information. After the setup of the system, the edge-map () method is called for every $<k, v>$ tuple enclosed in the split. Therefore, map () accepts a specific key, a specific value, and a specific mapper context. The outcome of the edge-map process is stocked in a specific buffer with the context function. When the complete edge-mapping fragment has been processed, the clean function is executed by a run command. Therefore, no action is carried out by default but the manager may choose to override this method. This scenario is shown in Fig. 6.

E. Tailored BP Machine Learning Algorithm

The BP network can learn and store a huge number of inputs and mapping relations of the expected outcome. The customization of BP algorithm includes the customized training and inference of BP neural network. It can automatically fine-tune the network weights and threshold limit values (TLVs). The TLV adjustment is performed via error approximation and error BP. Therefore, it is preferred to train the received big data. The innovation in the traditional BP model is the model-building parallelism. The proposed optimized BP model is trained and validated in parallel using various processing nodes. The fractional results produced by each processing node are merged using ensemble learning for improved results. The BP model is composed of the input layer, the hidden layer, and the output layer as depicted in Fig. 7.

The preferred variant of the BP network is the additional momentum technique, which is designed for building a parallel model by introducing the coefficient of momentum μ, using the algorithm of gradient descent. The coefficient along with weights can be shown as follows:

$$\Delta\omega(n+1) = \mu\Delta\omega(n) + \Lambda(1-\mu)\frac{gx}{gw} \tag{21}$$

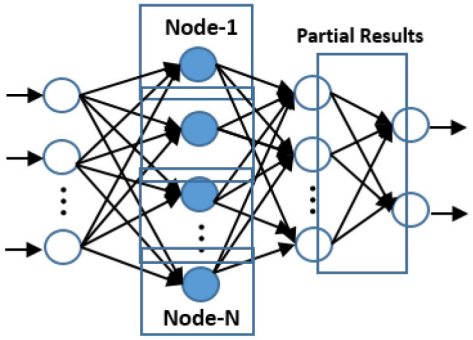

Fig. 7. Hybrid BP neural model structure.

where $\Delta\omega(n+1)$ and $\Delta\omega(n)$ symbolize the weights after the $(n+1)$th and the nth iterations. The value of μ must be between 0 and 1. The gx/gw denotes the negative value of gradient, e.g., error sum of squares to ω in the BP model. We utilized the variable learning rate approach for self-adaptive adjustment according to the variation in error. The adjustment of the learning rate in the self-adaptive adjustment can be calculated as

$$\Lambda(n+1) = \begin{cases} m_{++} & \Lambda(n), \quad X(n+1) < X(n) \\ m_{--} & \Lambda(n), \quad X(n+1) > X(n) \\ \Lambda(n) \end{cases} \tag{22}$$

where the incremental factor m_{++} is greater than 1 and the decremental factor m_{--} is between 0 and 1. Here, $X(n+1)$ and $X(n)$ denote the total error sum of squares after the $(n+1)$th and nth iterations. Finally, the Δ symbolizes the learning rate. The philosophy managing the learning directions of BP is the adjustment to the weight and threshold of network should be performed with the direction of the gradient

$$S_{i+1} = S_i - \Lambda_i g_i \tag{23}$$

where S_i denotes the matrix of existing weight TLV, g_i denotes the gradient of current operation, and Δ_i denotes the learning rate. Suppose the three-layer BP model with the input node is y_j, the node of hidden layer is x_j, and the node of output layer is z_i, then we can have

$$x_j = f\sum_{m=0}^{\infty} w_{jm}y_j - \theta_j = f(\mathrm{NET}_j) \tag{24}$$

where

$$\mathrm{NET}_j = \sum_{m=0}^{\infty} w_{jm}y_j - \theta_j. \tag{25}$$

The computational output of the output node is

$$z_j = f\sum_{m=0}^{\infty} v_{jm}y_j - \theta_j = f(\mathrm{NET}_j). \tag{26}$$

The output node's error is

$$\mathrm{ERROR} = \frac{1}{2}\sum_{m=0}^{\infty}(t_m - z_m)^2 \tag{27}$$

$$\mathrm{ERROR} = \frac{1}{2}\sum_{m=0}^{\infty}(t_m - F)^2 \tag{28}$$

where

$$F = \left(\sum_{m=0}^{\infty} v_{jm} y_j - \theta_j \right)^2. \tag{29}$$

IV. Experimental Results and Discussion

In this section, the experimental results are discussed in detail. The proposed scheme is implemented using the Apache Spark 3.0 parallel and distributed framework. Spark is customized using an optimized YARN cluster management approach. The Spark MLlib library is utilized for the execution of ML neural network model. The MLLib library is provided by Spark. We used the MLLib for the ML model execution with a 70–30 ratio of training and testing. We can also split the data set for training and testing using 80–20 ratio, but most of the research studies follow 70–30 ratio of split. Training a model with 80% of the data set can affect the model performance. The efficiency of the ML model is assessed using the key parameters including accuracy, specificity, precision, recall, F-score, specificity, and sensitivity, respectively. The values of the parameters are calculated based on the confusion matrix. The data ingestion results are carried out using the traditional as well as the proposed approaches. The data loading results are compared with the traditional approach of data ingestion. The execution time and throughput results are compared with state-of-the-art existing proposals. The BP model evaluation is performed using the confusion matrix values to find the accuracy of the customized model. Authentic IoT-enabled data sets with big data characteristics are utilized to evaluate the applicability and performance of the system. The data sets include the IoT-enabled health, traffic, pollution, and water data sets [39]–[42]. The selected parameters, e.g., selection criteria for the results and validation section, is the selection criteria utilized by the base paper. We used similar setup of the existing studies. Moreover, we used identical data sets for evaluation and comparative analysis. Furthermore, the identical configuration (including CPU, storage, nodes, and cluster) is favored for evaluation. The proposed framework is interoperable and supports the cross-platform applicability as it is implemented in the Apache Spark, which is an open source framework.

A. Data Ingestion

The data ingestion is performed using an optimized mapping algorithm of the MapReduce paradigm to load the data in parallel and speed up the overall execution time of big data processing and training module. We optimized the MR programming paradigm and proposed an optimized map-only algorithm for data ingestion. The map-only algorithm is integrated with proposed architecture for data loading. In map-only algorithm, all the tasks in the system are performed in parallel that improves the data ingestion competence of our proposed system. It is observed that it gets approximately no time to load the data split into the distributed mechanism of Spark when the size of the data is small, e.g., up to 2 GB. The data loading efficiency is highlighted in Fig. 8 that reveals drastic changes when the data size increases. It is evident from the figure that the proposed data loading is better than the traditional, i.e., classical approach. The proposed method loads the data with less time at each processing node of the cluster. The overall efficiency and improvement of the proposed big data ingestion module with a comparison to the traditional approach is also depicted in Fig. 9. In addition, the threshold value is highlighted in Fig. 10. The TLV is a specific range where the effect of the proposed approach is noted. The TLV of the proposed approach is approximately 2 GB. The data loading efficiency is revealed when the data exceeds the 2-GB size.

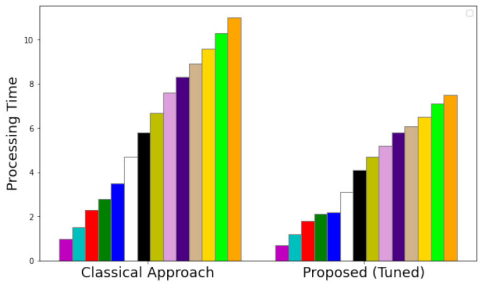

Fig. 8. Execution time of the proposed data loading approach.

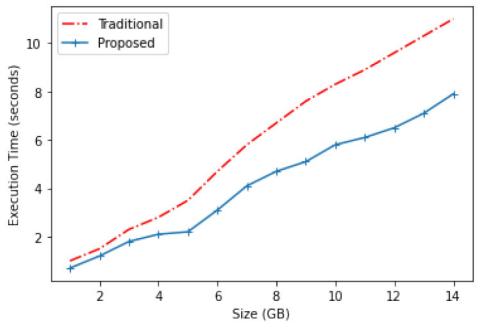

Fig. 9. Execution time comparison.

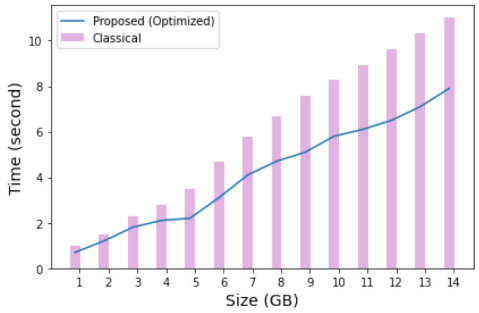

Fig. 10. Execution Time TLV.

B. ML Model Performance Evaluation

The ML model is implemented using the Apache Spark MLlib library. The evaluation of the ML model is performed using the confusion matrix. The parameters include accuracy, precision, recall, and F-measure. In addition, sensitivity, and specificity, are also considered. Accuracy is a ratio of the correctly classified records to the total number of records. Precision is a ratio between the correctly predicted positive

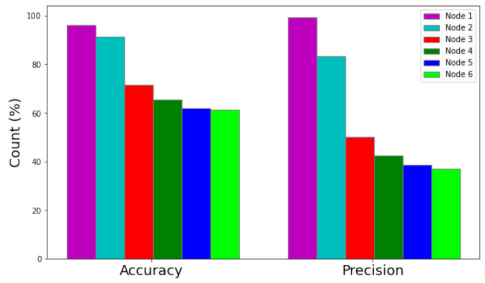

Fig. 11. BP model accuracy and precision.

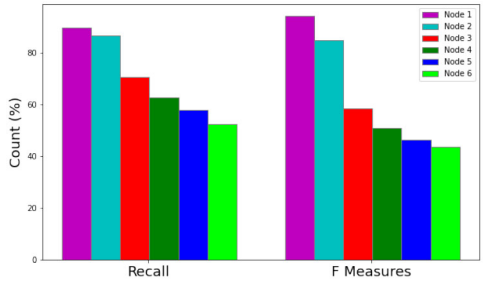

Fig. 12. BP model recall and F-measure.

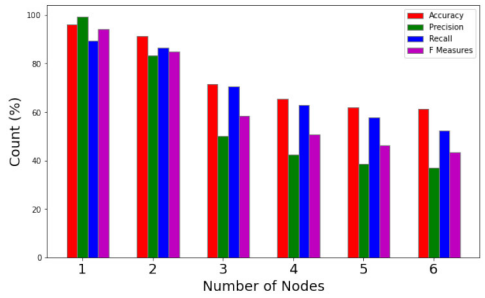

Fig. 13. KPIs comparison.

records over the total positive predicted records. Recall is a ratio between correctly predicted positive records over the total related records. A combination of weighted average precision and recall indicators tell about the overall measured accuracy. $F1$ score of 1 is considered the perfect and the model is considered failed if the score is equal to 0.

The accuracy and precision of the proposed system are presented in Fig. 11. The figure highlights the comparative analysis of accuracy and precision between different processing nodes. The recall and F-measure of the system for multiple nodes are also depicted in Fig. 12. Similarly, the comparison of all the KPIs is provided in Fig. 13. Moreover, we provided the sensitivity and specificity of the proposed system model in Fig. 14. The proposed optimized learning model is also realized without Spark. The model accuracy can be further enhanced by expending dissimilar batch size and progressively increasing the epochs number. It is revealed that we achieve 81.07% accuracy on batch size 200 and the number of epochs 170 as depicted in Fig. 15. The accuracy refers to how often the model's predicted results are correct. Accuracy is the ratio between correctly classified images to the total number of images.

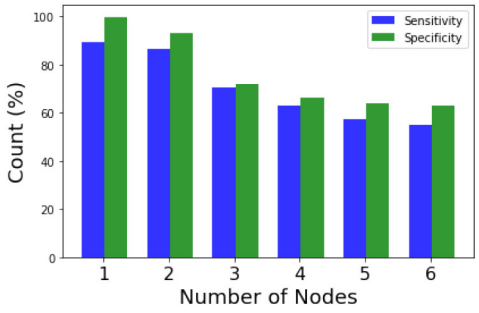

Fig. 14. Sensitivity and specificity of the weighted model.

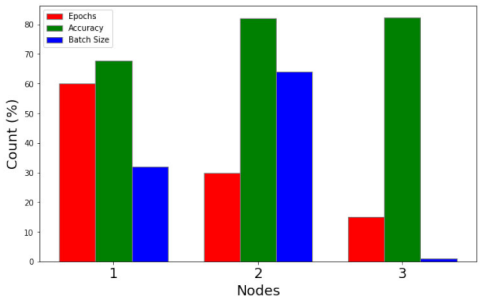

Fig. 15. Epoch-batch size-accuracy comparison.

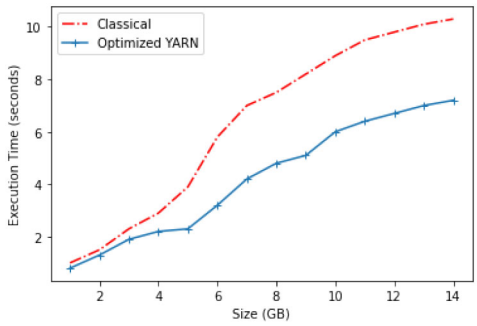

Fig. 16. Tailored YARN execution time.

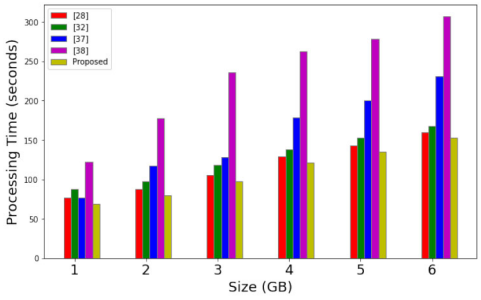

Fig. 17. Processing time comparison.

C. Optimized YARN

The experimental results of our proposed architecture for execution time and throughput are presented in Figs. 16 and 17, respectively. The customization in the YARN cluster management improves the processing time as shown in Fig. 16. The processing time in the context of parallel nodes is also elaborated in Fig. 17. The throughput increases with a reduction in the computation time. The computation time reduces as

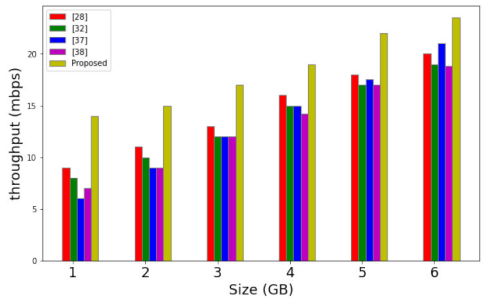

Fig. 18. Throughput comparison.

the parallel and distributed framework is customized with optimized replica number, block sizes, and the usage of a aligned utility, as shown in Fig. 18.

V. Conclusion

The massive data generated by IoT devices grow at an exponential rate at the network edge. Edge-enabled solutions offer efficient computing and control nearer to the network edge to resolve the scalability and latency challenges for IoT devices. The existing solutions face numerous challenges in their structural design, for instance, the high volume of data storage and processing, data heterogeneity, and processing time among others. Moreover, parallel data ingestion and robust mechanism for handling communication overhead is also missing in the existing approaches. To address these challenges, this article proposes an IoT-enabled big data analytics framework for edge-centric computing. In the proposed scheme, an edge intelligence module is introduced and equipped with an optimized weighted-tuned ML model. To process and store the big data efficiently at the network edges, the optimized processing module is introduced with the integration of the cloud platform. The proposed scheme is composed of two layers, i.e., IoT–edge and cloud processing. The data injection and storage are carried out with an optimized MapReduce parallel algorithm. Optimized YARN was used for efficient management of the cluster. The proposed data design was simulated based on a real-world data set using the Apache Spark and has much better results as compared to the traditional approaches.

References

[1] Y. Liu *et al.*, "Zero-bias deep learning for accurate identification of Internet-of-Things (IoT) devices," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2627–2634, Feb. 2021.

[2] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 103–109, Sep. 2018.

[3] W. Yu *et al.*, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017.

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

[5] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni, "Deploying edge computing nodes for large-scale IoT: A diversity aware approach," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3606–3614, Oct. 2018.

[6] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 110–115, Nov. 2018.

[7] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100318.

[8] X. Zheng, L. Tian, B. Hui, and X. Liu, "Distributed and privacy preserving graph data collection in Internet-of-Thing systems," *IEEE Internet Things J.*, early access, Sep. 13, 2021, doi: 10.1109/JIOT.2021.3112186.

[9] J. Ranjan and C. Foropon, "Big data analytics in building the competitive intelligence of organizations," *Int. J. Inf. Manage.*, vol. 56, Feb. 2021, Art. no. 102231.

[10] S. K. Singh and A.-N. El-Kassar, "Role of big data analytics in developing sustainable capabilities," *J. Clean. Prod.*, vol. 213, pp. 1264–1273, Mar. 2019.

[11] D. Blazquez and J. Domenech, "Big Data sources and methods for social and economic analyses," *Technol. Forecast. Social Change*, vol. 130, pp. 99–113, May 2018.

[12] Z. Lv, R. Lou, J. Li, A. K. Singh, and H. Song, "Big data analytics for 6G-enabled massive Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7 pp. 5350–5359, Apr. 2021.

[13] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, May 2017.

[14] W. Li *et al.*, "A comprehensive survey on machine learning-based big data analytics for IoT-enabled smart healthcare system," *Mobile Netw. Appl.*, vol. 26, pp. 234–252, Jan. 2021.

[15] R. Marculescu, D. Marculescu, and U. Ogras, "Edge AI: Systems design and ML for IoT data analytics," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2020, pp. 3565–3566.

[16] B. Nour, S. Cherkaoui, and Z. Mlika, "Federated learning and proactive computation reuse at the edge of smart homes," *IEEE Trans. Netw. Sci. Eng.*, early access, Nov. 30, 2021, doi: 10.1109/TNSE.2021.3131246.

[17] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data—AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021.

[18] Y. N. Malek *et al.*, "On the use of IoT and big data technologies for real-time monitoring and data processing," *Procedia Comput. Sci.*, vol. 113 pp. 429–434, Dec. 2017.

[19] R. Iqbal, F. Doctor, B. More, S. Mahmud, and U. Yousuf, "Big data analytics: Computational intelligence techniques and application areas," *Technol. Forecast. Social Change*, vol. 153, Apr. 2020, Art. no. 119253.

[20] A. Taherkordi, F. Eliassen, and G. Horn, "From IoT big data to IoT big services," in *Proc. Symp. Appl. Comput.*, 2017, pp. 485–491.

[21] A. Z. Chaudhry, R. Mumtaz, S. M. H. Zaidi, M. A. Tahir, and S. H. M. School, "Internet of Things (IoT) and machine learning (ML) enabled livestock monitoring," in *Proc. IEEE 17th Int. Conf. Smart Commun. Improving Qual. Life Using ICT IoT AI (HONET)*, 2020, pp. 151–155.

[22] V. Porkodi, D. Yuvaraj, J. Khan, S. A. Karuppusamy, P. M. Goel, and M. Sivaram, "A survey on various machine learning models in IOT applications," in *Proc. Int. Conf. Comput. Inf. Technol.*, 2020, pp. 1–4.

[23] S. Din, H. Ghayvat, A. Paul, A. Ahmad, M. M. Rathore, and I. Shafi, "An architecture to analyze big data in the Internet of Things," in *Proc. ICST*, Dec. 2015, pp. 677–682.

[24] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the Internet of Things using big data analytics," *Comput. Netw.*, vol. 101, pp. 63–80, Jun. 2016.

[25] M. M. Rathore, A. Paul, A. Ahmad, M. Anisetti, and G. Jeon, "Hadoop-based intelligent care system (HICS): Analytical approach for big data in IoT," *ACM Trans. Internet Technol.*, vol. 18, no. 1, p. 8, 2018.

[26] W. Hussain, J. M. Merigó, M. R. Raza, and H. Gao, "A new QoS prediction model using hybrid IOWA-ANFIS with fuzzy C-means, subtractive clustering and grid partitioning," *Inf. Sci.*, vol. 584, pp. 280–300, Jan. 2022.

[27] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of Things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, 2016, doi: 10.1109/ACCESS.2016.2529723.

[28] R. Tönjes *et al.*, "Real time IoT stream processing and large-scale data analytics for smart city applications," in *Proc. Eur. Conf. Netw. Commun.*, Bologna, Italy, 2014, pp. 1–5.

[29] M. M. Rathore, A. Paul, A. Ahmad, and G. Jeon, "IoT-based big data: From smart city towards next generation super city planning," *Int. J. Semantic Web Inf. Syst.*, vol. 13, no. 1, pp. 28–47, 2017.

[30] B. N. Silva, M. Khan, and K. Han, "Big data analytics embedded smart city architecture for performance enhancement through real-time data processing and decision-making," *Wireless Commun. Mobile Comput.*, vol. 2017, Jan. 2017, Art. no. 9429676.

[31] B. Cheng, S. Longo, F. Cirillo, M. Bauer, and E. Kovacs, "Building a big data platform for smart cities: Experience and lessons from santander," in *Proc. 4th IEEE Int. Congr. Big Data (BigData Congress)*, 2015, pp. 592–599.
[32] M. Rathore, A. Ahmad, A. Paul, and G. Jeon, "Efficient graph-oriented smart transportation using Internet of Things generated big data," in *Proc. 11th Int. Conf. Signal-Image Technol. Internet-Based Syst. (SITIS)*, 2015, pp. 512–519.
[33] W. Hussain, J. M. Merigo, H. Gao, A. M. Alkalbani, and F. A. Rabhi, "Integrated AHP-IOWA, POWA framework for ideal cloud provider selection and optimum resource management," *IEEE Trans. Services Comput.*, early access, Nov. 13, 2021, doi: 10.1109/TSC.2021.3124885.
[34] T. Wang, H. Ke, X. Zheng, K. Wang, A. K. Sangaiah, and A. Liu, "Big data cleaning based on mobile edge computing in industrial sensor-cloud," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1321–1329, Feb. 2020.
[35] C. K. Leung, D. Deng, C. S. H. Hoi, and W. Lee, "Constrained big data mining in an edge computing environment," in *Proc. Int. Conf. Big Data Appl. Services*, 2017, pp. 61–68.
[36] A. C. Nicolaescu, S. Mastorakis, and I. Psaras, "Store edge networked data (SEND): A data and performance driven edge storage framework," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2021, pp. 1–10, doi: 10.1109/INFOCOM42981.2021.9488804.
[37] A. Azad, M. Washik, S. Shannigrahi, N. Stergiou, F. R. Ortega, and S. Mastorakis, "CLEDGE: A hybrid cloud-edge computing framework over information centric networking," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, 2021, pp. 589–596.
[38] M. A. Jan *et al.*, "Lightweight mutual authentication and privacy-preservation scheme for intelligent wearable devices in industrial-CPS," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5829–5839, Aug. 2021.
[39] "UCI Machine Learning Repository." [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/ (Accessed: Apr. 1, 2021).
[40] "Dataset Collection," Traffic Dataset. [Online]. Available: http://iot.ee.surrey.ac.uk:8080/datasets.html/traffic (Accessed: 2021).
[41] "Dataset Collection," Pollution Dataset. [Online]. Available: http://iot.ee.surrey.ac.uk:8080/datasets.html/pollution (Accessed: 2021).
[42] "Dataset Water Meters," [Online]. Available: http://data.surrey.ca/dataset/water-meters (Accessed: 2021).

Muhammad Babar received the Ph.D. degree in computer software engineering from National University Sciences and Technology, Islamabad, Pakistan, in 2018.

He is currently associated with Allama Iqbal Open University, Islamabad, Pakistan. He has published his research work in various reputed IEEE, Elsevier, Springer, and other leading journals. His research areas include but not limited to big data analytics, machine learning, IoT, fog/edge computing, smart city design and planning, and security.

Mian Ahmad Jan (Senior Member, IEEE) received the Ph.D. degree in computer systems from the University of Technology Sydney, Ultimo, NSW, Australia, in 2016.

He is an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Mardan, Pakistan. He has published his research in the world-leading journals and conferences. His research interests are Internet of Things, wireless sensor networks, edge computing, and vehicular communications. He works mainly in energy-efficient and secure communication for these networks.

Dr. Jan has been a general chair of various high-ranked conferences.

Xiangjian He (Senior Member, IEEE) received the Ph.D. degree from the University of Technology Sydney, Sydney, NSW, Australia, in February 1999.

He is a Professor with the University of Technology Sydney, where he is the Leader of the Computer Vision and Pattern Recognition Laboratory, Global Big Data Technologies Center. He has been the recipient of numerous competitive national or regional grants. He has successfully supervised several postdoctoral research fellows and many Ph.D. students as a Principal Supervisor. His research work focuses on computer vision and network security.

Muhammad Usman Tariq received the Ph.D. degree from California Southern University, Irvine, CA, USA, in 2017.

He has more than 15 years' experience in industry and academia. He is a first inventor for four scientific patents. He has been working as a standards consultant and a trainer for various industries.

Spyridon Mastorakis (Member, IEEE) received the 5-year Diploma (equivalent to M.Eng. degree) in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 2014, and the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2017 and 2019, respectively.

He is an Assistant Professor of Computer Science with the University of Nebraska Omaha, Omaha, NE, USA. His research interests include network systems and protocols, Internet architectures, IoT and edge computing, and security.

Ryan Alturki (Senior Member, IEEE) received the Ph.D. degree from the University of Technology Sydney, Ultimo, NSW, Australia, in 2019.

He is currently an Assistant Professor with the Department of Information Science, College of Computers and Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia. He published several publications in high-ranked international journals, conferences, and chapter of books. His research interests include eHealth, mobile technologies, Internet of Things, artificial intelligence, cloud computing, and cybersecurity.