Journal of Information Systems Engineering and Management
2025, 10(35s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Optimizing the Computational Offloading of Deep Neural Networks for Human Activity Recognition

Mohammed Ali Ahmed[1] , Mohsen Nickray[2]
[1,2] Department of Computer and Information Technology, Faculty of Engineering, University of Qom

ARTICLE INFO

Received: 28 Dec 2024

Revised: 18 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

Research Aim: Study the possibility of optimizing the computational offloading of deep neural networks by reducing the volume of data sent to the cloud with a focus on the application of human activity recognition with deep learning.

Research method: In this research, three proposed methods of reducing the number of data samples, reducing the precision of data samples and compressing data samples are presented. In the first method, the data samples are deleted one in between or more before sending them. Data restoration in the cloud side is performed by interpolation estimates. In the precision reduction method, floating-point data samples are converted to integers with fewer precision before sending them. They are converted back on the cloud side by using the inverse conversion function. In the third method, the data is compressed with low overhead compression algorithms, either lossy or lossless, and is decompressed on the cloud side.

Findings: Among the two proposed methods of reducing the number of samples and reducing the precision of data samples, both methods only slightly reduce the accuracy of activity detection. The latter method is superior to the former method due to a more significant reduction in data volume. Although the lossy compression method shows better results than the lossless method, neither is as effective as the precision reduction method and the reduction in the number of data samples.

Conclusion: Practical results show that although the methods of reducing the number of samples and reducing their precision can decrease the volume of data sent without a significant effect on accuracy, the precision reduction method is superior due to greater data volume reduction. Furthermore, the delta compression method is not as effective as the other two methods.

Keywords: Computational Offloading, Deep Neural Networks, Human Activity Recognition

INTRODUCTION

Graves et al. explored the application of deep recurrent neural networks (RNNs) in speech recognition, demonstrating their ability to process time-series data efficiently, which is crucial for tasks like human activity recognition (HAR) where sequences of sensor data are involved [1]. In [2], Eshratifar and Pedram examined energy-efficient computation offloading for deep neural networks (DNNs) in mobile cloud environments, showing that selective offloading of certain DNN layers can significantly reduce energy consumption and improve performance, a critical consideration for real-time HAR systems. Dey et al. identified key challenges in offloading deep learning inference to edge devices, focusing on computational constraints and the need for efficient data transfer methods. They proposed a hybrid execution model, improving resource efficiency and performance in HAR applications [3]. In [4], Huang et al. introduced DeePar, a framework that optimizes deep learning computations by dynamically partitioning them across devices, edge, and cloud servers based on current network conditions, reducing latency in mobile applications. Kemp et al. proposed Cuckoo, a framework for computation offloading on smartphones, illustrating how mobile devices can benefit from offloading computation-intensive tasks like HAR to external servers, reducing device strain and improving performance [5]. Ran et al. demonstrated the potential of mobile devices to leverage cloud-based offloading for accelerating deep learning inference, showing significant performance gains, especially for computationally heavy applications such as HAR [6]. In [7], Fadishei explored energy-efficient human activity recognition on smartphones by using a test-cost sensitive algorithm, which reduces the volume of data processed and sent to the cloud, thus conserving energy while maintaining performance. Mahmoodi et al. studied joint scheduling and cloud offloading for mobile applications, focusing on optimizing latency and energy consumption, which is highly relevant for mobile HAR systems that require low-latency real-time inference [8]. Messaoudi et al. proposed an approach to mobile gaming computation offloading, which has parallels to HAR, as both involve transferring intensive computational tasks from mobile devices to external servers to improve performance and reduce energy use [9]. In [10], Zhang et al. examined energy-efficient offloading strategies for real-time video applications in mobile cloud environments, suggesting techniques that could similarly enhance HAR systems by reducing the energy burden of real-time data processing. Deyannis et al. explored the use of edge servers for GPU-assisted antivirus protection on Android devices, demonstrating the benefits of offloading computational tasks to edge servers, which can also be applied to HAR to handle the large computational demands of deep learning models [11]. Wang et al. proposed selective offloading for accelerating mobile web applications, which can be adapted to HAR by dynamically determining which parts of the computational workload should be processed locally versus offloaded to the cloud, based on network conditions and computational complexity [12]. In [13], Guo et al. reviewed FPGA-based neural network inference accelerators, showing how hardware accelerators can be integrated into mobile or edge devices to improve the efficiency of DNN computations, potentially benefiting HAR applications with hardware-optimized inference. Jeong et al. discussed computation offloading for machine learning web applications, focusing on the edge server environment, which provides an efficient way to offload complex tasks from mobile devices, an approach that could be adopted in HAR to improve performance while reducing device strain [14]. Eshratifar et al. developed JointDNN, an efficient training and inference engine for mobile cloud services, demonstrating how joint processing of data between mobile and cloud platforms can enhance the performance of HAR systems by optimizing both energy and computational efficiency [15]. In [16], Karki et al. introduced Tango, a deep neural network benchmark suite that evaluates various accelerators' performance, highlighting the importance of optimizing DNN computations in HAR through hardware acceleration and efficient task offloading. Shi et al. benchmarked state-of-the-art deep learning tools, providing insights into the efficiency of different platforms for deep learning tasks like HAR, where computational offloading is essential to manage the high resource demands of real-time inference [17]. In [18], Qin et al. proposed a novel combined prediction scheme using CNN and LSTM models for urban PM 2.5 concentration prediction, showcasing how combining different types of deep learning models can enhance prediction accuracy, a technique that could be applied to improve HAR systems' recognition capabilities. Durstewitz et al. explored the use of deep neural networks in psychiatry, which, like HAR, requires the processing of large amounts of data to identify patterns, showing the broader applicability of DNNs for complex data-driven tasks [19]. Shah et al. introduced a system for distributing deep neural networks in fog networks, focusing on minimizing system costs while offloading computations, a strategy that could improve HAR by ensuring that mobile devices handle fewer intensive tasks while maintaining system efficiency [20]. Nazemi et al. developed a hardware-friendly algorithm for scalable training and deployment of dimensionality reduction models on FPGAs, highlighting the potential for improving the efficiency of HAR systems through hardware acceleration and optimized model deployment [21]. In [22], Hirsa et al. applied supervised deep neural networks for pricing and calibration of options, demonstrating how DNNs can be used for complex decision-making tasks, similar to how they are applied in HAR to classify and predict human activities based on sensor data. Gordon et al. introduced COMET, a framework for migrating code execution to offload computation transparently, which is relevant to HAR as it enables mobile devices to handle only lightweight processing tasks while offloading intensive computations to external servers [23]. Li et al. discussed learning in IoT environments with edge computing, emphasizing the benefits of offloading deep learning tasks to edge devices to reduce network congestion and processing times, which can be highly beneficial in HAR systems that rely on continuous real-time data processing [24]. Teerapittayanon et al. proposed a distributed DNN framework that spans the cloud, edge, and end devices, optimizing the distribution of computations to ensure that mobile devices are not overloaded with intensive tasks, a strategy that would enhance HAR applications by distributing computational workloads effectively [25]. In [26], Eshratifar et al. introduced BottleNet, a DNN architecture designed for intelligent mobile cloud computing services, which compresses data before offloading to the cloud, reducing the volume of data transmitted and improving the energy efficiency of mobile HAR systems. Jeong et al. proposed IONN, a framework for incrementally offloading neural network computations from mobile devices to edge servers, highlighting an efficient way to handle the computational demands of DNN-based HAR systems while maintaining energy efficiency [27]. Rosloniec provided an overview of fundamental numerical methods for electrical engineering, offering insights into computational optimization techniques that could be applied to improve the efficiency of DNN computations in HAR systems [28]. McClarren discussed the application of computational methods in nuclear engineering, showing the relevance of these techniques for optimizing computational models, which can similarly be applied in HAR to manage the computational complexity of deep learning models [29]. In [30], TensorFlow, an open-source deep learning platform, provides tools for building and deploying DNN models, including applications for mobile and edge devices, which are critical for implementing HAR systems that require optimized offloading strategies. Weiss introduced the WISDM dataset for smartphone and smartwatch activity and biometrics, providing a valuable resource for training and testing HAR models, particularly for mobile applications where sensor data must be processed in real time [31]. The UCI Machine Learning Repository provides a dataset for smartphone-based recognition of human activities and postural transitions, offering a foundation for developing HAR systems that can benefit from computational offloading to reduce device strain during real-time inference [32]. Huang et al. proposed a method for human activity recognition that integrates edge computing with GRU networks, reducing computational costs and enhancing real-time performance, showing how edge-based offloading can improve the efficiency of HAR systems [33]. Yao et al. introduced deep compressive offloading, which speeds up neural network inference by trading edge computation for reduced network latency, offering a promising strategy for enhancing the performance of mobile HAR systems [34]. Yang et al. explored offloading optimization in edge computing for deep learning-enabled target tracking by UAVs, demonstrating techniques that can be applied to HAR by optimizing the distribution of computational tasks between mobile devices and edge servers [35]. Wang et al. introduced deep convolutional networks with a tunable speed–accuracy tradeoff for HAR using wearables, providing a method to balance performance and energy consumption, crucial for optimizing mobile HAR systems [36]. In [37], Sarkar and Kumar proposed a deep learning-based energy-efficient offloading strategy in heterogeneous fog computing networks, showing how optimized offloading can improve both energy efficiency and computational performance in HAR systems. Huang et al. applied deep reinforcement learning to optimize computation offloading in mobile-edge computing networks, providing a dynamic approach to managing offloading tasks based on network conditions and computational loads, which can enhance the performance of HAR systems [38]. Aghapour et al. introduced a task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI tasks in IoT edge environments, offering a scalable solution for HAR systems that rely on continuous real-time data processing [39].

Here are the contributions of the paper, listed as titles:

1. Optimization of Computational Offloading for Deep Neural Networks
2. Proposing Three Methods for Data Reduction:
    - Data Sample Reduction (Deleting samples and interpolating on the cloud side)
    - Precision Reduction (Converting floating-point data to lower precision integers)
    - Compression (Using low-overhead compression techniques: lossy and lossless)
3. Evaluation of Data Reduction Methods on WISDM and UCI Datasets
4. Comparison of Data Reduction Methods for Activity Recognition
5. Demonstrating the Superiority of Precision Reduction for Data Volume Reduction
6. Assessing the Effectiveness of Lossy and Lossless Compression
7. Practical Implications for Optimizing Data Transmission in Cloud-Based Human Activity Recognition

Problem statement and literature review

In recent years, we have witnessed a significant increase in research and development of mobile networks. With advancements in mobile terminals and the rising popularity of smartphones, new mobile applications such as facial recognition, image processing, interactive games, and augmented reality have gained considerable attention. Consequently, the expectations for mobile devices to run more demanding applications are increasing [1]. Nowadays, users utilize mobile phones for a wide range of daily activities, such as searching through music, playing video games, recording, editing, and uploading videos, analyzing their photo collections, indexing content, and managing financial affairs [2]. Despite the prominent role these devices play in individuals' lives, running complex applications on mobile devices poses challenges due to their limited resources, such as memory capacity, graphical processing speed, and battery power [3]. Mobile devices have relatively weak computational power, limited battery life, and hardware resources. Additionally, mobile applications typically require intensive computations and high energy consumption. Given the limitations of the computational resources available in mobile devices, these devices may not be able to run applications efficiently [4]. Today, due to the increasing demand for high-processing-volume applications, there is a need for powerful environments and resources capable of handling these heavy computations. One solution to this problem is known as offloading, in which low-power devices, such as mobile phones and Internet of Things (IoT) objects, offload their processing tasks to a cloud computing environment, delegating them to cloud servers [5]. Offloading can lead to energy savings and performance improvements, and it can also enhance the computational capabilities of mobile systems [2]. The computational operations of deep neural networks (DNN) involve two phases: training and inference, where the issue of offloading can be relevant in both phases. During the training phase, the parameters of the DNNs (such as the weights of the edges) are determined using pre-labeled input data, enabling the DNN to perform inference on previously unseen data during the inference phase. Each layer's processing can be considered a vector operation, where the parameters are iteratively updated as the DNN is trained with labeled data. Given that practical applications of smartphones are more prominent in the inference phase; most researchers focus on offloading computations for inference. Conversely, the training phase, due to its extensive computational resource requirements, is typically conducted on powerful servers [3]. Another reason for the limited offloading of the training phase is that once this phase is completed, the parameters of each layer remain fixed. Therefore, as long as the training data remains unchanged, the deep neural networks use the same parameters for inference on input data. Some researchers have focused on offloading all or parts of the computations of deep neural networks, proposing solutions to overcome obstacles such as battery usage limitations on mobile devices and their constrained computational resources. Offloading deep neural networks usually involves a trade-off, where savings in execution time and energy consumption come at the cost of reduced inference accuracy [6]. This trade-off is the focus of the present research, which aims to reduce the volume of data sent to the cloud during the inference phase by optimizing the first entry point of deep neural networks, thereby minimizing data size in various ways. The focus of this research is on human activity recognition (HAR). Activity recognition is crucial for providing services in the Internet of Things (IoT) world. Modern smartphones have become prominent devices for human activity recognition due to their ubiquity, sensing capabilities, and processing power. However, limited battery capacity and resources of smartphones hinder their full utilization for such sensing and processing capabilities [7]. Human activity recognition enables the detection of various physical activities performed by a smartphone user (such as walking, running, etc.) based on different inputs. This study aims to optimize communications in offloading the problem of human activity recognition with smartphones by examining the trade-off between the volume of transmitted data and recognition accuracy. Here, we have a comparison table about the previous methods in the literature and the current paper in Table 1.

Table 1: Comparative contributions from the references

Reference	**Method/Focus**	**Contributions**	**Comparison to Current Paper**
[1] Graves et al., 2013	Recurrent Neural Networks for Speech Recognition	Introduces deep recurrent neural networks (RNNs) for speech recognition.	Not directly related to offloading or activity recognition but foundational for deep learning methods applied in the current paper.
[2] Eshratifar & Pedram, 2018	Mobile Cloud Computing for DNNs	Optimizes offloading computation for deep neural networks in mobile	Focus on mobile-cloud offloading like the current paper, but doesn't specifically focus on

		environments, considering energy and performance.	reducing data volume, which is the focus of the current paper.
[3] Dey et al., 2019	Edge Computing for Deep Learning	Challenges and insights on offloading deep learning execution to edge devices.	Relevant to the computational offloading context, but current paper looks at optimizing data reduction in transmission rather than edge offloading.
[4] Huang et al., 2019	Hybrid Device-Edge-Cloud Execution	Proposes DeePar for hybrid execution of mobile deep learning apps.	Both papers address mobile deep learning, but current paper is focused on reducing data sent to the cloud, while this paper considers hybrid execution strategies.
[5] Kemp et al., 2010	Computation Offloading Framework	Focus on computation offloading for smartphones in mobile computing.	Early work on offloading, providing a framework that might serve as a basis for the current paper's offloading techniques, but with less focus on data reduction.
[6] Ran et al., 2017	Offloading Deep Learning to Mobile	Investigates offloading deep learning tasks to mobile devices via offloading.	Similar offloading context, but the current paper includes more specific data reduction methods.
[7] Fadishei, 2018	Energy-Efficient Activity Recognition	Focus on energy-efficient activity recognition using smartphones.	Related to activity recognition but does not focus on offloading or data reduction.
[8] Mahmoodi et al., 2016	Joint Scheduling and Cloud Offloading	Focus on scheduling and offloading decisions in cloud mobile applications.	Similar offloading concept, but current paper emphasizes methods to reduce the volume of data, not just scheduling.
[9] Messaoudi et al., 2018	Mobile Gaming Computation Offloading	Focus on offloading computation for mobile gaming applications.	Focus on a specific use case (gaming), while the current paper focuses on human activity recognition with data reduction methods.
[10] Zhang et al., 2016	Energy-Efficient Offloading for Real-Time Video	Examines energy-efficient offloading for video applications.	Different application (video vs. activity recognition) but similar offloading concerns; the current paper adds more focus on data reduction strategies.
[11] Deyannis et al., 2018	Edge Offloading for Antivirus	Focus on edge-assisted computation offloading for antivirus protection.	Similar edge/offloading context but focuses on a different application (antivirus), while the current paper addresses human activity recognition.

[12] Wang et al., 2013	Mobile Web Offloading	Selective offloading to accelerate mobile web performance.	Early work on selective offloading; the current paper builds on similar concepts but focuses on optimizing data size rather than computation offloading itself.
[13] Guo et al., 2019	FPGA-based Neural Network Inference	Surveys FPGA-based accelerators for deep learning inference.	Focuses on hardware accelerators for inference, while the current paper focuses more on optimizing cloud communication for deep learning tasks.
[14] Jeong et al., 2018	Computation Offloading for ML Web Apps	Computation offloading for machine learning web apps in edge environments.	Similar concept of offloading computations but the current paper's contribution is more focused on reducing data size rather than offloading execution.
[33] Huang et al., 2023	Edge Computing for Human Activity Recognition	Proposes a deep learning method for activity recognition using edge computing.	Similar in application (human activity recognition) and edge context, but the current paper focuses more on transmission data reduction techniques.
[34] Yao et al., 2020	Deep Compressive Offloading	Proposes a method to reduce network latency by offloading deep learning tasks.	Both explore offloading, but the current paper emphasizes data reduction methods to improve offloading efficiency.
[35] Yang et al., 2020	Target Tracking with Deep Learning	Offloading optimization in edge computing for target tracking in IoT.	Offloading optimization in a similar edge context but focused on different application (target tracking vs. activity recognition).
[36] Wang et al., 2021	Deep Convolutional Networks for Human Activity Recognition	Focus on deep learning for human activity recognition using wearables.	More aligned with the application of the current paper, but the current paper focuses more on reducing data volume sent to the cloud.
[37] Sarkar & Kumar, 2022	Energy-Efficient Offloading in Fog Networks	Proposes an energy-efficient strategy for computational offloading in fog networks.	Both papers address offloading, but the current paper goes deeper into optimizing the data sent rather than just energy.
[38] Huang et al., 2019	Reinforcement Learning for Computation Offloading	Deep reinforcement learning for offloading decisions in mobile-edge networks.	Focuses on decision-making strategies for offloading, while the current paper emphasizes optimizing data for transmission.

Journal of Information Systems Engineering and Management
2025, 10(35s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Proposed Method

As we know, deep neural network models are large enough that even considering a portion of the model on a mobile phone incurs significant overhead. Additionally, deep neural network models are typically dynamic, requiring frequent updates to their parameters and weights, making sending part or all of the model to a mobile phone costly. Therefore, this research focuses on the first entry point of deep neural networks, which is the data sent for inference, aiming to reduce its volume through various means. To date, comprehensive research on the impact of approaches to reducing the input data volume in offloading deep neural networks has not been conducted. In study [26], compression approaches were used to optimize the traffic sent to the deep neural network. In contrast, the present study investigates the effects of different approaches, where compression is one of them. Furthermore, unlike study [26], there is no requirement to use a specific type of deep neural network that imposes limitations on its applicability. The objective of this research is to optimize the offloading of computations in deep neural networks by reducing the volume of data sent to the cloud. Various approaches exist that can reduce the volume of data sent to the cloud, thereby improving the efficiency of offloading operations. Our proposed method includes three approaches: reducing the number of data samples, reducing the precision of data samples, and compressing data sample instances. However, it should be noted that each of these approaches creates a trade-off between accuracy and inference cost by deep neural networks, with the aim in the current study being to find effective and optimal points in these trade-offs, if available. The overall framework of the proposed method to achieve this objective is illustrated in Fig. 1 and consists of explaining these three approaches in the following sections.

3-1. Reducing the Number of Data Samples

The first proposed approach in this study to optimize the offloading of computations in deep neural networks is reducing the number of data samples sent to the cloud. Reducing the number of input data samples always decreases the volume of data sent to the cloud, leading to reduced communication costs (time, price, and energy consumption). Additionally, in some cases where input data is received from sensor sources (such as the case study of activity recognition described later), it can lead to energy savings in data sampling. This is because the energy consumed by the sensor layer in smartphones increases proportionally with the sampling rate. Therefore, reducing the number of samples received per unit of time reduces energy consumption on the mobile side. However, reducing samples will lower the accuracy of inference by deep neural networks. The aim of this research is to study the trade-off between accuracy and communication costs and to find an optimal point in this trade-off, if possible. It remains to be seen whether restoring some of the deleted samples can partially recover lost accuracy to some extent. For this purpose, the authors have experimented with different deletion scenarios of intermediate data samples—preserving one sample out of two, preserving one out of three, and preserving one out of four—and then recovering them using linear and cubic interpolation methods.

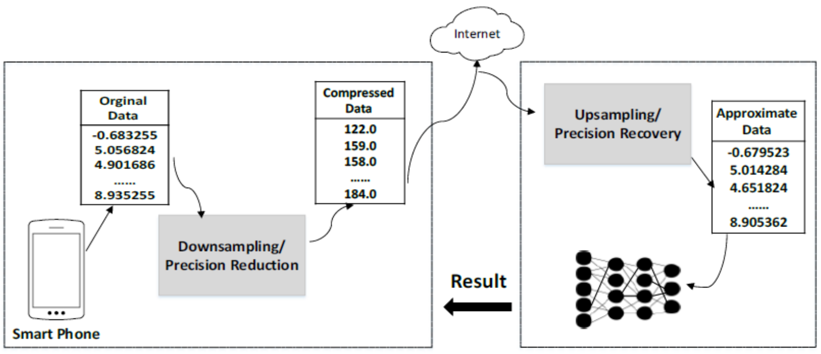


Fig. 1. General framework of the proposed method in this study

Interpolation is a well-established process for estimating values between specified data points. In this study, two types of interpolation methods—linear and cubic—have been employed to determine values at points between given data points. Assuming linearity in the variations, arithmetic mean interpolation can be performed according to Equation (1):

$$y = y_a + (y_b - y_a)\frac{x - x_a}{x_b - x_a} \tag{1}$$

where x_a and $x_b$are two existing points, and x is the point of interest for interpolated value estimation [28]. Additionally, considering a higher number of data points, interpolation with polynomials of higher degrees can be performed. For cubic interpolation, Equation (2) is used, where n represents the number of points, and x and y are the knot points to find the interpolated value [29]:

$$y_{n+1} = a_a + b_n x_{n+1} + c_n x_{n+1}^2 + d_n x_{n+1}^3 \tag{2}$$

3-2. Reducing the Decimal Precision of Data Samples

The second proposed approach in this paper for optimizing the offloading of deep neural network computations involves reducing the precision of the decimal (Precision) of the data samples sent to the cloud. To optimize offloading, the decimal precision of data can be reduced so that the number of bits required for each data sample decreases, reducing the volume of data sent to the cloud and saving energy consumption. It is evident that this method, similar to the previous approach, involves a trade-off between precision and the volume of transmitted data. Using the linear mapping relationship 2, where x is the initial variable and n is the number of bits, sensor data can be encoded as 8-bit, 16-bit, etc., integers instead of floating-point representation (as shown in Fig. 3-2. a). For example, as illustrated in Fig. 3-2. b, for 8-bit encoding, the smallest sensor reading is mapped to zero and the largest to 255, with other values proportionally mapped within this range.

$$f(x.n) = round\left(\frac{(x - \min(x))}{\max(x) - \min(x)} \times (2^n - 1)\right) \tag{3}$$

After receiving the reduced-precision data on the cloud side, they can be restored to values close to the original data using the inverse mapping relationship 4, as depicted in Fig. 3-2.c:

$$\mathrm{g}(\dot{x}.n) = \min(x) + \frac{(\min(x) - \min(x)}{(2^n - 1))} \times \dot{x}$$

In this relationship, x is the initial variable, $\dot{x}$ is the mapped variable after precision reduction, and n is the number of bits allocated to each data sample.

3-3. Data Sample Compression

Another approach investigated in this study for optimizing the outsourcing of deep neural network computations is the compression of data samples. Since most real-world data exhibit statistical redundancy, this method employs compression algorithms that typically utilize statistical redundancy to represent the sender's information more concisely, thus reducing the volume of data sent to the cloud. The aim of data compression is to reduce the data volume without causing significant alterations to its content. This approach aims to balance the trade-off between compression costs and communication costs, exploring whether imposing compression costs can substantially reduce communication costs. It is essential to consider that, despite the availability of various high-compression-ratio algorithms, a method with very low computational overhead must be chosen. One lightweight method for compressing data streams is the delta method. This method, assuming that the variation ranges of each data sample relative to the previous one is sufficiently small, transmits the differences with fewer bits instead of sending the full samples. Evidently, if this assumption does not hold, the aforementioned algorithm cannot effectively compress the data. In this research, both loss and lossless delta compression methods have been utilized.

Orginal Data
-0.964638
5.012288
4.903325
-0.612916
-1.184970
9.000000
-1.460000
8.880000

(a)

Compressed Data
122.0
159.0
158.0
122.0
119.0
184.0
117.0
184.0

(b)

Approximate Data
-0.683255
5.056824
4.901686
-0.683255
-1.148667
8.935255
-1.417647
8.935255

(c)

Fig. 2. The Effect of Data Recovery on 8-bit Precision Reduction

3-3-1. Loss Delta Compression

In the lossless compression approach, data compression and decompression do not result in any data loss. As shown in Fig. 3, this method operates under the assumption that the data samples are (n) bits (where (n = 32, 16, 8, 4)), the delta is (m) bits (where (m = 16, 8, 4, 2)), and (m < n). For example, if (n = 32) and (m = 16), instead of transmitting (n)-bit numbers to the cloud, their differences are transmitted in (m)-bit form. If the difference between data samples exceeds (m) bits (the delta capacity), the original data sample must be sent, and to distinguish this case, a marker (reserved data) is placed before the original data sample. The smallest negative number in the delta range is used as the reserved marker. Consequently, due to the overhead of these markers, the final compressed data volume might be larger than the initial data volume. This scenario occurs when the variation range between consecutive data samples is large, necessitating frequent use of the marker.

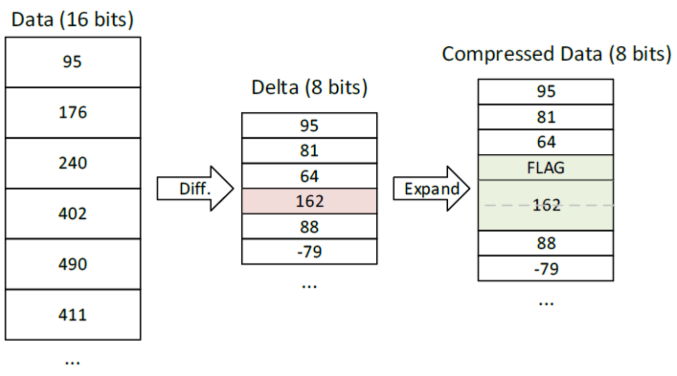


Fig. 3. Lossless Delta Compression of Data Samples

3-3-2. Compression with delta loss

The Compression with delta loss retains an approximate representation of the original data while sacrificing some of the original data in favor of a higher compression ratio. As shown in Fig. 4, when the difference between data samples exceeds (m) bits (in this figure, the specific case of (m = 8)), the maximum value representable in 8 bits is transmitted, and subsequent samples are used to correct the approximation. Consequently, in Lossless compression, an increase in data volume is not observed. However, due to the inherent data loss, the reduction in volume comes at the cost of precision.

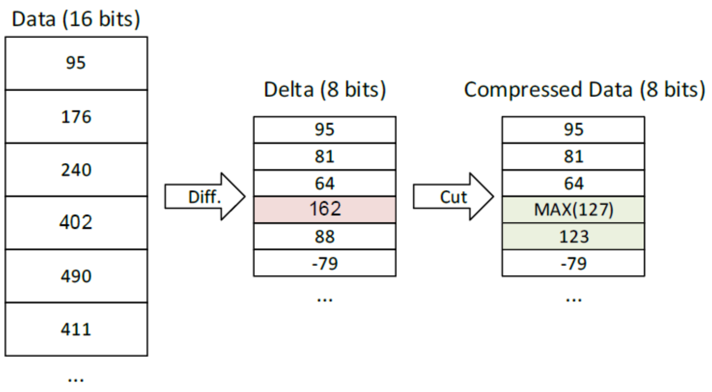


Fig. 4. Lossy delta compression of data samples

In the next chapter, experiments are conducted to evaluate the effectiveness of the proposed methods in this research and to study the trade-off between inference accuracy and the volume of data sent to the cloud.

RESULTS AND ANALYSIS

4-1. Implementation

Nearly every modern smartphone is equipped with a three-axis accelerometer that measures acceleration in all three spatial dimensions. In this study, the authors used data collected from accelerometer sensors and trained a Long Short-Term Memory (LSTM) neural network, implemented in Python using the TensorFlow library, as the reference model for experiments to recognize human activities (HAR) from accelerometer data. The neural network model comprises two fully connected layers and two LSTM layers with 64 units each.

4-2. Dataset

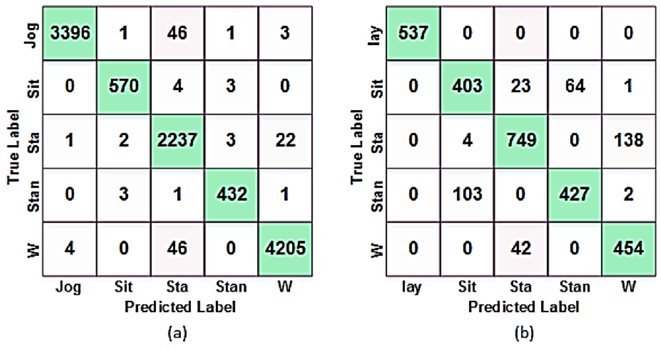
The proposed methods in this research were evaluated using data collected from accelerometer sensors, specifically the WISDM [31] and UCI [32] datasets. The WISDM dataset contains raw accelerometer and gyroscope data collected from smartphones and smartwatches at a rate of 10 Hz. It shows the distribution of values along the x, y, and z axes. The sensor data were collected from 51 subjects performing 18 activities for 3 minutes each. The data are segmented into 10-second intervals, with each sensor reading occurring every 100 milliseconds. For each axis, there are 100 consecutive readings for each activity, resulting in 300 decimal numbers per activity. The raw data in this dataset contain 8 decimal places, and each sample includes 100 numbers. There is a total of 1,098,203 samples in this dataset, which includes over two million samples across five activity classes: walking, running, stair activities, sitting, and standing. Eighty percent of the data is used for training, and the remaining twenty percent is used for testing.

The UCI dataset contains raw data with three axes (x, y, and z) collected at a fixed rate of 50 Hz using an accelerometer and gyroscope. The sensor signals (accelerometer and gyroscope) are divided into 2.56-second segments. Each sample includes 128 numbers, and there are 2,947 samples in total. This dataset includes five activity classes: sitting, standing, stair activities, walking, and lying down. Seventy percent of the data is used for training, and the remaining thirty percent is used for testing.

4-3. Experiments

In this section, an effort is made to study the optimal trade-off between accuracy and communication cost based on the proposed methods. To this end, the inference accuracy of the deep neural network for the original data samples from the WISDM dataset was first tested without the proposed methods. Fig. 4-1 (a) details the accuracy of activity recognition for this dataset, broken down by activity. The activities in this Fig. include Jogging (Jog), Sitting (Sit), Stair climbing (Sta), Standing (Stan), and Walking (W). The overall recognition accuracy in this case (without data transmission volume reduction) is 98.7%.

Next, the inference accuracy of the deep neural network for the original data samples from the UCI dataset was tested without the proposed methods. Fig. 5 (b) details the accuracy of activity recognition for this dataset, broken down by activity. The activities in this Fig. include Lying (Lay), Sitting (Sit), Stair climbing (Sta), Standing (Stan), and Walking (W). The overall recognition accuracy in this case (without data transmission volume reduction) is 87.8%.

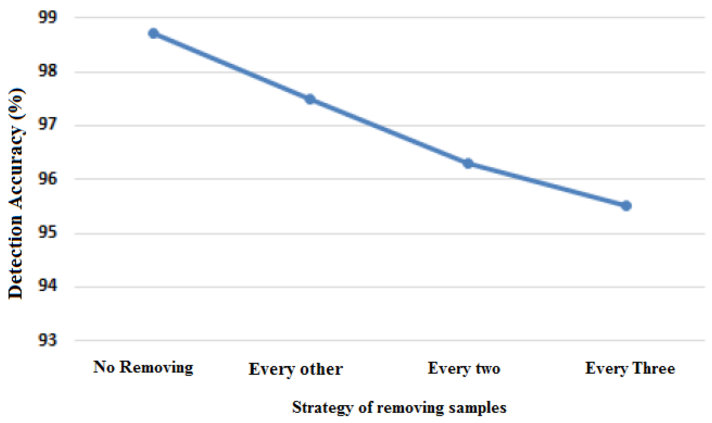

True Label
Predicted Label
(a)
(b)

True Label \ Predicted Label (a)	Jog	Sit	Sta	Stan	W
Jog	3396	1	46	1	3
Sit	0	570	4	3	0
Sta	1	2	2237	3	22
Stan	0	3	1	432	1
W	4	0	46	0	4205

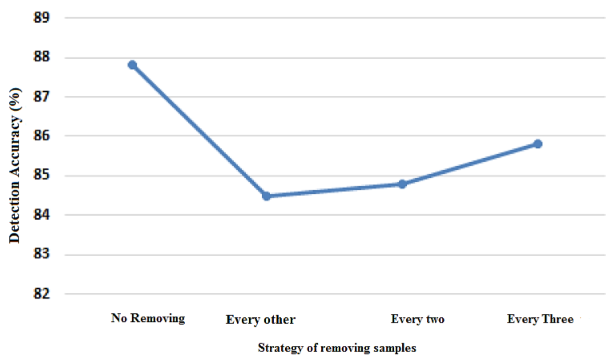
True Label \ Predicted Label (b)	lay	Sit	Sta	Stan	W
lay	537	0	0	0	0
Sit	0	403	23	64	1
Sta	0	4	749	0	138
Stan	0	103	0	427	2
W	0	0	42	0	454

Fig. 5. Activity Recognition Accuracy by Activity without Data Reduction

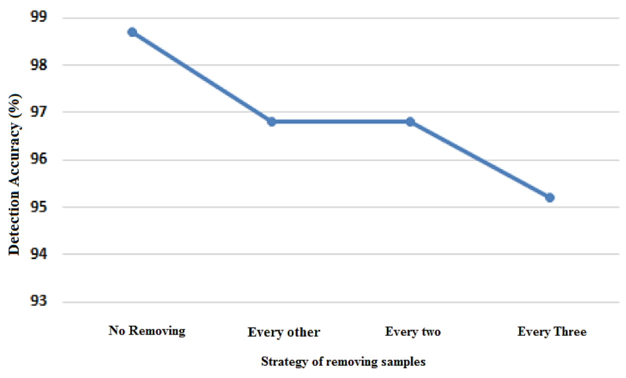
In the first experiment, the proposed method for reducing the volume of data sent to the cloud by decreasing the number of samples was evaluated. To this end, a fraction of the data was reduced by removing intermediate samples at various intervals (i.e., removing every other sample, every third sample, and every fourth sample) and then recovered using interpolation methods, specifically linear and cubic interpolation. Both the learning and inference steps were performed with the reduced data. Fig. 6 shows the effect of removing data samples before sending them to the cloud on accuracy. Part (a) shows the effect of removing every other sample, every two samples, every three samples, and every four samples with linear interpolation recovery for the WISDM dataset. Part (b) shows the effect of various intermediate sample removal scenarios with linear interpolation recovery for the UCI dataset. Part (c) shows the effect of removing every other sample, every third sample, and every fourth sample with cubic interpolation recovery for the WISDM dataset. Part (d) demonstrates the effect of various intermediate sample removal scenarios with cubic interpolation recovery for the UCI dataset. As shown in the figure, the sample removal approach can generally be considered beneficial. For example, removing every other sample reduces the data transmission volume by 50% while only negatively impacting prediction accuracy by 1%. Additionally, no significant difference is observed between linear and cubic interpolation recovery methods.



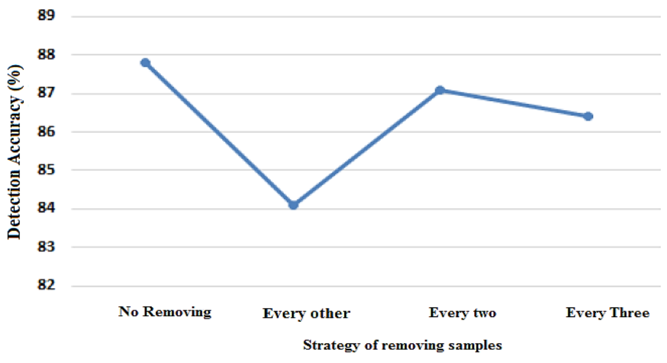
(a) Linear Interpolation Recovery of the WISDM Dataset



(b) Linear Interpolation Recovery of the UCI Dataset



(c) Cubic Interpolation Recovery of the WISDM Dataset



(d) Cubic Interpolation Recovery of the UCI Dataset

Fig. 6: The impact of data point removal before transmission to the cloud on accuracy.

Fig. 7 shows the effect of data point removal on activity-specific detection accuracy for the WISDM dataset, while Fig. 8 displays the impact on the UCI dataset for linear and cubic interpolation recovery methods under different removal scenarios. As observed in the figures, the primary negative impact on detection accuracy pertains to activities involving ascending and descending stairs. Hence, detection accuracy is activity dependent.

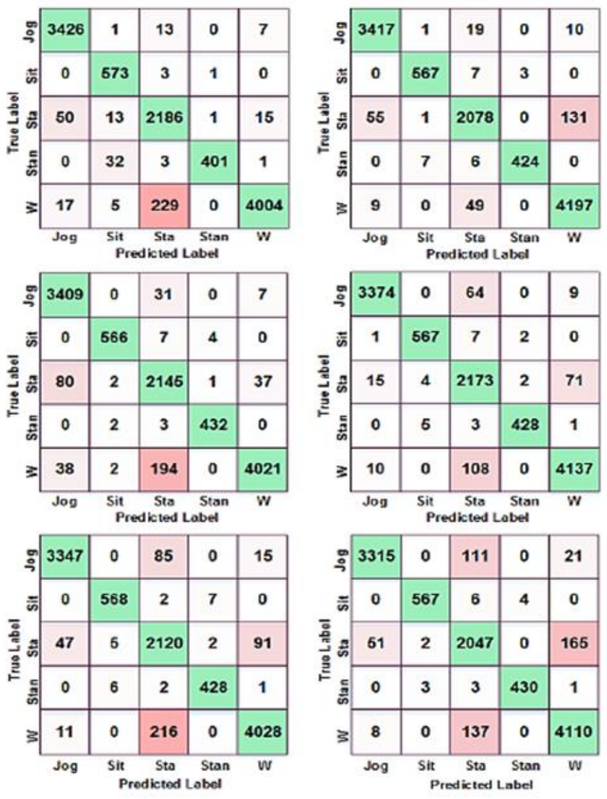


Fig. 7. Impact of WISDM Data Point Removal on Activity-Specific Detection Accuracy for Linear Interpolation (left) and Cubic Interpolation (right) with Removal of One in Between (top), Two in Between (middle), and Three in Between (bottom).

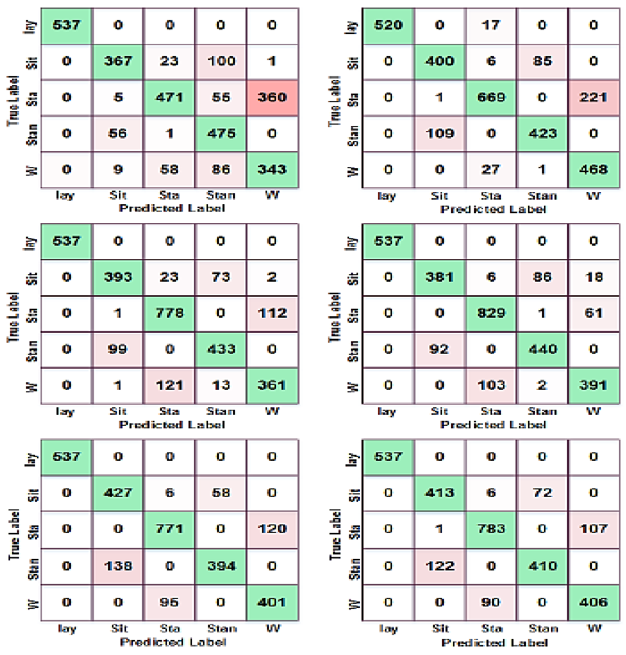
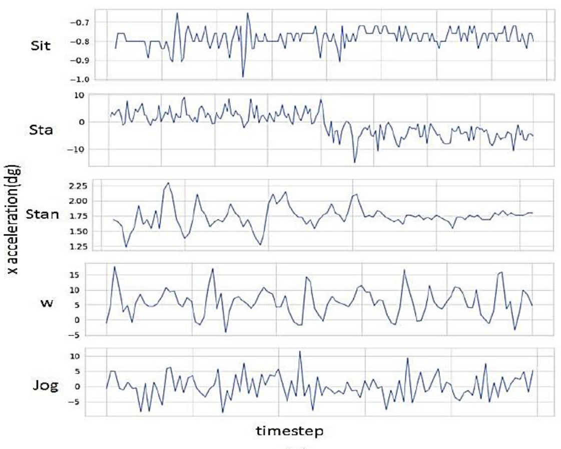
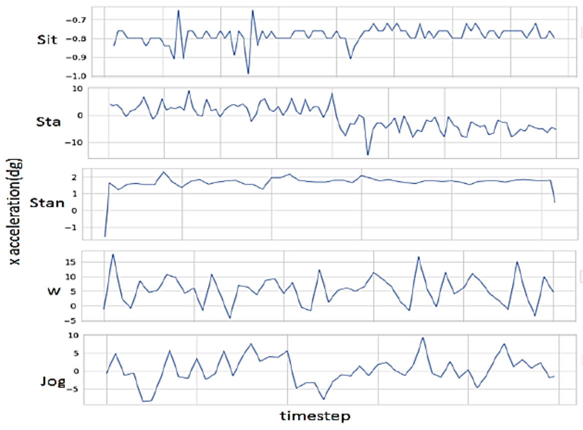


Fig. 8. Impact of UCI Data Point Removal on Activity-Specific Detection Accuracy for Linear Interpolation (left) and CubicInterpolation (right) with Removal of One in Between (top), Two in Between (middle), and Three in Between (bottom).

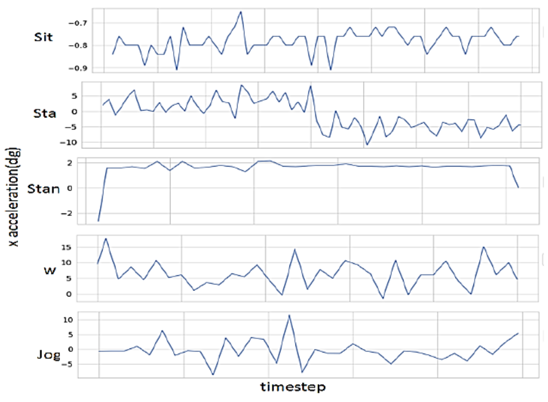
Fig. 9 details the effect of reducing the number of data points on the acceleration signal along the x-axis for activity-specific examples in the original data set in (a), removal of one in between in (b), removal of two in between in (c), and removal of three in between with linear interpolation in (d) for the WISDM data set, illustrating how acceleration changes over time in different activities.



(a)



(b)



(c)

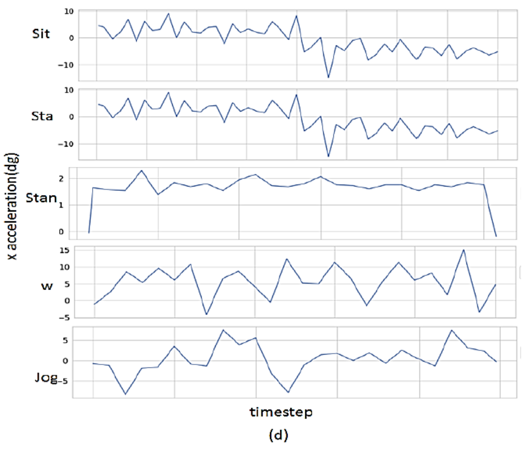


Fig. 9. The Effect of Reducing the Number of Samples and Linear Interpolation on the X-axis Signal

Fig. 10 details the effect of reducing the number of data points on the acceleration signal along the x-axis for activity-specific examples in the original data set in (a), removal of one in between in (b), removal of two in between in (c), and removal of three in between with cubic interpolation in (d) for the WISDM data set.

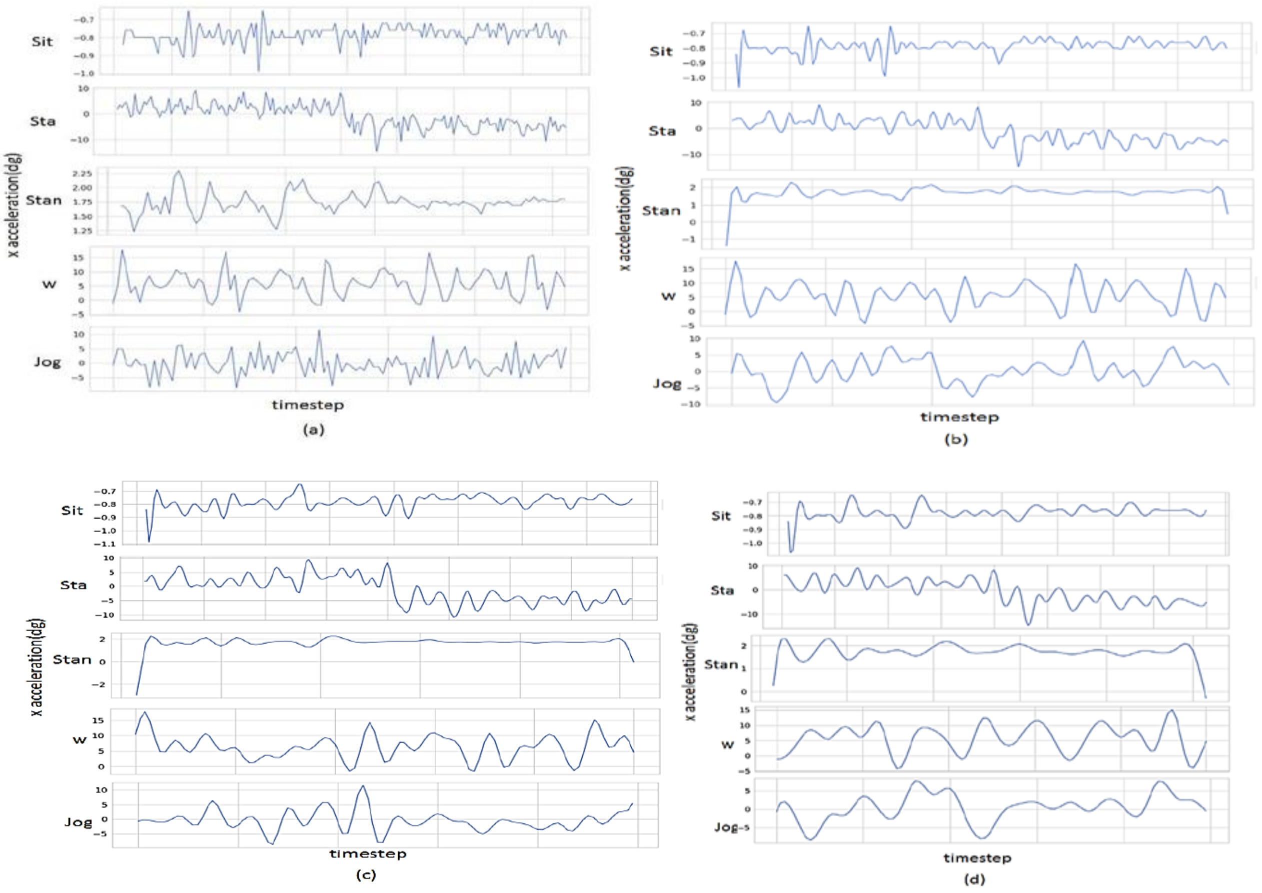


Fig. 10. The Effect of Reducing the Number of Samples and Cubic Interpolation on the X-axis Signal.

One of the other methods for reducing the volume of data sent to the cloud is precision reduction. Before transmission to the cloud, data are transformed into an integer number n-bit using a linear mapping function (n = 1, 2, 4, 8, 16, 32). After computational processing in the cloud, they are reverted to data close to the original using the inverse relationship. Fig. 11 (a) illustrates the impact of reducing the precision of data samples from the WISDM dataset prior to cloud transmission on accuracy for various linear mapping scenarios, while Fig. 11 (b) shows the impact on accuracy of reducing data samples from the UCI dataset for different linear mapping scenarios. As expected, in both datasets, reducing precision correlates with reduced recognition accuracy. However, this trade-off between accuracy and volume reduction is beneficial. For instance, in the 8-bit scenario of the WISDM dataset, achieving an eightfold reduction in transmitted volume (compared to 64-bit floating-point precision) results in only about a one percent decrease in recognition accuracy. It is noteworthy that even with further reduction in precision and using 4-bit precision, which reduces data volume by 16 times, recognition accuracy remains within an acceptable range. However, reducing precision to less than 4 bits severely compromises recognition accuracy to an unacceptable level.

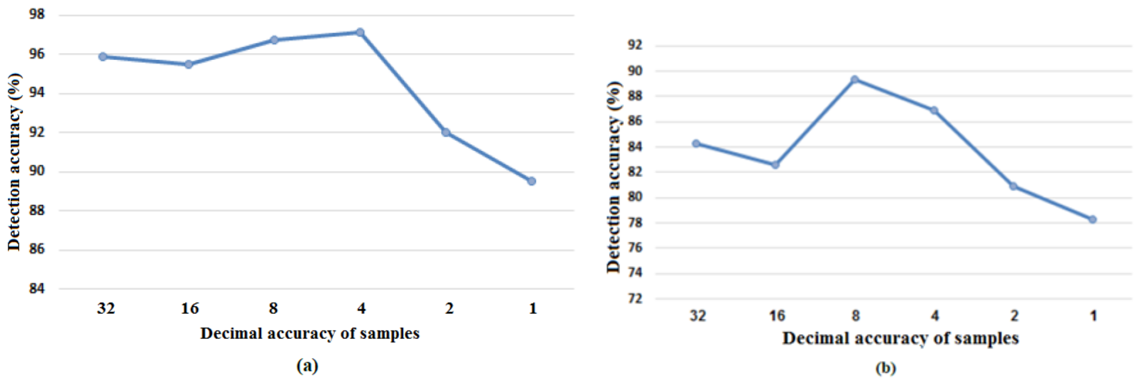
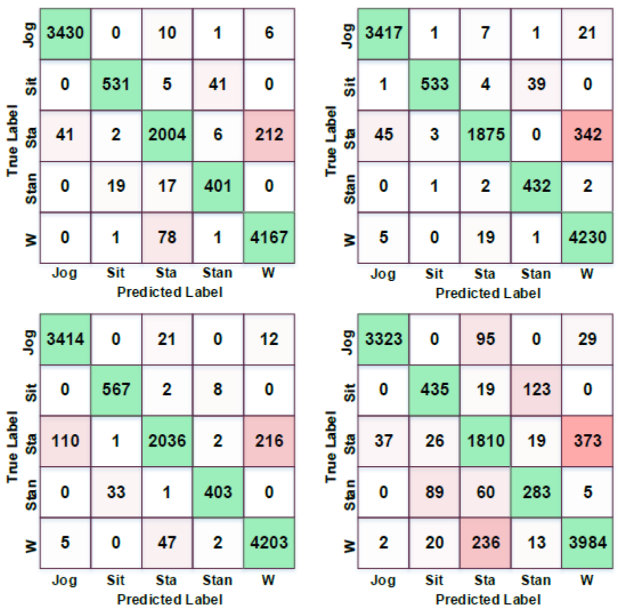


Fig. 11. The effect of reducing the precision of decimal data samples before transmission to the cloud on recognition accuracy for various linear mapping scenarios.

Fig. 12 illustrates the details of the effect of reducing the precision of decimal data samples from the WISDM dataset, while Fig. 13 details the effect of reducing the precision of decimal data samples from the UCI dataset on recognition accuracy for different activity classifications. As observed in these figures, like the first method, the most significant negative impact of reducing decimal precision occurs on the recognition of activities such as ascending and descending stairs. Moreover, excessive reduction in decimal precision has led to decreased recognition accuracy across various activities, particularly evident in the experiments with 1-bit precision


True Label
Predicted Label

	Jog	Sit	Sta	Stan	W
Jog	3430	0	10	1	6
Sit	0	531	5	41	0
Sta	41	2	2004	6	212
Stan	0	19	17	401	0
W	0	1	78	1	4167

	Jog	Sit	Sta	Stan	W
Jog	3417	1	7	1	21
Sit	1	533	4	39	0
Sta	45	3	1875	0	342
Stan	0	1	2	432	2
W	5	0	19	1	4230

	Jog	Sit	Sta	Stan	W
Jog	3414	0	21	0	12
Sit	0	567	2	8	0
Sta	110	1	2036	2	216
Stan	0	33	1	403	0
W	5	0	47	2	4203

	Jog	Sit	Sta	Stan	W
Jog	3323	0	95	0	29
Sit	0	435	19	123	0
Sta	37	26	1810	19	373
Stan	0	89	60	283	5
W	2	20	236	13	3984

Fig. 12: The impact of reducing the precision of decimal points in the WISDM dataset on the accuracy of activity recognition, for linear mappings of 32 bits (top left), 8 bits (top right), 16 bits (bottom left), and 1 bit (bottom right).

Journal of Information Systems Engineering and Management
2025, 10(35s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

Fig. 14 details the effect of reducing the precision of decimal points on the acceleration signal along the x-axis for activity-specific samples in the original dataset (a), with 8-bit precision (b), 4-bit precision (c), and 1-bit precision (d) for the WISDM dataset. The 1-bit precision (d) exhibits the most significant negative impact on decimal point accuracy reduction, resulting in decreased accuracy in activity recognition.

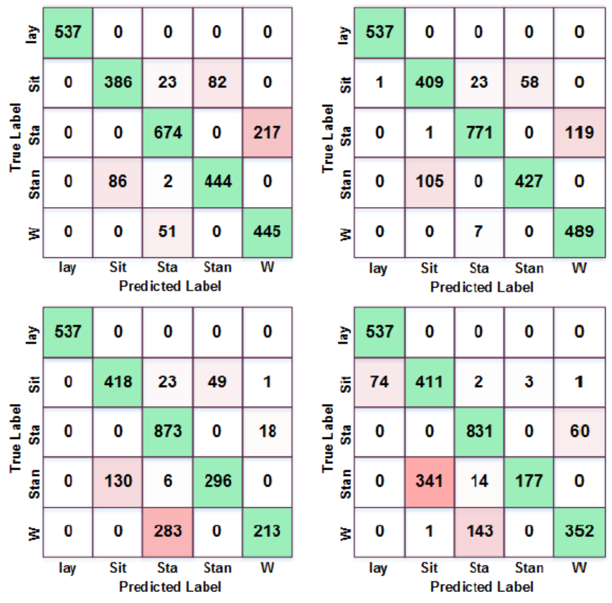
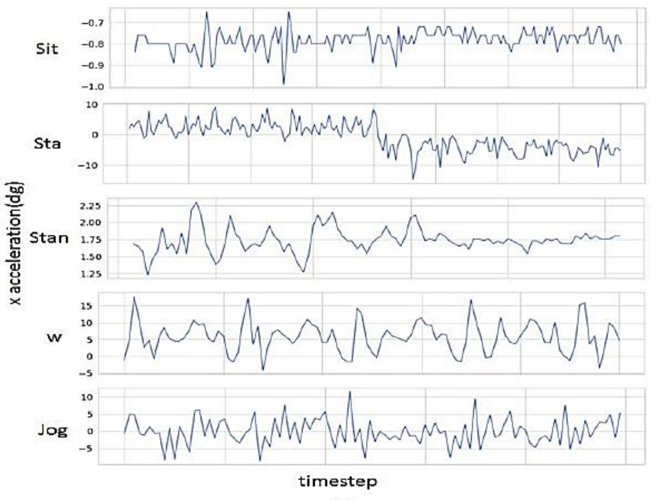


Fig. 13: The impact of reducing the precision of decimal points in the UCI dataset on the accuracy of activity recognition, for linear mappings of 32 bits (top left), 8 bits (top right), 16 bits (bottom left), and 1 bit (bottom right).

In data compression approaches, instead of transmitting n-bit numbers directly, their differences are sent in the form of m-bit deltas. Prior to this, data is converted from floating-point representation to fixed-point n-bit format. Both lossless and Lossless compression effects have been tested. The measure of compression effectiveness is the compression ratio defined by Equation (1).

$$\text{Compression ratio} = \frac{Uncompressed\ data\ size}{Compressed\ data\ size}$$



(a)

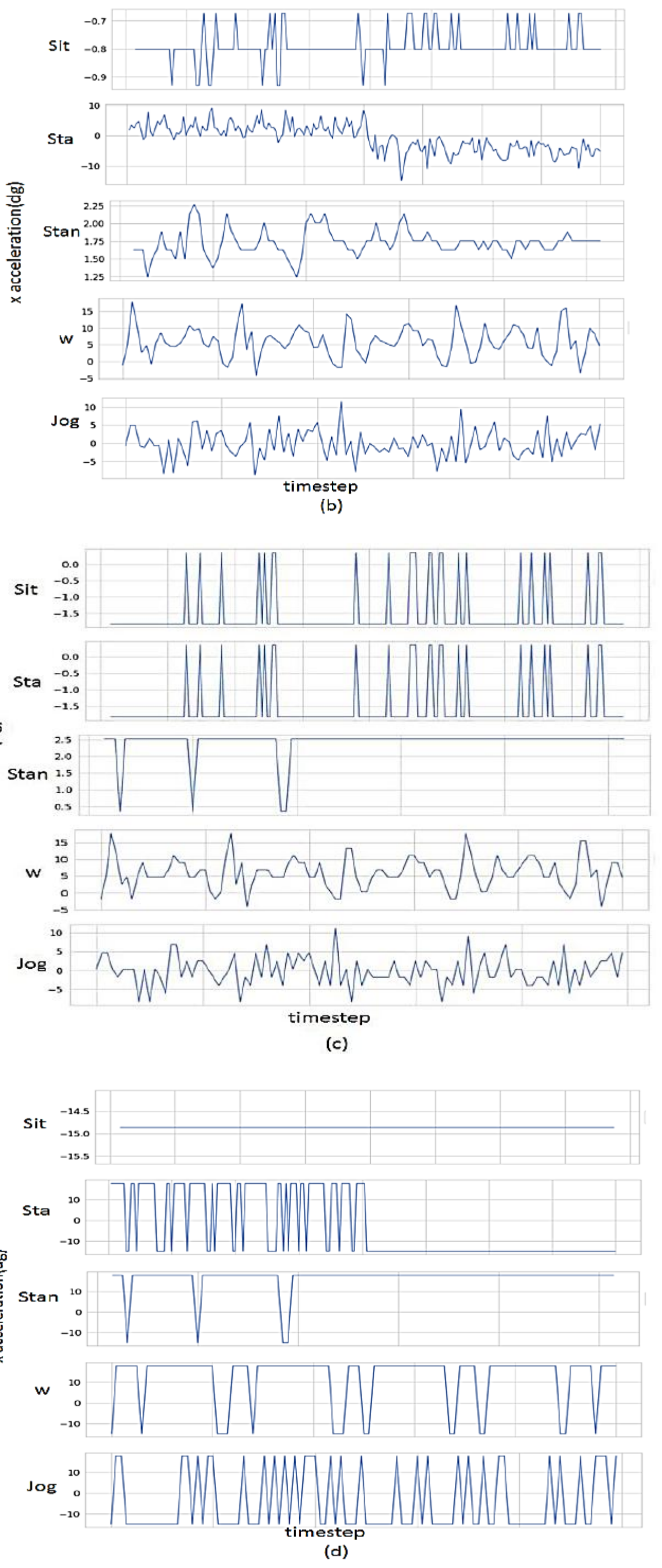


Fig. 14. The effect of reducing the precision of decimal points on the x-axis signal.

While lossless compression does not affect accuracy, its effectiveness must be assessed through the compression ratio measurement. On the other hand, proposed Lossless compression, despite offering a fixed compression ratio, should be evaluated for its impact on recognition accuracy. In the lossless scenario, rather than transmitting 32-bit fixed-point data directly, their differences are transmitted in 2, 4, 8, and 16-bit delta formats. If the data differences exceed delta capacity, the original data sample is transmitted, and an indicator is placed before the original data to distinguish this case. Despite compression, the final volume may exceed the initial volume. This process has been tested for 4, 8, and 16-bit data with 2, 4, and 8-bit deltas, and data volume has been examined. As Fig. 15 shows, in most cases, the final volume after compression exceeds the initial volume of the original data. Therefore, lossless delta compression is not an appropriate method for reducing the volume of data sent to the cloud, except for the 4-bit data case with 2-bit delta capacity.

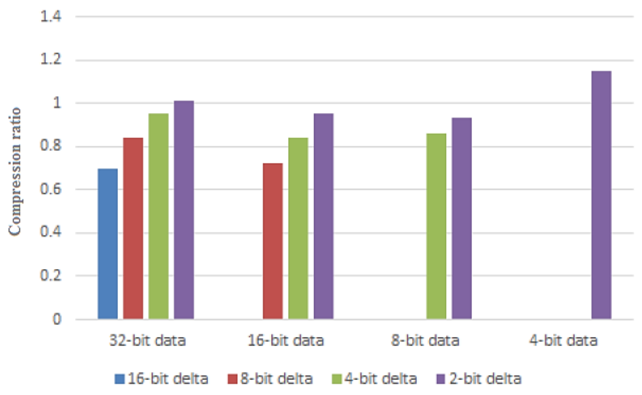


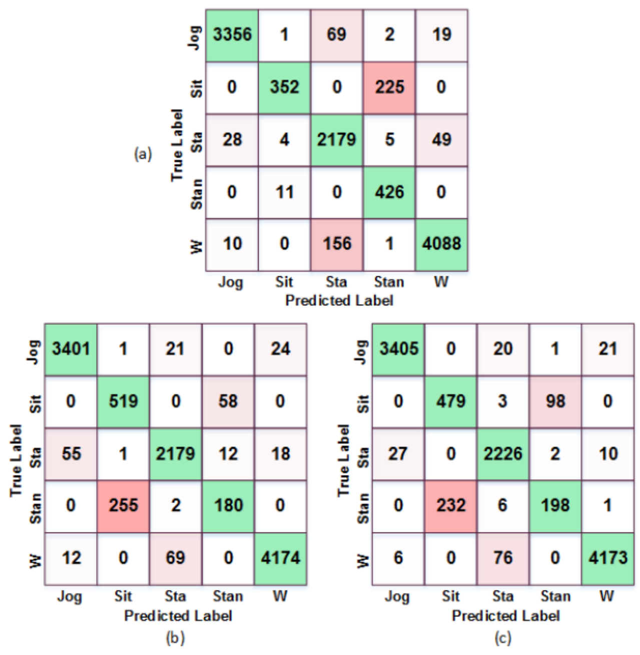
Fig. 15. The effect of lossless delta compression on data sample volume

Delta compression with loss maintains an approximate estimation of the original data, like Lossless compression. Instead of transmitting direct 32-bit fixed-point data, their differences are sent in the form of 2, 4, 8, and 16-bit deltas. The key difference lies in utilizing the maximum delta capacity for significant differences and using subsequent samples for correction. As a result, the compression ratio equals the ideal data size ratio to the delta size, but due to the loss of part of the original data, recognition accuracy decreases. In cases where the difference between data size and delta size is substantial, accuracy will significantly diminish. Therefore, experiments in this section were conducted with 4, 8, and 16-bit data compression using 2, 4, and 8-bit deltas, respectively, with accuracy results visible in Table 2.

Table 2. The impact of Lossless compression on data sample accuracy

Data sample size (bit)	delta size (bit)	Accuracy without compression	Accuracy with compression	Data overflow rate
8	6	96.7	94.7	0.27
4	3	97.0	95.4	0.09
4	2	97.0	95.0	0.29

Considering these results, Lossless delta compression performs better than lossless methods. For instance, in the case of 8-bit data with a 2% reduction in accuracy, the size of the transmitted data is halved. Fig. 16 illustrates the details of the impact of Lossless compression on the WISDM dataset's activity recognition accuracy for 8-bit data with 6-bit delta and 4-bit data with 2-bit and 3-bit deltas.



(a)

True Label \ Predicted Label	Jog	Sit	Sta	Stan	W
Jog	3356	1	69	2	19
Sit	0	352	0	225	0
Sta	28	4	2179	5	49
Stan	0	11	0	426	0
W	10	0	156	1	4088

(b)

True Label \ Predicted Label	Jog	Sit	Sta	Stan	W
Jog	3401	1	21	0	24
Sit	0	519	0	58	0
Sta	55	1	2179	12	18
Stan	0	255	2	180	0
W	12	0	69	0	4174

(c)

True Label \ Predicted Label	Jog	Sit	Sta	Stan	W
Jog	3405	0	20	1	21
Sit	0	479	3	98	0
Sta	27	0	2226	2	10
Stan	0	232	6	198	1
W	6	0	76	0	4173

Fig. 16. The effect of sample compression on the WISDM dataset's activity recognition accuracy. (a) 8-bit data with a 6-bit delta, (b) 4-bit data with a 2-bit delta, and (c) 4-bit data with a 3-bit delta.

In [26], researchers converted data to 8 bits and achieved an 84-fold savings in transmitted data by applying compression, losing only 2% accuracy. In contrast, this study experimented with different bit sizes and achieved up to a 32-fold reduction in transmitted data with a 2% loss in accuracy. However, the delta compression method is not as effective as the previous two methods in all cases.

CONCLUSION

This study focuses on optimizing outsourcing computational tasks to cloud platforms by reducing the volume of transmitted data, with an emphasis on human activity recognition using deep learning. Three methods for reducing the number of data samples, reducing the precision of data samples, and compressing data samples have been proposed and their efficacy tested on the WISDM and UCI datasets. In the first method, data samples are selectively deleted before transmission and estimated recovery methods are employed at the cloud side. In the second method, data samples are mapped to integer values with fewer bits using a linear mapping function before transmission, and an inverse mapping estimation is performed at the cloud side. In the third method, data samples are compressed using a low-overhead compression algorithm, either lossy or lossless, and recovered at the cloud side. Both proposed methods for reducing the volume of data samples and reducing the precision of data samples result in only a slight decrease in activity recognition accuracy. The method of reducing the precision of data samples demonstrates superiority due to a more significant reduction in data volume compared to the first method. Using only 4 bits for transmitting each data sample, which reduces the data volume by a factor of 16, incurs a limited accuracy loss of only one percent. Although the Lossless compression method yields better results compared to the lossless method, neither of these two methods matches the effectiveness of the precision reduction and data sample reduction methods.

REFERENCES

[1] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). Ieee

[2] Eshratifar, A.E. and Pedram, M., 2018, May. Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment. In Proceedings of the 2018 on Great Lakes Symposium on VLSI (pp. 111-116).

[3] Dey, S., Mondal, J. and Mukherjee, A., 2019, March. Offloaded execution of deep learning inference at edge: Challenges and insights. In 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 855-861). IEEE.

[4] Huang, Y., Wang, F., Wang, F. and Liu, J., 2019, April. DeePar: A hybrid device-edge-cloud execution framework for mobile deep learning applications. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 892-897). IEEE.

[5] Kemp, R., Palmer, N., Kielmann, T. and Bal, H., 2010, October. Cuckoo: a computation offloading framework for smartphones. In International Conference on Mobile Computing, Applications, and Services (pp. 59-79). Springer, Berlin, Heidelberg.

[6] Ran, X., Chen, H., Liu, Z. and Chen, J., 2017, August. Delivering deep learning to mobile devices via offloading. In Proceedings of the Workshop on Virtual Reality and Augmented Reality Network (pp. 42-47).

[7] Fadishei, H., 2018. Energy-Efficient Human Activity Recognition on Smartphones: A Test-Cost Sensitive Approach. International Journal of Information & Communication Technology Research, 10(3), pp.42-49.

[8] Mahmoodi, S.E., Uma, R.N. and Subbalakshmi, K.P., 2016. Optimal joint scheduling and cloud offloading for mobile applications. IEEE Transactions on Cloud Computing, 7(2), pp.301-313.

[9] Messaoudi, F., Ksentini, A. and Bertin, P., 2018, May. Toward a mobile gaming based-computation offloading. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.

[10] Zhang, L., Fu, D., Liu, J., Ngai, E.C.H. and Zhu, W., 2016. On energy-efficient offloading in mobile cloud for real-time video applications. IEEE Transactions on Circuits and Systems for Video Technology, 27(1), pp.170-181.

[11] Deyannis, D., Tsirbas, R., Vasiliadis, G., Montella, R., Kosta, S. and Ioannidis, S., 2018, June. Enabling gpu-assisted antivirus protection on android devices through edge offloading. In Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking (pp. 13-18).

[12] Wang, X.S., Shen, H. and Wetherall, D., 2013, August. Accelerating the mobile web with selective offloading. In Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing (pp. 45-50).

[13] Guo, K., Zeng, S., Yu, J., Wang, Y. and Yang, H., 2019. [DL] A survey of FPGA-based neural network inference accelerators. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 12(1), pp.1-26.

[14] Jeong, H.J., Jeong, I., Lee, H.J. and Moon, S.M., 2018, July. Computation offloading for machine learning web apps in the edge server environment. In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS) (pp. 1492-1499). IEEE.

[15] Eshratifar, A.E., Abrishami, M.S. and Pedram, M., 2019. JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services. IEEE Transactions on Mobile Computing.

[16] Karki, A., Keshava, C.P., Shivakumar, S.M., Skow, J., Hegde, G.M. and Jeon, H., 2019, March. Tango: A deep neural network benchmark suite for various accelerators. In 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 137-138). IEEE.

[17] Shi, S., Wang, Q., Xu, P. and Chu, X., 2016, November. Benchmarking state-of-the-art deep learning software tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* (pp. 99-104). IEEE.

[18] Qin, D., Yu, J., Zou, G., Yong, R., Zhao, Q. and Zhang, B., 2019. A novel combined prediction scheme based on CNN and LSTM for urban PM 2.5 concentration. IEEE Access, 7, pp.20050-20059.

[19] Durstewitz, D., Koppe, G. and Meyer-Lindenberg, A., 2019. Deep neural networks in psychiatry. Molecular psychiatry, 24(11), pp.1583-1598.

[20] Shah, S.D.A., Zhao, H.P. and Kim, H., 2018, October. Distributed deep neural networks with system cost minimization in fog networks. In TENCON 2018-2018 IEEE Region 10 Conference (pp. 1193-1196). IEEE.

[21] Nazemi, M., Eshratifar, A.E. and Pedram, M., 2018, March. A hardware-friendly algorithm for scalable training and deployment of dimensionality reduction models on FPGA. In 2018 19th International Symposium on Quality Electronic Design (ISQED) (pp. 395-400). IEEE.

[22] Hirsa, A., Karatas, T. and Oskoui, A., 2019. Supervised deep neural networks (DNNS) for pricing/calibration of vanilla/exotic options under various different processes. arXiv preprint arXiv:1902.05810.

[23] Gordon, M.S., Jamshidi, D.A., Mahlke, S., Mao, Z.M. and Chen, X., 2012. {COMET}: Code Offload by Migrating Execution Transparently. In 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12) (pp. 93-106).

[24] Li, H., Ota, K. and Dong, M., 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, 32(1), pp.96-101.
[25] Teerapittayanon, S., McDanel, B. and Kung, H.T., 2017, June. Distributed deep neural networks over the cloud, the edge and end devices. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (pp. 328-339). IEEE.
[26] Eshratifar, A.E., Esmaili, A. and Pedram, M., 2019, July. Bottlenet: A deep learning architecture for intelligent mobile cloud computing services. In 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) (pp. 1-6). IEEE.
[27] Jeong, H.J., Lee, H.J., Shin, C.H. and Moon, S.M., 2018, October. IONN: Incremental offloading of neural network computations from mobile devices to edge servers. In Proceedings of the ACM Symposium on Cloud Computing (pp. 401-411).
[28] Rosloniec, S., 2008. Fundamental numerical methods for electrical engineering (Vol. 18). Springer Science & Business Media.
[29] McClarren, R., 2017. *Computational nuclear engineering and radiological science using python*. Academic Press
[30] https://www.tensorflow.org/.
[31] Weiss, G.M., 2019. Wisdm smartphone and smartwatch activity and biometrics dataset. UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set.
[32] Data set: http://archive.ics.uci.edu/ml/datasets/(Smartphone Based Recognition of Human Activities and Postural Transitions)
[33] Huang, Xiaocheng, Youwei Yuan, Chaoqi Chang, Yiming Gao, Chao Zheng, and Lamei Yan. "Human Activity Recognition Method Based on Edge Computing-Assisted and GRU Deep Learning Network." *Applied Sciences* 13, no. 16 (2023): 9059.
[34] Yao, Shuochao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency." In *Proceedings of the 18th conference on embedded networked sensor systems*, pp. 476-488. 2020.
[35] Yang, Bo, Xuelin Cao, Chau Yuen, and Lijun Qian. "Offloading optimization in edge computing for deep-learning-enabled target tracking by internet of UAVs." *IEEE Internet of Things Journal* 8, no. 12 (2020): 9878-9893.
[36] Wang, Xing, Lei Zhang, Wenbo Huang, Shuoyuan Wang, Hao Wu, Jun He, and Aiguo Song. "Deep convolutional networks with tunable speed–accuracy tradeoff for human activity recognition using wearables." *IEEE Transactions on Instrumentation and Measurement* 71 (2021): 1-12.
[37] Sarkar, Indranil, and Sanjay Kumar. "Deep learning-based energy-efficient computational offloading strategy in heterogeneous fog computing networks." *The Journal of Supercomputing* 78, no. 13 (2022): 15089-15106.
[38] Huang, Liang, Suzhi Bi, and Ying-Jun Angela Zhang. "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks." *IEEE Transactions on Mobile Computing* 19, no. 11 (2019): 2581-2593.
[39] Aghapour, Zahra, Saeed Sharifian, and Hassan Taheri. "Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments." *Computer Networks* 223 (2023): 109577.