Edge-Cloud Computing for Internet of Things Data Analytics: Embedding Intelligence in the Edge With Deep Learning

Ananda Mohon Ghosh and Katarina Grolinger, *Member, IEEE*

***Abstract*—Rapid growth in numbers of connected devices including sensors, mobile, wearable, and other Internet of Things (IoT) devices, is creating an explosion of data that are moving across the network. To carry out machine learning (ML), IoT data are typically transferred to the cloud or another centralized system for storage and processing; however, this causes latencies and increases network traffic. Edge computing has the potential to remedy those issues by moving computation closer to the network edge and data sources. On the other hand, edge computing is limited in terms of computational power, and thus, is not well-suited for ML tasks. Consequently, this article aims to combine edge and cloud computing for IoT data analytics by taking advantage of edge nodes to reduce data transfer. In order to process data close to the source, sensors are grouped according to locations, and feature learning is performed on the close by edge node. For comparison reasons, similarity-based processing is also considered. Feature learning is carried out with deep learning—the encoder part of the trained autoencoder is placed on the edge and the decoder part is placed on the cloud. The evaluation was performed on the task of human activity recognition from sensor data. The results show that when sliding windows are used in the preparation step, data can be reduced on the edge up to 80% without significant loss in accuracy.**

***Index Terms*—Autoencoders (AEs), data reduction, deep learning (DL), edge computing (EC), human activity recognition (HAR), Internet of Things (IoT).**

I. Introduction

CISCO estimates that the number of connected devices will exceed 28 billion by the year 2022, up from 18 billion in 2017 [1]. More than half of those devices, over 14.6 billion, will be machine-to-machine connections. The number of connected devices, together with a flood of data they generate, will increase network traffic. According to Cisco, by the year 2022, global annual internet traffic will reach 4.8 zettabytes, up from 1.5 zettabytes in 2017 [1]. Although this will come with positive impacts including new applications/services and increased use of the existing ones, it will also increase demand on network bandwidth and put pressure on the already strained communication infrastructure.

The Internet of Things (IoT) provides a platform for devices to connect to the Internet and other devices and enables devices to collect data about their environment. IoT supports smart systems such as smart cities, smart health care, smart transportation, and smart energy; however, the realization of these smart systems relies on the ability to analyze these data [2]. On the other hand, most IoT edge devices, such as sensors, do not have computation abilities to perform complex data analytics computations and, therefore, have been primarily responsible for monitoring the environment and transmitting data to a more powerful system, often the cloud or a centralized system, for storage and processing [3].

Consequently, a typical IoT data analytics involve sending data to the cloud, analyzing it on the cloud, and subsequently delivering results to another device. For example, process monitoring data from a smart factory may be transferred to a data center thousands of miles away where they are stored and processed; then, the results are sent back to the same factory for process optimization. This workflow increases not only network traffic, but also data transfer latencies. However, data analytics computation cannot be performed solely on the connected devices as they have limited computation resources.

Combining edge with cloud computing has the potential to reduce IoT network traffic and associated latencies while still supporting complex data analytics tasks. Edge computing (EC) pushes the computation away from the cloud or a centralized system and to the edges of the network and sources of data, and thus, reduces network traffic and latencies. Although EC has been recognized as a powerful approach for tasks such as mobile task offloading [4] and content delivery [5], its use for data analytic has remained limited [6], [7].

In recent years, deep learning (DL) has demonstrated success in a variety of domains including image classification [8] and human activity recognition (HAR) [9]. In the IoT context, DL ability to carry out representation learning and to transform data into hierarchical abstract representations is beneficial for IoT data analytics because it can enable learning the good features. A type of DL, a deep autoencoder (AE) is a NNs trained to learn data encoding in an unsupervised manner. Together with

Manuscript received May 27, 2020; accepted July 5, 2020. Date of publication July 13, 2020; date of current version November 20, 2020. This work was supported by Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-2018-06222. Paper no. TII-20-2637. *(Corresponding author: Katarina Grolinger.)*

The authors are with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON N6A 3K7, Canada (e-mail: aghosh@uwo.ca; kgroling@uwo.ca).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.3008711

encodings, the reconstruction is learned enabling AE to restore its inputs from the reduced encodings, possibly with some loss of information.

This article investigates combining edge and cloud computing with IoT data analytics. The main contributions are the reduction of network traffic and latencies for machine learning (ML) tasks by employing the edge nodes and the evaluation of the degree of data reduction that can be achieved on the edge without a significant impact on the ML task accuracy. Edge nodes act as intermediaries between IoT devices and the cloud reducing the quantity of data sent to the cloud. The encoder part of the trained AE is employed on the edge to create data encodings that are sent to the cloud. ML task on the cloud is carried out in two ways—directly with encoded data, or the original data are first restored with the decoder part of the AE and then used for the ML task.

As IoT data can originate from different sensors and locations, this study explores feature learning from all data fused together, from data grouped by their source locations, and from data grouped according to sensor similarities. The evaluation scenario involves HAR from sensors including accelerometers and gyroscopes mounted on different parts of the human body. Results show that the presented approach can reduce transferred data up to 80% without a considerable impact on HAR accuracy.

The remainder of the article is organized as follows. Section II provides the background, Section III discusses related work, Sections IV and V present edge-cloud ML model and evaluation methodology, respectively. Section VI discusses the results. Finally, Section VII concludes this article.

II. Background

This section introduces EC and discusses DL-based dimensionality reduction.

A. Edge Computing

Centralized infrastructure systems, such as the cloud, store data, execute business logic, and perform data analytics tasks far away from the end users and data sources. They offer great advantages of immense computation capability, high scalability and reliability, pay-as-you-go billing model, and low initial cost. However, with zettabyte-sized traffic and the explosion of connected devices, transferring all data to the cloud for processing is not practical or even feasible. EC has emerged as a way of dealing with these challenges by pushing computation to the edges of the network and sources of data [6].

Fog computing shares many characteristics with EC. Occasionally, the terms "fog" and "edge" are used interchangeably [10]; but EC focuses more on nodes closer to IoT devices, whereas fog can include any resource located anywhere between the end device and the cloud. In this work, we use the term "edge" to refer to the end devices themselves, such as sensor nodes and smartphones, as well as computation nodes located close to the network edge such as edge servers.

The key idea behind the EC is to reduce the network traffic by bringing computation closer to the data sources. It has been investigated extensively in mobile computing: mobile EC offloads computation and data storage to the edge (e.g., base stations) to reduce network latencies, improve user experience, reduce battery consumption, and introduce location-awareness [11]. EC is especially suitable for applications with ultralow latency requirements and for content delivery and caching [12].

Even though EC provides advantages of reduced traffic, computing resources available on the edge are not comparable to those present in the cloud. Thus, computationally intensive tasks, such as ML, are not well-suited for edge devices. Nevertheless, the edge can supplement the cloud computing and perform part of the computation, consequently reducing network traffic and latencies.

B. Deep Learning

DL is a class of ML approaches in which the models are composed of multiple computational layers responsible for learning data representations with different levels of abstraction [13]. Due to its representation capabilities, ability to learn complex models, and diversity of architectures [2], DL has been successful in many domains including various vision tasks, natural language processing, and speech recognition.

AEs are a subcategory of DL approaches used for learning data representations (encodings) in an unsupervised way. Essentially, an AE is a neural network (NN) that learns to reconstruct its inputs; bottleneck NN layers prevent it from merely copying the input to output and force it to learn data representations. An AE consists of encoder and decoder, each one possibly composed of several stacked layers. The encoder part of the network is responsible for reducing dimensionality (encoding); therefore, the number of neurons typically reduces starting from the input layer to the last encoder layer. In contract, the decoder part is responsible for reconstructing the input signal from encoded values, and thus, typically consists of layers with a gradually increasing number of neurons. AE can be used for noise removal and anomaly detection, but they often serve as a preprocessing step for another ML task [2]. Once an AE is trained, the encoder network can be used for dimensionality reduction by taking encoder outputs (encodings) as inputs to another ML model.

In this article, the encoder part of the trained AE is deployed on the edge to reduce dimensionality before the data are sent to the cloud. Moreover, the decoder is employed on the cloud to reconstruct the original signal.

III. Related Work

EC has been gaining popularity, especially with applications that require fast response time and those with limited bandwidth because it locates computation close to data sources. Applications of EC including smart street lamps [14], face identification [15], smart manufacturing [16], and vehicular networks [17] have demonstrated great success and prompted further investigations.

Wang *et al.* [18] presented a survey on mobile edge networks focusing on computing-related issues, edge offloading, and communication techniques for edge-based computing. The use cases highlighted in their study include IoT, connected

vehicles, content delivery, and big data analysis. At the same time, Wang *et al.* [18] identified real-time analytics as one of the open challenges. Similarly, Abbas *et al.* [12] surveyed mobile EC and also identified big data analytics as a future research direction. While Wang *et al.* [18] and Abbas *et al.* [12] examined mobile EC, El-Sayed *et al.* [6] focused on IoT applications of EC. They compared the characteristics of cloud, multicloud, fog, and EC, and identified low bandwidth utilization and latencies as the main EC advantages. Mao *et al.* [19] see EC as a key enabling technology for realizing the IoT vision and, similar to Wang *et al.* [18] and Abbas *et al.* [12], recognize data analytics as one of the future research directions in EC.

The discussed surveys [6], [12], [18], [19] note the potential of EC in data analytics and point out the importance of EC in IoT for handling the rapid increase of the number of connected devices. Our study contributes to employing EC for data analytics by combining edge and cloud computing for the delivery of ML applications.

Smart cities are one of the commonly discussed use cases and applications of EC. Mohammad *et al.* [20] examined possibilities of service-oriented middleware for cloud and fog enabled smart city services. They did not discuss specific smart city services but focused on the middleware. Their experiments demonstrated the benefits of EC in terms of response time. Tang *et al.* [21] presented a hierarchical fog computing architecture for the support of connected devices in smart cities. In addition to the hierarchy of fog nodes, the proposed model includes the cloud as the top layer. The evaluation was performed on an event detection task in a smart pipeline monitoring system: the preliminary results demonstrated the feasibility of the proposed architecture. Similar to Mohammad *et al.* [20] and Tang *et al.* [21], our study also employs both edge/fog and cloud, but differs from theirs in that it also includes an extensive evaluation of the presented edge-cloud architecture.

Another example of the cloud-fog approach is the work of Wang *et al.* [22] on combining fog and cloud computing for real-time traffic management. In their system, vehicles act as edge nodes, and roadside units take the role of cloudlets and communicate between vehicles and the cloud. Their study [22] focuses on message passing and processing while our work deals with ML for IoT.

He *et al.* [23] proposed a multitier fog computing model for large-scale IoT data analytics in Smart Cities. They evaluated the proposed model on the classification tasks—the results demonstrated that fogs can improve the performance of smart city services. While the work of He *et al.* [23] deals with fog architecture, our study takes advantage of both edge and cloud for the ML task.

There are also applications of fog computing in health care. Rahmani *et al.* [24] presented a fog-assisted architecture for smart e-Health which embeds intelligence between sensors and the cloud. In their study, fog nodes are quite powerful and thus able to handle data filtering, compression, fusion, and analysis with only minimal data sent to the cloud. Our study, on the other hand, still performs a large part of the computation on the cloud. Ritrovato *et al.* [25] also dealt with health care and proposed an EC anomaly detection system for streaming data. While they are concerned with stream processing algorithms, our study deals with ML algorithms.

The main difference between our work and the reviewed studies is that we combine the edge and cloud for ML tasks. Several studies also employed edge-cloud architectures [20], [22], [24], but for non-ML tasks. For ML tasks, the processing on the edge must accommodate final ML computation on the cloud while still reducing network traffic. The most similar work to ours is the work of Tang *et al.* [21] as they also consider ML and feature extraction on the edge; however, they only extract the signal's mean and variance, which limits application scenarios, while we use a generic approach based on AEs. Moreover, we also evaluate the degree of feature reduction that can be carried out without significantly impacting ML accuracy.

As our study evaluates the presented approach on sensor-based (HAR, it is important to mention a few works from this category. Given the fact that DL has been quite successful and extensively used for HAR [26], the work of Wang *et al.* [26] surveyed DL approaches for activity recognition; they highlighted the importance of model selection and the significance of preprocessing including the sliding window technique. Zdravevski *et al.* [27] specifically focused on feature engineering for HAR: they first extracted a large number of features (3232 for MHEALTH dataset), and then reduced them by combining different feature reduction techniques. Ferrari *et al.* [28] investigated personalization of HAR models and also applied sliding windows and feature generation. Li *et al.* [29] are concerned with recognizing transitions between activities; they first extracted 118 features and created fixed size segments. Next, each segment was analyzed to determine if there was a change of activity within the segment.

These HAR studies do not employ EC, nor are they concerned with network traffic. As the sliding window technique improves accuracy, it is commonly used in HAR studies [26]–[28]; consequently, we also use it. While others are interested in the sliding window technique effect on accuracy, we investigate its impact on data reduction on the edge. AE and principal component analysis (PCA) have been used to improve HAR accuracy [9], but our study uses those techniques to reduce network traffic.

IV. Edge-Cloud ML System Model

The overall architecture of the edge-cloud ML model is depicted in Fig. 1 and details of each of the three main components, preprocessing, data reduction, and Cloud ML, are described in the following sections.

A. Preprocessing

The edge-cloud ML process starts with data from IoT sensors being passed to the edge for preprocessing in contrast to traditional ML where the data are sent directly to the cloud. The preprocessing always includes normalization while the sliding window technique is optional, and its impact is evaluated in the experiments.

1) Normalization: To avoid dominance of features with large values and to improve training convergence, the data are normalized using standardization (z-score). In place of standardization,

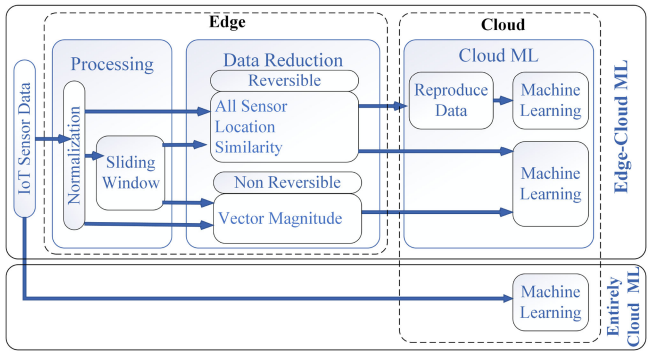

Fig. 1. Edge-cloud ML system model architecture.

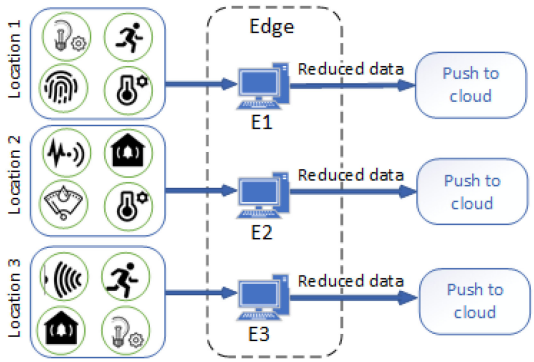

Fig. 2. Location-based data reduction.

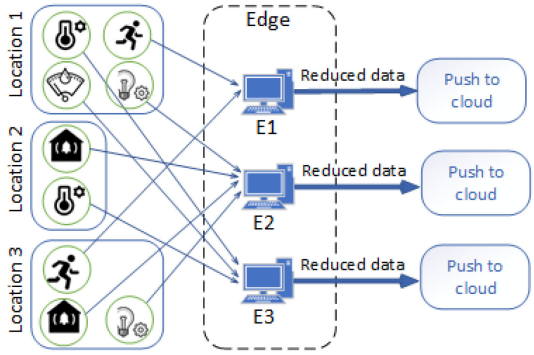

Fig. 3. Similarity-based data reduction.

min–max scaling could be used, but standardization was selected because of its ability to handle outliers. Each feature is rescaled to have zero mean and unit variance as given by

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the original feature value, μ and σ are that feature mean and standard deviation, and $\hat{x}$ is the normalized value.

2) Sliding Window: At this point, one data sample consists of several readings, potentially from different sensors and different locations, for the same time step t. For the time series data, the windows sliding technique is applied to help the model capture time-dependencies or to prepare data in the format needed by the ML algorithms [30]. In HAR, the sliding window has achieved great success [30]. This study investigates if features can be reduced after the sliding window technique is applied and examines the effect on the degree of data reduction. With the sliding window of length l, the first sample consists of all readings for the first l time steps; with the f number of features, this results in the $l \times f$ matrix for each sample. Next, the window slides for k steps and the second sample consists of the readings from the time step k to $k + l$. The window keeps sliding to create the remaining samples. In this work, the sliding step $k = 1$ is used as this results in a higher number of samples for training and allows the input to capture shifts in temporal patterns. Once the system is trained, different sliding steps can be applied depending on the specifics of the use case and data transfer constraints.

B. Data Reduction

Data reduction happens on the edge in order to reduce the quantity of data sent to the cloud. Feature reduction challenges include selecting the technique and the number of features. In the centralized systems, these decisions are primarily driven by the ML accuracy while in the edge-cloud environment, network traffic also needs to be considered. Even though AEs are well-suited for feature learning on the edge; it remains a challenge to determine the maximum feature reduction and the corresponding traffic reduction while maintaining the accuracy of the ML tasks.

Data reduction can be carried out directly with normalized data or with samples created by the sliding window technique. Regardless of whether the sliding window technique is used or not, the data reduction approach is the same. Two categories of approaches are considered—reversible and nonreversible.

1) Reversible: Reversible approaches are the approaches that reduce data with an ability to reproduce the original data from the reduced representations. With these approaches data reduction executes on the edge, reduced data are sent over the network, and on the cloud, ML can be performed directly on the reduced data, or the original data can be reproduced first. Here, we focus on AEs as, once the AE is trained, the encoder can reduce data on the edge and decoder can restore the original data on the cloud. AE performance is compared to data reduction with PCA [31]. When PCA is applied to reduce dimensions, original data can be reconstructed using the eigenvectors. For both, AE and PCA, the accuracy of reconstructed data depends on the degree of dimensionality reduction.

As illustrated in Fig. 1, for both reversible techniques, AE and PCA, three different scenarios are considered—all sensors, location-based, and similarity-based scenarios. In the all sensors approach, all the data are considered together, and data reduction is performed on the merged data from all sensors.

The location-based scenario considers data reduction based on a group of colocated sensors, as illustrated in Fig. 2. For example, in location 1, there are four different sensors, and their data are sent to the edge node E1 because of its close proximity to those sensors. Similarly, the location 2 sensors send their data to E2, and so on. The idea is to keep the edge part of the processing as close as possible to the sources of data and reduce the distance data needs to travel before reduction. The data that arrive at one node are reduced together on that node; consequently, there is one AE for each of the edge nodes.

The similarity-based scenario illustrated in Fig. 3 groups sensors based on their similarity. For example, all gyroscopes could represent one group and all accelerometers another one. This will result in more homogeneous data groups, but it can also increase the distance of sensors from the edge nodes. As with the location-based scenario, there is one AE for each of the edge nodes.

2) Nonreversible: Nonreversible approaches include those without the way of reproducing the original data after the data have been reduced. Here, we consider the vector magnitude which is suitable for sensors that measure values in multidimensional space such as accelerometers and gyroscopes. The dimensionality is reduced as follows:

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

Here x, y, z are the measurements in Euler coordinates and d is the vector magnitude. Consequently, vector magnitude reduces dimensions in 3:1 ratio.

C. Cloud ML

The reduced data are sent from the edge to the cloud for further ML processing. As illustrated in Fig. 1, there are two possible ways to carry out the ML task. The first option is to reproduce original data and use these reproduced data for the ML task. This is possible only if the reversible technique was used for data reduction. The second option is to carry out the ML task directly on the reduced data, which works for both reversible and nonreversible reduction techniques. As the reproduced data have a larger number of features, training ML model for such data is more computationally expensive than training ML model for the reduced number of features.

V. Evaluation Methodology

This section introduces the dataset and describes the evaluation methodology.

A. Dataset and Preprocessing

The presented approach was evaluated on the HAR task with the (Mobile Health) MHEALTH Mobile Health dataset [32]. Activity types such as walking, running, or sitting, are determined with the help of sensors. The MHEALTH dataset contains recordings of body motions for ten individuals while performing different activities. The recordings were acquired by three types of sensors (accelerometer, gyroscope, and magnetometer) with each one obtaining three readings corresponding to three axes. All three sensors are placed on the left ankle and the right wrist while on the chest, there is only the accelerometer. This makes a total of 21 features: 7 sensors × 3 axis. The labels are 12 physical activities (standing, walking, etc.) and the sampling rate is 50 Hz.

Data from all individuals are merged together and used as such throughout experiments. For the training and the test, data were split 80:20, respectively. As illustrated in Fig. 1, after data are normalized, the preprocessing can continue with or without applying the sliding window. To evaluate the impact of the window size on activity detection accuracy and on obtainable reduction rate, three window sizes are considered: 25, 50, and 100 time steps. With 50 Hz sampling rate, these sizes correspond to 0.5, 1, and 2 s.

TABLE I
Number of Features Before and After Reduction for Reversible Approaches (Window Size 100)

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

B. Data Reduction

The two categories of data reduction on the edge are considered—reversible and nonreversible.

1) Reversible: Reversible data reduction with and without sliding window is applied with three scenarios:—all sensors, location-based, and similarity-based. Table I shows the number of features before and after data reduction for reversible technique with window size 100 for the three scenarios. In the table and remainder of the article, we use term *Direct* to refer to the approach without the sliding window.

The table gives the number of features before and after reduction for a 66% and 70% reduction, but experiments were also carried out for a 80%, 90%, and 95% reduction. The starting reduction of 66% was selected to match the maximal reduction of the vector magnitude approach 3:1. For all sensors, there is no edge location, as everything happens on a single node. For location-based and similarity-based approaches, edge location identifies a node where data are aggregated. For the location-based approach, there are three nodes corresponding to sensors located on the arm, leg, and chest. For the similarity-based approach, the nodes correspond to the type of sensor—accelerometer, gyroscope, and magnetometer. For example, L2 in location-based techniques aggregates readings from three sensors; therefore, the number of original features for the direct approach is 9 (3 sensors with 3 axes each), as shown in Table I. For scenarios with windows, the number of features is the window length × the number of features; for example, for location-based approach and L2, the number of features is 9 × 100.

It is important to note that the direct option is only used with 66% data reduction rate. At that value, the number of features is already very low, for example, 7 for all sensors, and further

reduction is considered undesirable. The number of features in the approaches with the sliding window is much higher as buffering happens on the edge and only the compressed data are sent to the cloud. The approaches with the sliding window require buffering on the edge, which results in a much higher number of features and creates a possibility for greater reduction rates. Moreover, it is expected that the sliding window technique will achieve better accuracy because of its past successes in HAR [26].

All of the data reduction variants considered in Table I are carried out with AE and with PCA. Regardless if PCA or AE are used, data reduction rates remain the same as presented in Table I. For AE, a reduction degree is controlled by setting the number of neurons in the bottleneck layer where for PCA it is controlled by the number of selected principal components. For the location-based and similarity-based approaches, there is one AE for each edge node responsible for reducing data that are arriving on that node. Similarly, PCA works independently on each node.

Overall, there are 90 data reduction experiments with sliding windows: 3 scenarios $\times$ 3 window sizes $\times$ 5 reduction rates $\times$ 2 algorithms (AE, PCA). Additionally, for direct reduction, there are six experiments: 3 scenarios $\times$ 2 algorithms. This makes a total of 96 data reduction experiments.

2) Nonreversible: The only nonreversible approach considered is the vector magnitude. With three-axis data as those recorded with an accelerometer, gyroscope, and magnetometer, this results in 3:1 reduction. As with reversible approaches, the reduction is applied with and without the sliding window. Without the window, vector magnitude for all data reduces 21 features to seven. For sliding windows 25, 50, and 100, features are reduced from 525, 1050, and 2100 to 175, 350, and 700. This makes a total of 4 experiments: 3 for each window size + direct reduction (no sliding window).

C. Cloud ML

After data are reduced on the edge, they are sent to the cloud for the final processing. In the HAR task, this final step is classification: recognizing the type of activity. While on the edge there are ML models on each of the nodes taking care of processing data arriving at that node, on the cloud, there is a single model merging data from all edge nodes. Here, as illustrated in Fig. 1, we consider three approaches to carrying out this task:

1) ML With Reduced Data: Reduced data arriving from all edge nodes are used as such directly for classification. This approach is applicable for both, reversible and nonreversible reduction techniques. For the HAR task, a feed-forward NN (FFNN) was used for the final classification. The number of output nodes is 12 (12 activities) and the number of input nodes in the FFNN corresponds to the number of features after the data reduction and reduces with increased reduction rates. A different number of hidden layers and neurons was used depending on the number of input features—for the number of input features more than 350, FFNN had three hidden layers, and for 350 or less, it had two hidden layers. The number of nodes in the hidden layers gradually decreased, for example, 420-128-32-12 for 420 input features. Experiments were performed with different numbers of neurons in the hidden layer, but this strategy showed high accuracy.

2) ML With Reproduced Data: This approach consists of two steps, reproducing the original data from their reduced representation and then classifying the reproduced data. Clearly, this is only applicable when reversible techniques are used to reduce data on the edge. As the number of features here matches the number of features in the original dataset, this approach requires more complex models and is more computationally expensive than ML with reduced data. However, as the original data are reproduced, these reproduced data can be used for other tasks. As with ML with reduced data, classification is carried out with the FFNN and the same strategy is applied for choosing the number of layers and neurons in hidden layers. As the number of features is higher with reproduced than with reduced data, the same strategy will result in a higher number of layers and neurons.

3) Entirely Cloud-Based ML: This is not edge-cloud architecture, but a traditional cloud-based technique where everything is sent to the cloud for ML. It is considered in the evaluation solely for comparison reasons. Again, FFNN is used for classification, and the same strategy was applied for determining the number of layers as with edge-cloud approaches.

As discussed in Section V.B.1, evaluation considers 96 experiments that were performed for reversible techniques. With two approaches on the cloud (with reduced and with reproduced data) this makes a total of 192 experiments with reversible techniques. There are four experiments for nonreversible approaches (Section V.B.2) and four (one direct and 3 sliding windows) for entirely cloud-based ML. This makes for a total of 200 experiments.

VI. Results and Discussion

This section first presents data reduction and network traffic results and then discusses findings.

A. Data Reduction

For the evaluation, in addition to accuracy, precision and recall metrics are used because of their frequency in HAR studies [27]

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

where TP and TN are true positives and true negatives, and FP and FN are false positives and false negatives.

First, to enable comparisons, the accuracy of traditional ML without any data reduction is presented in Table II. Next, Table III shows classification accuracy for reversible techniques with 66% data reduction and sliding window size 100. It can be observed that similar to traditional ML (see Table II), the sliding window technique achieves better accuracy than direct approaches, regardless of the algorithm (AE or PAC) or scenario (all sensors, location-based, or similarity-based). However, there

TABLE II
CLASSIFICATION ACCURACY: ENTIRELY CLOUD ML

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

TABLE III
CLASSIFICATION ACCURACY: REVERSIBLE APPROACHES, SLIDING WINDOW 100, 66% DATA REDUCTION

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%

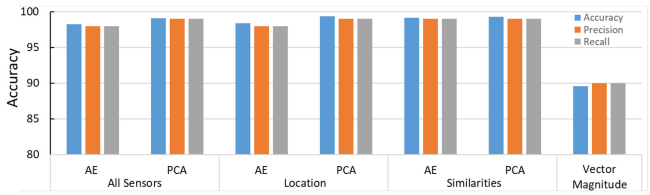

Fig. 4. Classification accuracy: direct (no Window), 66% reduction, reversible and nonreversible approaches.

is no big difference between AE and PCA or between reduced and reproduced approaches. Although data reduction was 66%, the accuracy is very close to the accuracy of the traditional cloud-based ML. Precision and recall were also calculated; they showed similar patterns to the accuracy observed in Table III.

Like Table III, Fig. 4 is concerned with the 66% reduction, but it includes both, reversible and nonreversible approaches, in contrast to Table III which considers only reversible approaches. All reversible approaches achieve much higher accuracy than the nonreversible, vector magnitude approach.

Table III and Fig. 4 analyze classification accuracy for 66% data reduction and Table III considers solely sliding window 100. To analyze the impact of data reduction rates on the accuracy, Fig. 5 illustrates accuracy changes with respect to reduction rate for each of the three scenarios—all sensors, location-based, and similarity-based. It can be observed that accuracy slowly decreases as reduction increases from 66% to 90%, and then experiences sharper drop with a 95% reduction rate. Overall, AE shows slightly higher accuracy than PCA, and classification with reproduced data is slightly better with reproduced than with reduced data. When data are reproduced, information embedded in the encoder part of AE or in the PCA eigenvectors is used to expand the encoded values, thus slightly increasing the HAR accuracy for reproduced data. Nevertheless, even with a 95% reduction rate, the accuracy drops less than 0.25% in comparison to traditional ML.

Fig. 6 shows the same evaluation as Fig. 5, but for window size 50. While for window size 100, the accuracy remained relatively stable for reductions 66% to 80%, with sliding window 50, the accuracy drops as reduction rate changes. Nevertheless, the accuracy for the 95% reduction is similar for windows 100 and 50; however, window 50 is more desirable as it shortens FFNN training time and reduces required data accumulation on the edge. As for window size 100, for window size 50, AE with reproduced data is slightly better than the other approaches.

Next, Fig. 7 shows accuracy for window size 25. Whereas with windows sizes 100 and 50, classification with reduced and reproduced data achieved similar accuracy, with window size 25, approaches with reduced data show lower accuracy than with reproduced data for all reduction rates and both algorithms, PCA and AE. As with other window sizes, AE with reproduced data achieved the highest accuracy for almost all reduction rates. Comparing window 25 and 100 results for AE with reproduced data, window 100 shows slightly better accuracy, but window 25 may be more desirable because of its shorter training time.

Figs. 5–7 compare the accuracy of reversible approaches for different window sizes, but it is also important to compare sliding window approaches with direct one: Fig. 8 shows this comparison. As direct approaches are only considered for a 66% reduction rate, this figure compares them to sliding window approaches with the same reduction rate. All the sliding window approaches, regardless of the size of the window, outperform direct approaches further confirming the benefit of using sliding windows. As with Figs. 5–8 shows that windows 100 and 50 achieve similar accuracy while for window size 25, accuracy for reduced data decreases.

Finally, the accuracy of nonreversible approaches for different window sizes is depicted in Fig. 9. As with reversible approaches, window sizes 100 and 50 achieve similar results while accuracy for window size 25 is lower. Comparing the accuracy of the non-reversible approach with windows (see Fig. 9) and the same approach without windows (direct) (see Fig. 4), the accuracy is increased by almost 10% when sliding windows were added. With sliding windows, the nonreversible vector magnitude approach reaches accuracy close to reversible techniques; nevertheless, vector magnitude is computationally much more simple than the other approaches.

B. Network Traffic

So far, the analysis has considered data reduction in terms of number of features. However, when the system is deployed, the actual traffic will not exactly follow the feature reduction rates due to network overheads. To examine network traffic, the describe edge-cloud system has been simulated. The direct approach only needs a single edge node while location and similarity-based scenarios require three nodes, one for each location/similarity group, as discussed in Section IV-B. Each edge node is simulated by one virtual machine with 2 GB of memory and single core CPU. The cloud was emulated by a window server with 6 GB of memory and CPU with octa-core processor. The communication between the edge nodes and the server was established with the TCP-IP protocol and consequently, the

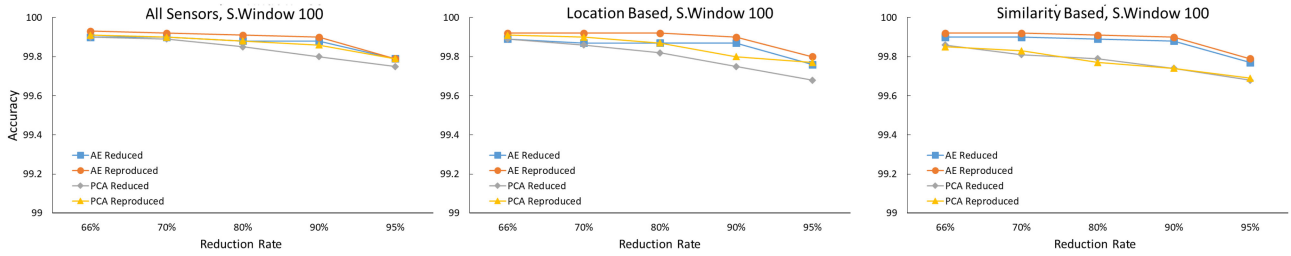

Fig. 5. Classification accuracy for different reduction rates: window size 100.

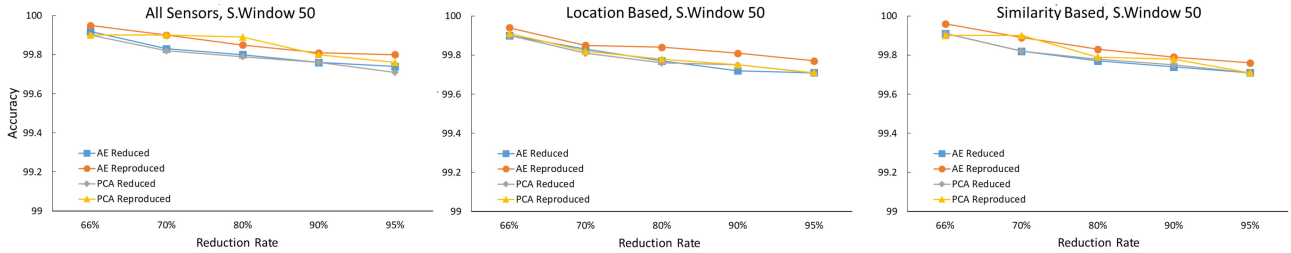

Fig. 6. Classification accuracy for different reduction rates: window size 20.

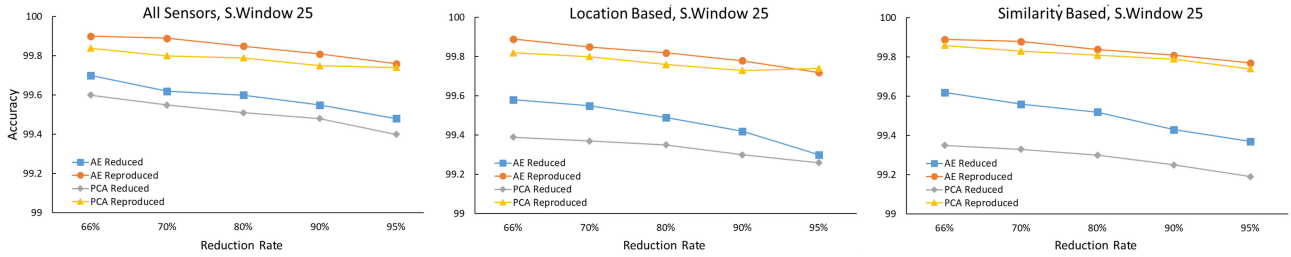

Fig. 7. Classification accuracy for different reduction rates: window size 25.

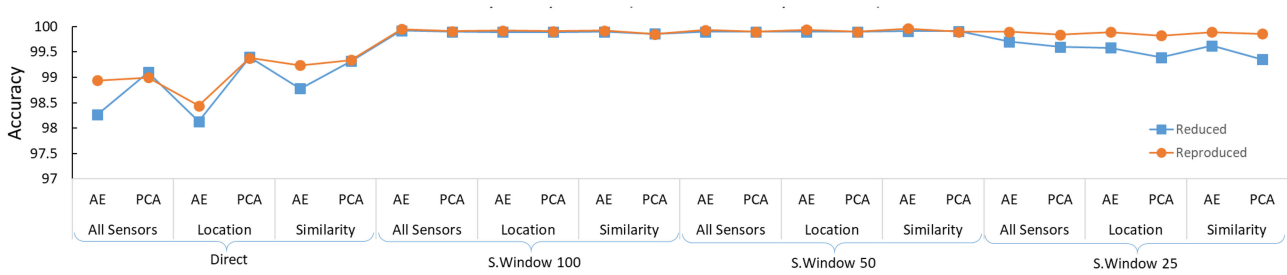

Fig. 8. Classification accuracy: direct and sliding window approaches, 66% reduction rate.

consumed bandwidth was measured over sockets. Multithreaded socket server ensures that multiple edge nodes can communicate with the server concurrently. Encoder parts of the trained AEs are located on the edge nodes and decoder parts as well as the classification models are deployed on the server. The change in the computation capacity of the edge nodes would not change the network traffic, but it must be sufficient to carry out data encoding. The minimum computing resources required on the edge depend on the size of the encoder network which is affected by the sliding window size.

Table IV compares the network traffic without feature reduction with the traffic in presence of 66% feature reduction for each of the scenarios. It also includes traffic reduction percentages indicating how much traffic was reduced in respect to the same scenario without reduction. For example, traffic reduction 49.71% for sliding window 100, for all sensors, indicates change from 7129 MB for no reduction to 3585 MB for all sensors, for sliding window 100. It can be observed that for each scenario, network traffic remains similar for all three scenarios, all sensors, location-based, and similarity-based. However, for both, reduced and nonreduced data, network traffic is much higher for scenarios with sliding windows than for direct approaches. The reason for this is that the sliding step size $k = 1$ results in readings belonging to different sliding windows and thus the

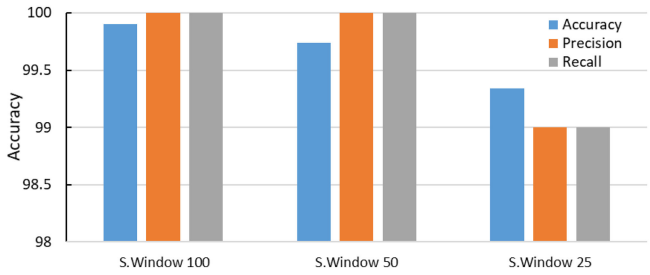

Fig. 9. Classification accuracy: vector magnitude with sliding windows.

TABLE IV
NETWORK TRAFFIC: NO REDUCTION AND 66% DATA REDUCTION

Scenarios	No Reduc.	Reduced: All Sensors	Reduced: Location Based	Reduced: 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

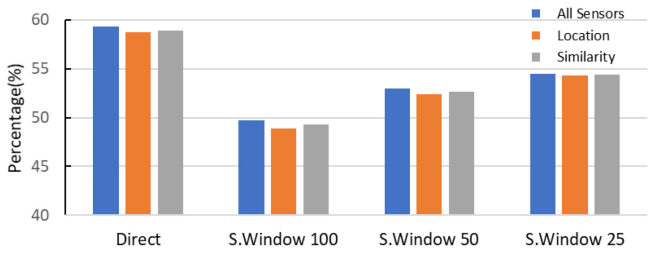

Fig. 10. Network traffic reduction.

same readings are included in the transfer multiple times. This can be avoided by using the siding step equal to the length of the sliding window if the application allows for window length delays.

Fig. 10 compares network traffic reduction for different data reduction approaches and scenarios, for 66% feature reduction. Similar conclusions can be drawn as with Table IV: scenarios (all sensors, location, and similarity) do not have major impact on traffic reduction. While feature reduction was 66% for all approaches and scenarios, network traffic reduction varied among approaches: the direct approach resulted in the highest reduction (around 59%) and, as the window size increased, the reduction rate decreased. These variations are caused by changes in network overheads as the quantity of data sent altogether changes. Although window size 25 has an advantage of reduced network traffic in comparison to window sizes 50 and 100, its accuracy is lower, indicating the need to compromise between the two.

C. Discussion

Experiment results demonstrate that the presented approach is capable of reducing data sent to the cloud up to 80% without significant loss in accuracy. The sliding window technique increases accuracy even when reduction is carried out on data preprocessed with sliding windows. Window sizes 50 and 100 achieve similar accuracy while window size 25 results in reduced accuracy for all scenarios (see Fig. 8): all sensors, location-based, and similarity-based. With sufficiently large sliding windows, even a 90% reduction results only in a small loss of accuracy: Fig. 5 shows relatively stable accuracy up to a 90% reduction.

However, the sliding window scenarios have a disadvantage of data aggregation on the edge. With the sliding step one, if reduced data is sent to the cloud on every time step, network traffic will still be high as sensor readings will belong to different windows. The sliding step equal to (or larger than) the window size prevents the readings from belonging to different windows and reduces network traffic, but the application scenario must allow for a window length delay.

Classification with reproduced data was slightly better than with reduced data; the difference increased for smaller windows (see Fig. 8). For larger window sizes 50 and 100, AE and PCA performed similarly, but for window size 25 AE outperformed PCA for all reduction rates (Figs. 5–7). This may be caused by the AE's ability to capture complex relations through nonlinear transformations in contrast to PCA's linear transformations which were not as successful.

Location and similarity-based approaches achieved very similar results to all sensors (see Fig. 4), but the former have the advantage of enabling data reduction on different nodes. Moreover, the location-based approach allows for locating edge nodes closer to the sources of data, and thus, could be beneficial for geographically distributed scenarios.

The rate of network traffic reduction is lower than the data reduction rate as illustrated in Table IV and Fig. 10 due to network overheads. Moreover, network traffic analysis highlighted the need to avoid overlapping sliding windows as they result in high network traffic. However, overlapping windows can still be used for training the networks, AE and classification FFNN, but should be avoided in edge-cloud deployments.

The presented approach was evaluated on the HAR task with MHEALTH dataset, but it can be applied for other IoT tasks and data sets. High correlation among different sensor readings will result in high data reduction rates. Moreover, different applications and ML tasks will have different reduction rates depending on how valuable are various parts of the data for that specific application/task.

VII. CONCLUSION

Traditionally, ML with IoT data was done by transferring data to the cloud or another centralized system for storage and processing. With the explosion of connected devices, this would lead to increased latencies and it would put a strain on communication networks.

This article explored merging edge and cloud computing for ML with IoT data with the objective of reducing network traffic and latencies. Three scenarios were examined—all sensors together consider all the data at once, location-based scenario groups data according to the IoT device locations, and similarity-based scenario groups data according to the similarities of sensors. The evaluation were carried out on the HAR task considering two nonreversible approaches, AE and PCA, and one nonreversible approach, vector magnitude. The results showed that data and the corresponding network traffic could be reduced even up to about 80% without significant loss of accuracy if a large sliding window was used in the preprocessing. Location and similarity-based approaches achieved similar accuracy to all sensors approaches, but they could carry out reduction on different edge nodes.

Future work will investigate the presented edge-cloud approach for different applications and with different sensors.

References

[1] CISCO, *Cisco Global Cloud Index: Forecast and Methodology*, 2016–2021. Accessed: Dec. 27, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th Int. Conf. Intell. Syst. Control*, 2016, pp. 1–8.

[12] 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.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] 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.

[15] 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.

[16] 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.

[17] 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, pp. 976–986, Feb. 2019.

[18] 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.

[19] 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.

[20] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "Smartcityware: A service-oriented middleware for cloud and fog enabled smart city services," *IEEE Access*, vol. 5, pp. 17 576–17 588, 2017.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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, pp. 3–11, 2019.

[27] 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.

[28] A. Ferrari, D. Micucci, M. Mobilio, and P. Napoletano, "On the personalization of classification models for human activity recognition," *IEEE Access*, vol. 8, pp. 32 066–32 079, 2020.

[29] 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.

[30] 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.

[31] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Comput.*, vol. 9, no. 7, pp. 1493–1516, 1997.

[32] 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.

Ananda Mohon Ghosh received the B.Sc. degree in computer science and engineering from Khulna University, Khulna, Bangladesh, in 2015, and the M.E.Sc. degree in software engineering from Western University, London, ON, Canada, in 2020.

In industry, he has been involved in data science and business intelligence roles. His current research interests includes Internet of Things, algorithms, data analytics, natural language processing, and computer vision.

Katarina Grolinger (Member, IEEE) received the B.Sc. and M.Sc. degrees in mechanical engineering from the University of Zagreb, Zagreb, Croatia, in 1994 and 1997, respectively, and the M.Eng. and Ph.D. degrees in software engineering from Western University, London, ON, Canada in 2009 and 2013, respectively.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Western University. She has been involved area of in the software engineering in academia and industry for over 20 years. Her current research interests include machine learning, sensor data analytics, edge computing, data management, and IoT.