Integrating Intelligence at the Periphery with Deep Learning: Edge-Cloud Computing for IoT Data Analytics

[1]Diyar Ali Kadhim, [2]Thikra Jaryan Abbas
[3]Seyed Ebrahim Dashti "corresponding author",
[1,2]Department of Computer Engineering, Shiraz Branch, Islamic Azad University
[3]Department of Computer Engineering, Jahrom Branch, Islamic Azad University

ABSTRACT:
There is a deluge of data flowing across the network as a result of the proliferation of various linked devices, such as sensors, mobile, wearable, and other IoT devices. Machine learning (ML) operations often include moving data from the internet of things (IoT) to a central server, which increases network traffic and introduces delays. By bringing processing closer to the periphery of the network and the data sources, edge computing may solve such problems. However, ML tasks are not well-suited to edge computing because to its restricted processing capability. Thus, the purpose of this article is to explore ways to integrate cloud and edge computing in order to analyse data from the Internet of Things (IoT) by making use of edge nodes to minimise data transfer. Feature learning is executed on the adjacent edge node to process data close to the source, once sensors are grouped according to locations. We also take similarity-based processing into account while making comparisons. Machine learning is used to carry out feature extraction. The learned autoencoder's encoder component is stored on the edge, while the decoder component is stored in the cloud. Human activity recognition using sensor data was the task that was evaluated. The findings demonstrate that sliding windows, when used during the preparation phase, allow for data reduction on the edge of up to 80% without noticeably compromising accuracy

KEYWORDS: Cloud computing, Edge computing, Intelligence, IoT, Deep Learning.

Introduction

From 18 billion in 2017 to more than 28 billion in 2022, according to CISCO's projections [1]. With approximately 14.6 billion units, more than 50% of those devices will be interconnected amongst machines. The proliferation of connected devices and the deluge of data they produce will cause a rise in network traffic. After reaching 1.5 zettabytes in 2017, Cisco predicts that yearly worldwide internet traffic will reach 4.8 zettabytes by 2022 [1]. This will have both good and bad effects on the communication infrastructure. On one hand, it will lead to the development of new applications and services and the expansion of current ones. On the other hand, it will raise the demand for network bandwidth and strain the system.

With the help of the Internet of Things (IoT), everyday objects may share information about their surroundings with one another and with the rest of the world online. The capacity to analyse this data is crucial to the realisation of smart systems like smart cities, smart healthcare, smart transportation, and smart energy, which are supported by the IoT [2]. However, environmental monitoring and data transmission to a more powerful system, typically the cloud or a centralised system, for processing and storage have traditionally been the primary functions of most IoT edge devices, like sensors, since they lack the computational capabilities to execute complex data analytics computations [3].

ISSN: 2173-0695

Accordingly, conventional Internet of Things (IoT) data analytics include uploading data to a cloud service, having that data analysed, and then providing the results to yet another device. An example of this would be sending process monitoring data from a smart factory to a data centre hundreds of miles away, where it would be stored and optimised. Then, the factory would get the findings. Both network traffic and data transfer latencies are increased by this procedure. Unfortunately, the linked devices do not have the processing power to do data analytics computations on their own.

By integrating cloud and edge computing, we can do complex data analytics without increasing the load on the Internet of Things (IoT) networks or the delays that come with it. In order to decrease network traffic and latencies, edge computing (EC) moves processing to the network's periphery and data sources, rather than the cloud or a centralised system. Despite EC's widespread acclaim for applications like content distribution and mobile task offloading, its application to data analytics has been slow to catch on [6, 7].

A number of fields have seen recent success using deep learning (DL), such as picture categorization [8] and human activity recognition (HAR) [9]. Data analytics for the Internet of Things (IoT) benefit from DL's representation learning and data transformation capabilities, which allow for the learning of useful characteristics. One subset of DL, deep autoencoders (AEs) are NNs that have been unsupervisedly trained to learn data encoding. Learning the reconstruction alongside the encodings allows AE to recover its inputs from the compressed encodings, although at the expense of some data.

This article delves into the topic of Internet of Things data analytics by merging cloud and edge computing. The key findings include a decrease in network traffic and latency for ML jobs through the use of edge nodes and an assessment of the amount of data reduction that can be accomplished on the edge without significantly affecting the accuracy of ML tasks. The role of edge nodes is to reduce the amount of data transferred to the cloud by acting as middlemen between the cloud and Internet of Things devices. Edge computing makes use of the trained AE's encoder component to generate data encodings before sending them to the cloud. You can utilise encoded data straight for an ML job in the cloud, or you can use the decoder component of the AE to recover the original data before using it for the ML work.

This study investigates feature learning from fused data, data grouped by source locations, and data categorised according to sensor similarities, all of which are important considerations since IoT data might come from many sensors and places. In this test case, HAR is derived from data collected by sensors placed on various human body parts, such as gyroscopes and accelerometers. The results demonstrate that the proposed method may significantly cut transmitted data by 80% while maintaining high HAR accuracy. What follows is an outline of the rest of the article. Part II gives some context, Part III talks about previous work in the field, and Parts IV and V show the edge-cloud ML model and assessment technique, respectively. The findings are detailed in Section VI.
Section VII serves as the article's last section.

II- Context

In this part, we will go over DL-based dimensionality reduction and EC.

A.Edge Computing

Cloud computing and other centralised infrastructure systems store data, run business logic and do data analytics at a distance from both the consumers and the sources of the data. The pay-as-you-go pricing mechanism, minimal starting cost, high scalability, and dependability are some of the major advantages they provide. However, it is not practical or even viable to move all data to the cloud for processing due to zettabyte-sized traffic and the growth of linked devices. EC has arisen as a solution to these problems by relocating computing to data sources and the periphery of the network [6].
Many features of fog computing are similar to those of EC. Fog and edge computing is sometimes thought to mean the same thing [10], but edge computing is mainly concerned with nodes near Internet of Things (IoT) devices, and fog may refer to any resource between the end device and the cloud. For the purposes of this study, "edge" might be either the end devices themselves, such as

ISSN: 2173-0695

sensor nodes or cellphones, or the computing nodes that are physically near the network's periphery, like edge servers.

Reducing network traffic is the primary goal of edge computing, which is why it moves processing closer to the data sources. Mobile edge computing (EC) has been the subject of much research in mobile computing for its ability to decrease network latencies, enhance user experience, decrease battery usage, and bring location awareness by moving processing and data storage to the edge, such as base stations [11]. Content delivery and caching, as well as other applications requiring extremely low latency, are ideal for EC [12]. The edge computing resources are not up to pace with the cloud resources, despite the fact that EC offers the benefit of lower traffic. Therefore, edge devices aren't the best choice for computationally heavy applications like ML. However, by doing some computation on the edge, cloud computing may be supplemented, resulting in reduced network traffic and delays.

B.Deep Learning

Deep learning (DL) is a subset of machine learning (ML) that relies on models with several computational layers to learn data representations at varying degrees of abstraction [13]. Many disciplines have shown success with DL, including vision tasks, audio recognition, and natural language processing, thanks to its representation capabilities, capacity to train complicated models, and diversity of architectures [2].

For the purpose of unsupervised learning of data representations (encodings), AEs are a subset of DL methods. The core of an AE is a layer of bottleneck neural networks (NNs) that prohibit the network from just replicating the input to output and instead force it to learn data representations, thereby making it an NN that learns to recreate its inputs. Both the encoder and the decoder, which may each have several stacked layers, make up an AE. Typically, the number of neurons decreases from the input layer all the way to the last encoder layer since the encoder section of the network is responsible for lowering dimensionality (encoding). On the other hand, layers with a growing number of neurons make up the decoder, which is responsible for reassembling the input signal from encoded information. While AEs have several potential applications, including as anomaly identification and noise reduction, they are most commonly employed as a preprocessing step in the ML pipeline [2]. By feeding the encoder network's outputs (encodings) into another ML model, the learned AE may be utilised for dimensionality reduction.

Prior to data transmission to the cloud, the learned AE's encoder component is placed on the edge in order to decrease dimensionality. Additionally, the decoder is used in the cloud to restore the initial signal.

Related Work

Computing proximity to data sources has led to EC's rising popularity, particularly in applications that demand quick reaction times or have restricted bandwidth. Smart street lighting [14], face identification [15], smart manufacturing [16], and vehicular networks [17] are only a few examples of the many EC applications that have been widely adopted and studied.

Concerning computation, edge offloading, and communication methods for edge-based computing, Wang et al. [18] offered a review of mobile edge networks. Internet of Things (IoT), linked cars, content distribution, and big data analysis are some of the application cases that their study emphasises. Meanwhile, real-time analytics was named as one of the unanswered questions by Wang et al. [18]. In a similar vein, Abbas et al. [12] examined mobile EC and similarly recommended big data analytics for further study. Although Wang et al. [18] and Abbas et al. [12] concentrated on EC in mobile devices, El-Sayed et al. [6] emphasised its usage in the Internet of Things. After looking at EC, multi-cloud, fog, and cloud characteristics side by side, they found that EC had the best latencies and lowest bandwidth utilisation. Like Wang et al. [18] and Abbas et al. [12], Mao et al. [19] consider data analytics to be a promising area for future study in EC, and they also view EC as an essential enabling technology for the Internet of Things (IoT) concept.

ISSN: 2173-0695

Discussion in the polls [6, 12, 18, 19] focuses on the use of EC in data analytics and how crucial it is for the Internet of Things (IoT) to manage the exponential growth in the number of connected devices. Through the integration of cloud and edge computing, our research adds to the use of EC for data analytics in the delivery of ML applications.

A widely-discussed use case and application of EC is smart cities. Cloud and fog-enabled smart city services were investigated by Mohammad et al. [20] in relation to service-oriented middleware. They centred on the middleware rather than the particular smart city services. The advantages of EC in relation to reaction time were shown in their trials. A hierarchical fog computing architecture was introduced by Tang et al. [21] to assist linked devices in smart cities. The suggested model incorporates the cloud as the uppermost layer in addition to the fog node hierarchy. Preliminary results showed that the suggested architecture was feasible when tested on an event detection job in a smart pipeline monitoring system. Though we use cloud and edge/fog technologies, our research varies from those of Mohammad et al. [20] and Tang et al. [21] by including a thorough assessment of the proposed edge-cloud architecture.

Wang et al.'s [22] research on using cloud and fog computing together for real-time traffic control is another instance of the cloud-fog concept. Their technology is structured such that automobiles communicate with the cloud as edge nodes and roadside equipment as cloudlets. While they address message passing and processing in their study [22], we address ML for the Internet of Things in our work.

on their proposal for big-scale analytics on Smart City IoT data, he et al. [23] laid up a multi-tier fog computing approach. Fogs can enhance the performance of smart city services, according to their evaluation of the suggested model on categorization tasks. In contrast to He et al. [23], which focuses on fog architecture, our study utilises both the edge and cloud to accomplish the ML goal.

Fog computing has other uses in the medical field as well. By integrating intelligence between sensors and the cloud, Rahmani et al. [24] introduced a fog-assisted architecture for smart e-Health. Data filtering, compression, fusion, and analysis may be handled by fog nodes, according to their study, with minimum data transferred to the cloud because of their impressive capabilities. However, a significant portion of the computation for our study is still carried out on the cloud. Another group that addressed healthcare was Ritrovato et al. [25], who suggested a method for EC anomaly detection in streaming data. Our research is on ML algorithms, whereas theirs is on stream processing methods. Our approach integrates edge and cloud computing for ML tasks, which sets it apart from the examined research. While not all of these research used edge-cloud architectures for ML tasks, several did [20], [22], and [24]. In order to complete ML jobs in the cloud, processing must occur on the edge, but network traffic must be kept to a minimum. Tang et al. [21] is the closest work to ours in terms of edge ML and feature extraction; however, they restrict their feature extraction to the signal's mean and variance, which limits the applications they may examine, in contrast to our general method based on AEs. In addition, we assess the extent to which feature reduction is feasible without substantially affecting ML accuracy.

Several studies in this area should be included as our study assesses the offered method on sensor-based (HAR). The study of Wang et al. [26] reviewed DL techniques for activity identification, highlighting the relevance of model selection and preprocessing, including the sliding window technique, given that DL has been widely utilised for HAR [26] and has been highly effective. In their study on feature engineering for HAR, Zdravevski et al. [27] collected 3,232 features from the MHEALTH dataset and subsequently reduced them using a combination of feature reduction strategies. Sliding windows and feature creation were also utilised in the investigation of HAR model personalisation by Ferrari et al. [28]. After extracting 118 characteristics and creating segments of predetermined size, Li et al. [29] set out to recognise activity transitions. The next step was to examine each segment individually to see whether any activity had changed inside it.

Both EC and network traffic are unrelated to these HAR experiments. We follow the trend of other

ISSN: 2173-0695

HAR research that employ the sliding window approach [26]-[28] because of how much better it makes the results. We study the effect of the sliding window approach on edge data reduction, whereas others focus on its influence on accuracy. While prior research has shown that AE and PCA can increase HAR accuracy, our study finds that they actually perform better at reducing network traffic.

IV- Model of the Edge-Cloud ML system

Figure 1 shows the general layout of the edge-cloud ML model, and the pieces that make it up preprocessing, data reduction, and Cloud ML are broken down into their respective parts below.

A. Initial Steps

Unlike standard ML, which sends data straight to the cloud, the edge-cloud ML process begins by passing data from IoT sensors to the edge for preprocessing. Although the sliding window approach is experimentally assessed for its potential influence, normalisation is an integral part of the preprocessing.

No. 1: Standardisation Data is normalised using standardisation (z-score) to prevent features with big values from dominating and to increase training convergence. Although min-max scaling is an alternative to standardisation, the latter was chosen for its robustness against outliers. All features are resized such that their means are zero and their variances are one, as specified by

$$\hat{x} = \frac{x - \mu}{\sigma}$$

where
the original feature value is represented by x, the feature mean and standard deviation by μ and σ, and the normalised value is denoted by xˆ.
2) Sliding Window: Currently, a data sample for a given time step t may include readings from several sensors in various places. The Windows sliding approach is used to prepare time series data for ML algorithms or to assist the model grasp time dependencies [30]. The sliding window is a highly effective tool in HAR [30]. This research looks on the possibility of feature reduction following the application of the sliding window approach and how it affects the amount of data reduction. For each sample, the l × f matrix is produced by combining all readings from the first l time steps with the f-number of features and a sliding window of length l. The second sample includes data from time step k to k + l, after which the window slides for k steps. The remaining samples are created as the window continues to slide. To increase the amount of training samples and enable the input to catch changes in temporal patterns, this study uses the sliding step k = 1. After training the system, several sliding phases can be implemented according to use case details and data transit limitations.

B. Reducing Data

To lessen the load on the cloud, data reduction takes place locally. Choosing the method and the quantity of characteristics are two of the obstacles to feature reduction. While ML precision is the main factor in centralised systems, network traffic is an additional consideration in an edge-cloud setting. It is still difficult to find the optimal trade-off between feature reduction and traffic reduction without sacrificing ML task accuracy, even if AEs are ideal for edge feature learning. Normalised data or samples generated using the sliding window method can be used for data reduction directly. There is no difference in the data reduction strategy between using the sliding window technique and not using it. Reversible and nonreversible methods are both taken into account.

1) Reversible: Methods that may be used to decrease data while still retaining the capacity to replicate the original data are known as reversible techniques. These methods use edge computing to reduce data before sending it over the network. In the cloud, either the reduced data or the original data can be subjected to machine learning (ML), or both can be done simultaneously. Since trained

ISSN: 2173-0695

AEs allow encoders to minimise data on the edge and decoders to recover original data on the cloud, they are the primary focus here. The efficiency of AE is contrasted with that of PCA for data reduction [31]. The eigenvectors allow for the reconstruction of the original data when principal component analysis (PCA) is used to decrease dimensionality. Data reconstruction accuracy in AE and PCA is proportional to the dimensionality reduction level.

All sensors, location-based, and similarity-based situations are evaluated for both reversible approaches, AE and PCA, as shown in Fig. 1. When using an all-sensors technique, all of the data is combined and then data reduction is applied to the combined dataset.

As shown in Figure 2, the location-based scenario takes into account data reduction using a cluster of sensors that are colocated. As an example, the four sensors at site 1 are in close proximity to one other, therefore their data is relayed to the edge node E1. The data from position 2 sensors is also sent to E2. The goal is to minimise the distance data must travel before reduction by keeping the edge portion of the processing near to the data sources. As a result of a shared reduction of data at each receiving node, each edge node has its own AE.

Sensors are grouped according to how similar they are in the similarity-based scenario shown in Figure 3. All accelerometers may stand for one set, and all gyroscopes for another. This can lead to sensors being further away from the edge nodes, but it will also produce more consistent data sets. Just as in the location-based scenario, every edge node has its own AE.

2) Nonreversible: Methods that do not allow for the restoration of the original data set following data reduction are considered nonreversible. In this case, we take into account the vector magnitude, which is appropriate for sensors like gyroscopes and accelerometers that measure values in three-dimensional space. Here is how the dimensionality is reduced:

$$d = \sqrt{x^2 + y^2 + z^2}.$$

The
vector magnitude is denoted by d, and the three variables x, y, and z are the measurements in Euler coordinates. Consequently, there is a 3:1 ratio between vector magnitude and dimensionality reduction.

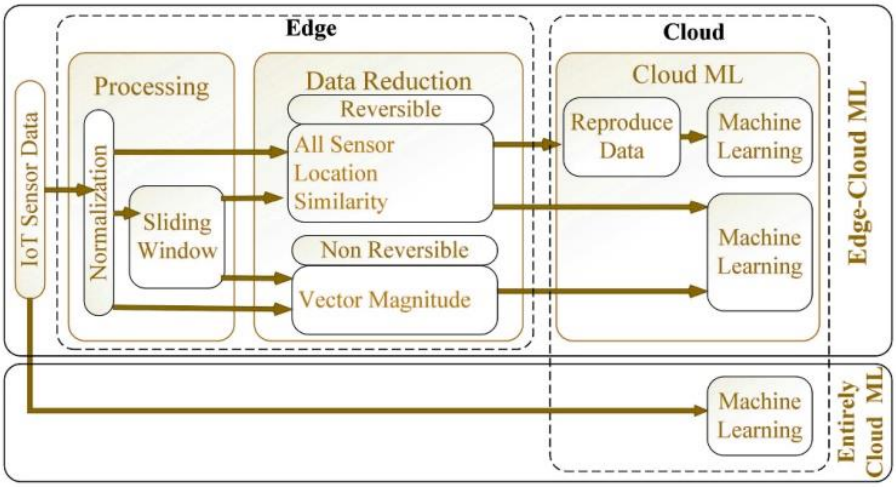

Figure 1: The architecture of the edge-cloud ML system.

ISSN: 2173-0695

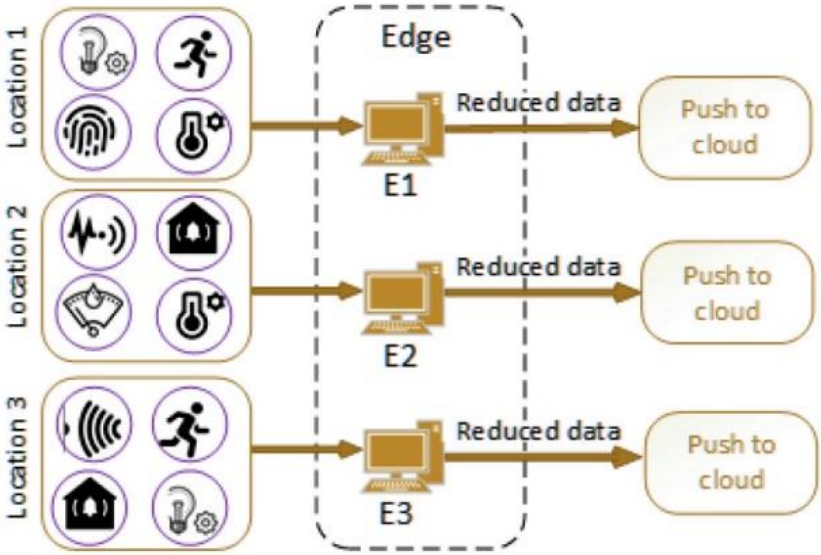

Figure 2: Reducing data depending on location.

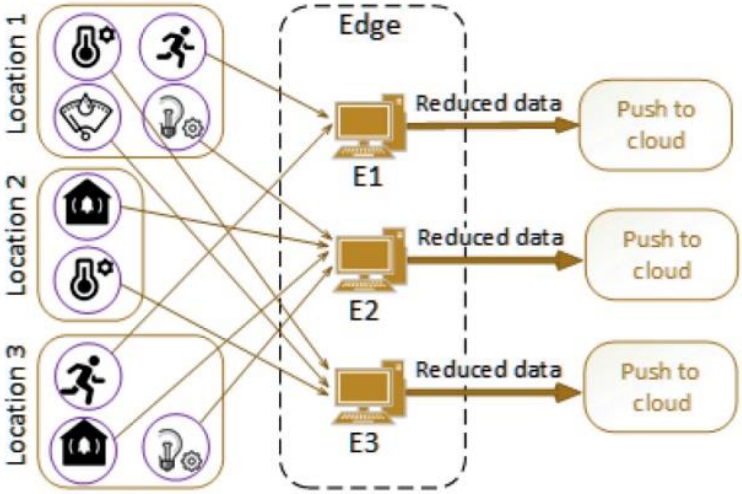

Figure 3: Data reduction based on similarities.

C. Machine learning on the cloud

After data reduction at the edge, it is transferred to the cloud to undergo additional ML processing. Figure 1 shows that the ML work may be executed in two different methods. Reproducing the original data and using it for the ML job is the first alternative. Assuming the reversible approach was employed for data reduction, this becomes feasible. Another choice, applicable to both reversible and nonreversible reduction methods, is to perform the ML job directly on the reduced data. Training the ML model using the recreated data is computationally more expensive than with the reduced data because to the increased number of characteristics in the reproduced data.

V- Approach to Assessment

The dataset and assessment technique are presented in this section.

A. Datasets and Preprocessing

The proposed method was tested using the HAR task and the (Mobile Health) MHEALTH dataset [32]. With the use of sensors, we can identify the many sorts of activities, including walking, running, or sitting. Ten people's motion captures from various activities are housed in the MHEALTH dataset.

ISSN: 2173-0695

Each of the three sensor types accelerometer, gyroscope, and magnetometer obtained three readings along each of the three axes that make up the recordings. While the accelerometer is the only sensor on the chest, the other two are attached to the left and right wrists. A grand total of 21 characteristics are thus achieved: Seven sensors along three dimensions. Twelve distinct types of physical activity labels are used, with a sample rate of fifty hertz. Throughout the studies, data from all people is combined and used in this manner. The data was divided 80:20 across the training and test sets, accordingly. Preprocessing can proceed with or without the sliding window, as shown in Fig. 1, following data normalisation. We take a look at three different window sizes 25, 50, and 100 time steps to see how they affect the accuracy of activity detection and the achievable decrease rate. Such dimensions are 1/2, 1, and 2 s at a sampling rate of 50 Hz.

B. Reducing Data

We take into account both reversible and nonreversible edge data reduction techniques.

1) Reversible: A total of three scenarios all sensors, location-based, and similarity-based are tested using reversible data reduction options, with and without a sliding window. With a 100-window-size window, the reversible approach yielded the same amount of features in all three cases (Table I). "Direct" is used to describe the method that does not include the sliding window in the table and the rest of the article.

There were trials conducted for reductions of80%,90%, and 95%; the table shows the number of characteristics before and after reductions of 70% and 66%. We chose a 66% initial reduction to align with the maximum reduction of the vector magnitude technique, which is 3:1. Because the whole activity takes place on only one node, there is no such thing as an edge position for any sensor. Edge location specifies a node where data is gathered for location-based and similarity-based methods. Three nodes represent sensors on the arm, leg, and chest in the location-based method. Nodes in the similarity-based method represent different kinds of sensors, such as accelerometers, gyroscopes, and magnetometers. As seen in Table I, the number of original features for the direct approach is nine, as an example, because L2 in location-based approaches collects information from three sensors. This is because each sensor has three axes. As an example, with a location-based method with L2, the number of features is 9 × 100, which is the product of the number of features and the length of the window in a scenario with windows.

Keep in mind that a data reduction rate of 66% is the minimum acceptable for using the direct option. There are currently relatively few characteristics at that value for instance, seven for all sensors and any further decrease would be bad. Since sliding-window techniques do buffering locally and transmit only compressed data to the cloud, the feature count is substantially larger. Approaches using the sliding window necessitate edge buffering, leading to a significantly higher feature count and the prospect of higher reduction rates. Given its track record of performance in HAR, the sliding window approach is also likely to yield improved accuracy [26].

You can see that AE and PCA are used for every data reduction variation in Table I. The data reduction rates shown in Table I are consistent regardless of whether PCA or AE is utilised. In AE, the reduction degree is modulated by adjusting the bottleneck layer neuron count, whereas in PCA, it is modulated by adjusting the number of main components. Each edge node has an associated AE that is in charge of lowering the data coming at that node in the location-based and similarity-based methods. Likewise, PCA operates autonomously on every node.

Three situations, three window widths, five reduction rates, and two algorithms (AE and PCA) comprise a total of ninety data reduction tests with sliding windows. Furthermore, six experiments are available for direct reduction: 2-algorithms per scenario. With this, 96 data reduction trials have been conducted.

2) it can't be undone; the vector magnitude is the sole method that doesn't allow for that. This leads to a decrease of 3:1 when dealing with three-axis data, such as that captured by an accelerometer, gyroscope, and magnetometer. The reduction can be applied with or without the sliding window, just

ISSN: 2173-0695

as reversible techniques. There are seven characteristics instead of twenty-one when the window is removed from the vector magnitude for all data. Reduced characteristics for sliding windows 25, 50, and 100 are 175, 350, and 700 instead of 525, 1050, and 2100. Three trials for each window size plus one for direct reduction (no sliding window) yields a total of four.

C. Machine learning on the cloud

Sending data reduced on the edge to the cloud for final processing is the standard procedure. This last phase of the HAR task is activity type recognition, often known as classification. A single model in the cloud integrates data from all edge nodes, in contrast to the distributed architecture of the edge, where individual ML models analyse data as it arrives at each node. As shown in Figure 1, we will now examine three methods for doing this task:

1) ML with Reduced Data: Nodes at the edge provide their reduced data straight to the classification process. This method works for reduction strategies that can be undone as well as those that can't. A feed-forward NN (FFNN) was employed for the last classification in the HAR challenge. Twelve (12 actions) output nodes make up the FFNN, whereas the number of input nodes decreases as the reduction rate increases and is proportional to the number of features remaining after data reduction. The amount of input features determined the number of hidden layers and neurons employed. For input features greater than 350, FFNN had three hidden layers; for input features less than 350, it had two hidden layers. For each of the 420 input characteristics, the hidden layer node count reduced progressively; for instance, from 420-128-32-12. This method demonstrated good accuracy in experiments with varying numbers of neurons in the buried layer.

2) ML With Reproduced Data: First, you reduce the original data to a more manageable form, and then you recreate it. Then, you categorise it. This works only when data reduction on the edge is done using reversible approaches. Because the amount of features here is equal to the number of features in the original dataset, this method is computationally more expensive and needs more complicated models compared to ML with reduced data. The data that is reproduced, however, might serve different purposes much like the original data. Classification is done using the FFNN in the same way as with ML with decreased data; the same approach is used for selecting the amount of layers and neurons in hidden layers. A larger number of layers and neurons will be formed by using the same technique with replicated data rather than reduced data, due to the higher number of characteristics with the former.

3) ML Runs Wholly on the Cloud: This isn't an edge-cloud architecture, but rather a tried-and-true method of ML that relies on the cloud. Only for the sake of comparison is it taken into account in the evaluation. Classification once again makes use of FFNN, and the same method as with edge-cloud techniques was utilised to determine the number of layers.

The assessment takes into account 96 trials conducted for reversible procedures, as mentioned in Section V.B.1. A total of 192 tests utilising reversible procedures were conducted using two cloud-based approaches: one with decreased data and the other with repeated data. Section V.B.2 has four experiments with nonreversible techniques, whereas Section V.B.2 contains four experiments involving totally cloud-based ML, one of which is direct and three of which are sliding windows. There will be a grand total of 200 trials conducted.

Table I:Features for Reversible Approaches Before and After Reduction (WINDOW SIZE 100)

ISSN: 2173-0695

Scenario	Edge Location	Original Features	66% Reduction	70% Reduction
All Sensors				
Direct	-	21	7	-
S.Window	-	21 × 100	693	630
Location Based				
Direct	L1	3	1	-
	L2	9	3	-
	L3	9	3	-
S.Window	L1	3×100	99	90
	L2	9×100	297	270
	L3	9×100	297	270
Similarity Based				
Direct	S1	9	3	-
	S2	6	2	-
	S3	6	2	-
S.Window	S1	9×100	297	270
	S2	6×100	198	180
	S3	6×100	198	180

VI- Results and Discussion

Prior to delving into the conclusions, this part offers the results of data reduction and network traffic analysis.

A. Reducing Data

Due to their prevalence in HAR research [27], in which TP and TN denote true positives and true negatives, and FP and FN denote false positives and false negatives, respectively, accuracy, precision, and recall are utilised for the evaluation.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

As a first step in facilitating comparisons, Table II displays the accuracy of conventional ML that does not employ data reduction. The accuracy of classification for reversible approaches is displayed in Table III, following a data reduction of 66% and a sliding window size of 100. Regardless of the algorithm (AE or PAC) or situation (all sensors, location-based, or similarity-based), the sliding window technique provides superior accuracy than direct techniques, much as traditional ML (see Table II). The reduced and reproduced techniques, as well as AE and PCA, are essentially interchangeable. Despite a 66% decrease in data, the accuracy is comparable to that of conventional cloud-based ML. Along with accuracy, precision and recall were computed, and they exhibited patterns that were comparable to those in Table III.

ISSN: 2173-0695

Table II: Entire Cloud Machine Learning for Accurate Classification

	Direct	Window 100	Window 50	Window 25
Accuracy	98.94%	100%	100%	99.99%

Table III: 66% Data Reduction, Reversible Approaches, and Sliding Window 100 for Classification Accuracy

Scenarios	AE		PCA	
	Reduced	Reproduce	Reduced	Reproduce
All Sensors				
Direct	98.27%	99.24%	98%	98.17%
S.Window	100%	100%	100%	100%
Location Based				
Direct	98.13%	98.44%	99.4%	99.38%
S.Window	99.89%	99.92%	99.89%	99.86%
Similarity based				
Direct	98.74%	99.24%	99.32%	99.34%
S.Window	99.90%	99.92%	99.86%	99.85%

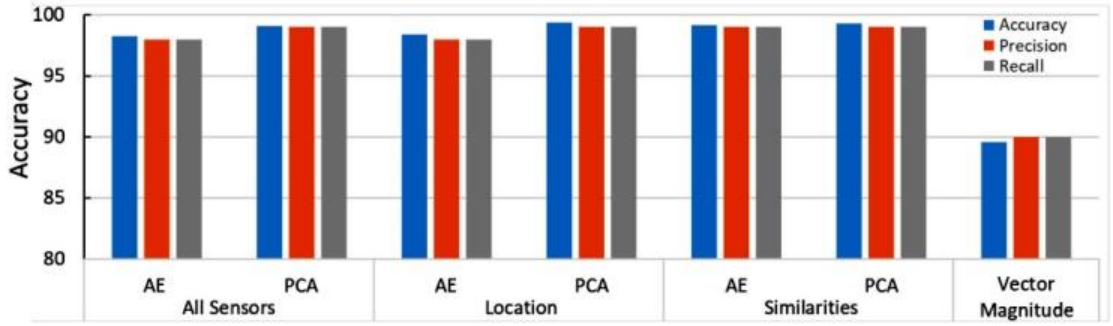

Figure 4:Methods for accurate classification: direct (no Window), reversible (with a 66% reduction), and nonreversible (with no reduction).

In contrast to Table III, which just covers reversible ways, Fig. 4 incorporates both reversible and nonreversible methods, although both deal with the 66% decrease. When compared to the nonreversible vector magnitude method, all reversible methods provide substantially better results.
Figure 4 and Table III both examine the accuracy of classification with a 66% reduction in data, with Table III focusing exclusively on sliding window 100. In order to examine how data reduction rates affect accuracy, Fig. 5 shows how accuracy varies in relation to the reduction rate for all sensors, location-based, and similarity-based situations. From 66% to 90% reduction, accuracy declines gradually, and at 95% reduction, the decline is more dramatic. On the whole, AE outperforms PCA in terms of accuracy, and classification using replicated data is marginally superior than reduced data. Data reproduction somewhat improves HAR accuracy by making advantage of information already present in the encoder portion of AE or the PCA eigenvectors to enlarge the encoded values. The accuracy decreases by less than 0.25% compared to classical ML, even at a 95% reduction rate.
The assessment for window size 50 is shown in Figure 6, which is identical to Figure 5. Sliding window 50 causes a decline in accuracy with changes to the reduction rate, in contrast to window size 100, where the accuracy was fairly constant for reductions ranging from 66% to 80%. While both

Windows 100 and 50 achieve 95% reduction accuracy, Window 50 is preferable due to its shorter FFNN training time and reduced edge data gathering requirements. When comparing methods with different window sizes, AE using recreated data performs marginally better with a 50-size window compared to the others.

Figure 7 next displays the accuracy for a 25-pane window. Classification using reduced and reproduced data yielded comparable results with 100 and 50 pixel windows, however with 25 pixel windows, reduced data methods generate worse results than reproduced data approaches across the board for both PCA and AE algorithms. The most accurate reduction rates were nearly always attained by using recreated data in AE, as was the case with other window widths. Results for AE using recreated data from windows 25 and 100 are comparable; however, window 25 may be preferable due to its shorter training period, whereas window 100 exhibits somewhat superior accuracy.

While Figs. 5-7 illustrate how reversible techniques fare with varying window widths, Fig. 8 shows how sliding window approaches fare against direct ones, which is as crucial. This graphic compares direct techniques to sliding window approaches with the same reduction rate, while direct approaches are only examined for 66% reduction rates. No matter the size of the window, all sliding window methods perform better than direct approaches, further proving that sliding windows are beneficial. Reducing the window size to 25 reduces accuracy for reduced data, as shown in Figures 5-8, while increasing the window size to 100 or 50 achieves equivalent accuracy.

Fig. 9 finalises the discussion by showing the accuracy of nonreversible techniques for various window widths. Similar to reversible techniques, 100 and 50 window sizes produce comparable results, whereas 25 window size yields poorer accuracy. The accuracy was shown to increase by about 10% when sliding windows were introduced to the non-reversible technique with windows (see Fig. 9) compared to the identical approach without windows (direct) (see Fig. 4). The nonreversible vector magnitude methodology achieves reversible-level precision with sliding windows; nonetheless, vector magnitude is computationally far easier than the other methods.

B. Data Transfer Over a Network

So far, the investigation has taken data reduction into account in relation to many characteristics. But because to network overheads, real traffic won't precisely track the feature reduction rates once the system is up and running. The edge-cloud architecture that was described has been used to mimic network traffic analysis. As mentioned in Section IV-B, location and similarity-based situations necessitate three nodes, one for each location/similarity group, but the direct technique only requires one edge node. An individual virtual computer with 2 GB of RAM and a single-core CPU is used to mimic each edge node. An octa-core processor and 6 GB of RAM on a Windows server were used to mimic the cloud. Sockets allowed for the measurement of bandwidth consumption once the TCP-IP protocol had been used to establish connection between the server and the edge nodes. The ability for several edge nodes to connect with the server at the same time is guaranteed by a multithreaded socket server. Nodes near the network's periphery house the trained AEs' encoders, while servers host the decoders and classification models. Network traffic would remain unchanged regardless of changes to the edge nodes' computing capability; nevertheless, this capacity must be adequate to perform data encoding. The sliding window size determines the minimal computational resources needed at the edge, which are based on the size of the encoder network.

Table IV shows the comparison between the network traffic in each scenario with and without 66% feature reduction. Additionally, it provides data on the percentage of traffic reduction, which shows the amount of traffic reduced compared to the identical situation without reduction, as measured in years. Take, for instance, a 49.71% drop in traffic for all sensors during sliding window 100. This would result in a decrease from 7129 MB for no reduction to 3585 MB for all sensors during sliding window 100. The network traffic in the three cases (all sensors, location-based, and similarity-based) is quite consistent with one another. Network traffic is significantly greater for situations with sliding windows compared to direct approaches, regardless of whether the data is decreased or not.

ISSN: 2173-0695

This is because the transfer includes the same values more than once since various sliding windows provide readings with a sliding step size of k = 1. If the application permits delays in window length, this may be circumvented by utilising the siding step that is proportional to the window's length.
Figure 10 shows a comparison of several data reduction strategies and scenarios for a 66% decrease in network traffic. Table IV's findings that none of the scenarios all sensors, location, and similarity significantly affect traffic reduction are also applicable here. Although all techniques and scenarios reduced features by 66%, different approaches reduced network traffic in different ways. The direct approach reduced network traffic by around 59%, and the reduction rate declined as the window size rose. When the total amount of data delivered varies, the resulting changes in network overheads are what create these fluctuations. There has to be a balance between the two, since window size 25 reduces network traffic but is less accurate than 50 and 100.

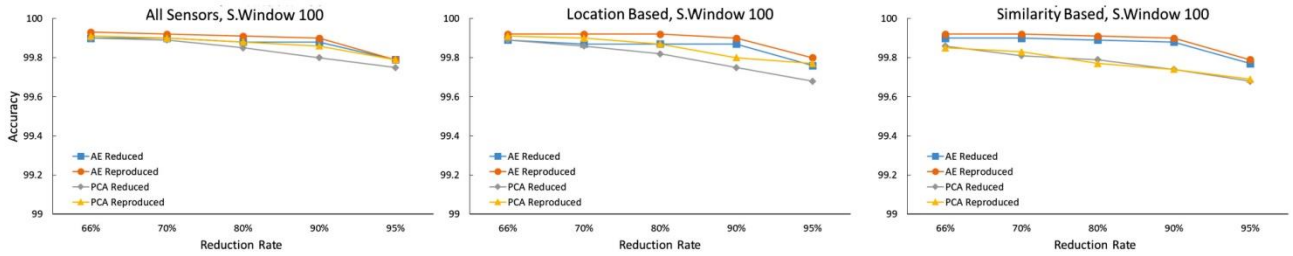

Figure 5:Accuracy of classification for various reduction rates: 100 window size.

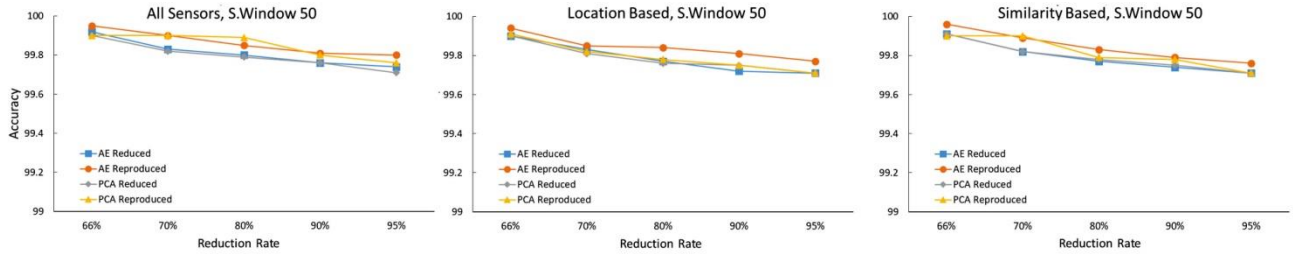

Figure 6:Accuracy of classification for various reduction rates: 20 window size.

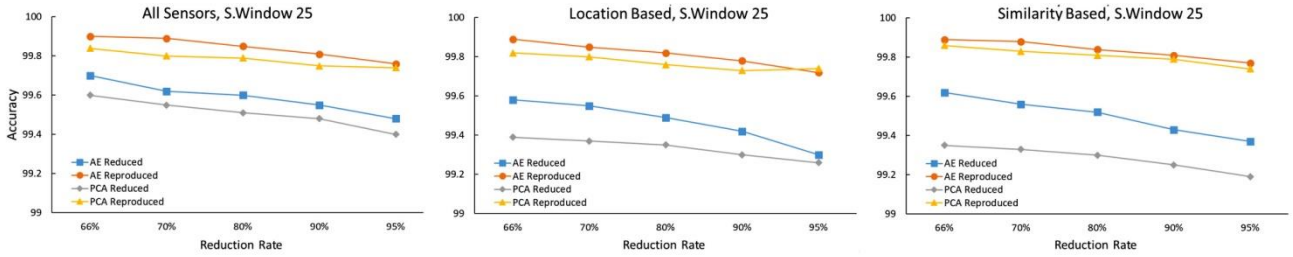

Figure 7:Accuracy of classification for various reduction rates: 25 window size.

ISSN: 2173-0695

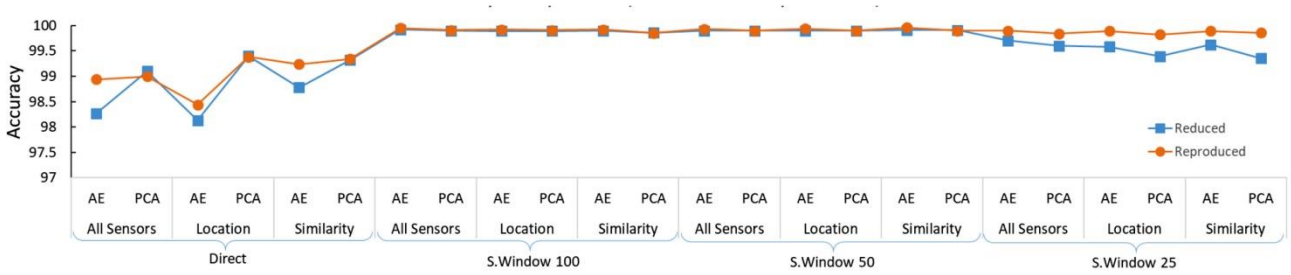

Figure 8:Improved classification accuracy (66 percent decrease rate) using direct and sliding window methods.

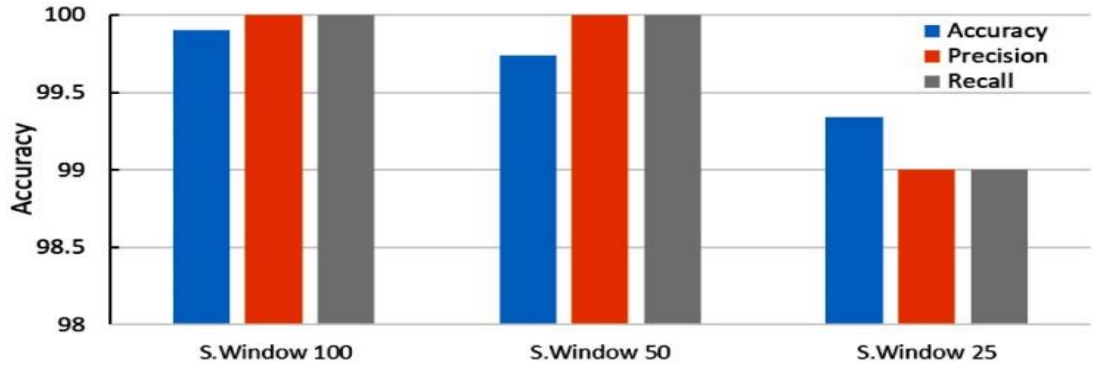

Figure 9:Precision in classification: magnitude of a vector with sliding windows.

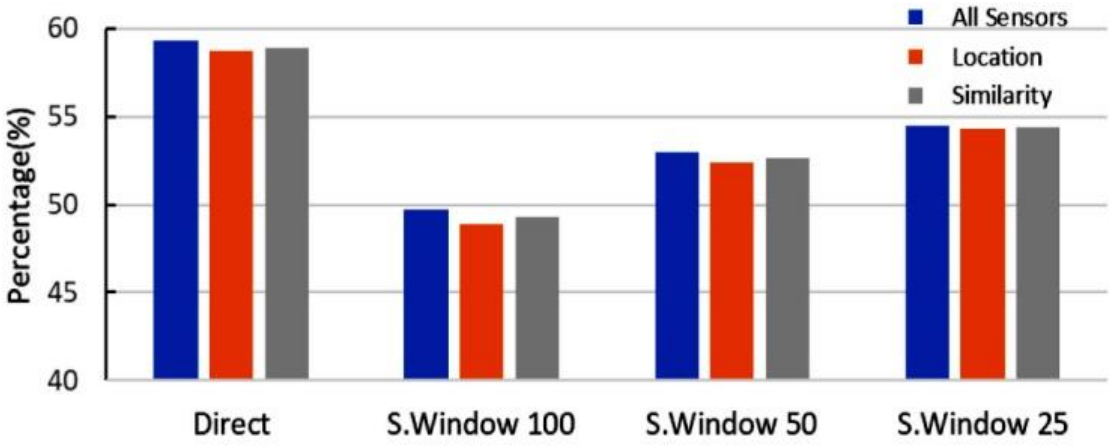

Figure 10:The mitigation of network traffic.

ISSN: 2173-0695

Table IV:No Reduction in Network Traffic and a 66% Decrease in Data

Scenarios	No Reduc.	Reduced		
		All Sensors	Location Based	Similarity Based
Consumed Bandwidth MB				
Direct	80.30	32.62	33.14	32.95
S.Window 100	7129.01	3585.40	3646.89	3612.23
S.Window 50	3548.76	1669.28	1690.51	1680.22
S.Window 25	1842.30	838.88	841.28	840.78
Traffic Reduction %				
Direct	-	59.38	58.73	58.97
S.Window 100	-	49.70	48.84	49.33
S.Window 50	-	52.96	52.36	52.65
S.Window 25	-	53.46	54.33	54.36

C. Discussion

The shown method can reduce data transmitted to the cloud by as much as 80% without substantially sacrificing accuracy, according to the experimental findings. Reducing data that has already been preprocessed using the sliding window approach does not diminish the accuracy achieved using the sliding window technique. The accuracy for all three cases (all sensors, location-based, and similarity-based) is lowered with a window size of 25, although it is equivalent for 50 and 100. Figure 5 demonstrates that accuracy remains reasonably constant up to a 90% decrease, indicating that with big enough sliding windows, even a 90% reduction only leads to a tiny loss of accuracy.

Data aggregation at the edge is a drawback of sliding window situations. Network traffic will remain high with sliding step one, even with reduced data delivered to the cloud on every time step, because sensor readings will correspond to separate windows. To save network traffic and ensure that readings do not belong to distinct windows, a sliding step that is equal to or bigger than the window size is used; nevertheless, the application scenario must account for a delay of window length.

Using recreated data for classification significantly outperformed using reduced data; this disparity became more pronounced for narrower windows (refer to Fig. 8). Both AE and PCA were equally effective for 50 and 100 window sizes, however AE achieved better results for all reduction rates when using a 25 window size (Figs. 5-7). This could be because, unlike PCA's linear transformations, which failed to capture complicated relations, AE's nonlinear transformations were able to do just that.

While both location- and similarity-based methods produced identical findings across all sensors (refer to Fig. 4), location-based methods allow for data reduction on separate nodes, which is a benefit. In addition, geo-distributed situations may benefit from the location-based strategy, which places edge nodes closer to data sources.

Table IV and Figure 10 show that, because of network overheads, the data reduction rate is higher than the rate of network traffic reduction. Also, because they cause a lot of traffic on the network, overlapping sliding windows should be avoided. When training networks, AE, and classification FFNN, overlapping windows are still acceptable; however, edge-cloud deployments should avoid them.

While this method was tested on the HAR task using the MHEALTH dataset, it is applicable to a wide range of IoT tasks and datasets. Data reduction rates will be high if sensor values are highly correlated with one another. In addition, the reduction rates for various applications and ML jobs will vary based

ISSN: 2173-0695

on the relative value of the data portions used for those particular activities.

VII- Conclusion

The old way of doing machine learning with data from the Internet of Things was moving the data to a central location, like the cloud, for processing. This would increase latency and strain communication networks due to the proliferation of linked devices.

In this piece, we looked at how to lessen network traffic and delays by combining IoT data with cloud and edge computing for machine learning. A location-based scenario grouped data according to the locations of the IoT devices, a similarity-based scenario grouped data according to the similarities of sensors, and an all-sensors-together scenario considered all the data at once. Two nonreversible methods, AE and PCA, as well as one nonreversible method, vector magnitude, were considered in the HAR task evaluation. The results shown that by utilising a broad sliding window during preprocessing, data and associated network traffic might be decreased by as much as 80% without a noticeable decrease in accuracy. While all sensor approaches achieved comparable accuracy, location and similarity-based methods might reduce on distinct edge nodes.

Applying the provided edge-cloud technique to other applications and sensor types will be the focus of future research.

References

1. CISCO, Cisco Global Cloud Index: Forecast and Methodology, 2016–2021. Accessed: Dec. 27, 2019. [Online]. Available: https:
2. //www.cisco.com/c/en/us/solutions/collateral/service-provider/global- cloud-index-gci/white-paper-c11-738085.html
3. A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with Big Data: Challenges and approaches," IEEE Access, vol. 5, pp. 777–797, 2017.
4. H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: Perspectives and challenges," IEEE Int. Things J., vol. 4, no. 1, pp. 75–87, Feb. 2017.
5. M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," IEEE J. Sel. Areas Commun., vol. 36, no. 3, pp. 587–597, Mar. 2018.
6. R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," IEEE Commun. Mag., vol. 78, no. 2, pp. 680–698, Jan. 2018.
7. H. El-Sayed et al., "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," IEEE Access, vol. 6, pp. 1706–1717, 2018.
8. A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, R. M. Parizi, and K. R. Choo, "Fog data analytics: A taxonomy and process model," J. Netw. Comput. Appl., vol. 128, pp. 90–104, 2019.
9. B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in Proc Conf. Comput. Vision Pattern. Recognit., 2018, pp. 8697–8710.
10. H. F. Nweke, Y. W. Teh, M. A. A.-G., and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," Expert Syst. Appl., vol. 105, pp. 233–261, 2018.
11. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Int. Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.
12. A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in Proc. 10th Int. Conf. Intell. Syst. Control, 2016, pp. 1–8.
13. N. Abbas, A. Zhang, Y. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Int. Things J., vol. 5, no. 1, pp. 450–465, Feb. 2018.
14. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

ISSN: 2173-0695

15. G. G. Jia, G. G. Han, A. Li, and J. Du, "SSL: Smart street lamp based on fog computing for smarter cities," IEEE Trans. Ind. Informat., vol. 14, no. 11, pp. 4995–5004, Nov. 2018.
16. P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," IEEE Trans. Ind. Informat., vol. 13, no. 4, pp. 1910–1920, Aug. 2017.
17. L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," IEEE Trans. Ind. Informat., vol. 14, no. 10, pp. 4665–4673, Oct. 2018.
18. Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," IEEE Trans. Ind. Informat., vol. 15, no. 2,
19. pp. 976–986, Feb. 2019.
20. S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol. 5, pp. 6757–6779, 2017.
21. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surv. Tut., vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.
22. N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mah- moud, "Smartcityware: A service-oriented middleware for cloud and fog enabled smart city services," IEEE Access, vol. 5, pp. 17 576–17 588,
23. 2017.
24. B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," IEEE Trans. Ind. Informat., vol. 13, no. 5, pp. 2140–2150, Oct. 2017.
25. X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog- enabled real-time traffic management system," IEEE Trans. Ind. Informat., vol. 14, no. 10, pp. 4568–4578, Oct. 2018.
26. J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multi-tier fog computing with large-scale IoT data analytics for smart cities," IEEE Internet Things J., vol. 5, no. 5, pp. 677–686, Apr. 2018.
27. A. M. Rahmani et al., "Exploiting smart e-health gateways at the edge of healthcare Internet-of-things: A fog computing approach," Future Gener. Comput. Syst., vol. 78, no. 2, pp. 641–658, 2018.
28. P. Ritrovato, F. Xhafa, and A. Giordano, "Edge and cluster computing as enabling infrastructure for internet of medical things," in Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl., 2018, pp. 717–723.
29. J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor- based activity recognition: A survey," Pattern Recognit. Lett., vol. 119,
30. pp. 3–11, 2019.
31. E. Zdravevski et al., "Improving activity recognition accuracy in ambient- assisted living systems by automated feature engineering," IEEE Access, vol. 5, pp. 5262–5280, 2017.
32. A. Ferrari, D. Micucci, M. Mobilio, and P. Napoletano, "On the person- alization of classification models for human activity recognition," IEEE Access, vol. 8, pp. 32 066–32 079, 2020.
33. J.-H. Li, L. Tian, H. Wang, Y. An, K. Wang, and L. Yu, "Segmentation and recognition of basic and transitional activities for continuous physical human activity," IEEE Access, vol. 7, pp. 42 565–42 576, 2019.
34. M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating energy data for machine learning with recurrent generative adversarial networks," Energies, vol. 13, no. 1, 2020, Art. no. 130.
35. N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," Neural Comput., vol. 9, no. 7, pp. 1493–1516, 1997.
36. C. Banos et al., "Design, implementation and validation of a novel open framework for agile development of mobile health applications," Biomed. Eng. Online, vol. 14, no. 2, 2015, Art. no. S6.

ISSN: 2173-0695