electronics

MDPI

Article

A Hybrid Deep Learning Model Using CNN and K-Mean Clustering for Energy Efficient Modelling in Mobile EdgeIoT

Dhananjay Bisen [1], Umesh Kumar Lilhore [2,3], Poongodi Manoharan [4,*], Fadl Dahan [5], Olfa Mzoughi [6], Fahima Hajjej [7], Praneet Saurabh [8] and Kaamran Raahemifar [9,10,11]

1 Department of Information Technology, Madhav Institute of Technology and Science, Gwalior 474005, Madhya Pradesh, India
2 Department of Computer Science and Engineering, Chandigarh University, Mohali 140413, Punjab, India
3 School of Computing, University of Louisiana, Lafayette, LA 70504, USA
4 Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha P.O. Box 5825, Qatar
5 Department of Management Information Systems, College of Business Administration, Hawtat Bani Tamim, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
6 Department of Computer Sciences, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
7 Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
8 Department of Computer Science and Engineering, Manipal University, Jaipur 303007, Rajasthan, India
9 Data Science and Artificial Intelligence Program, College of Information Sciences and Technology (IST), Penn State University, State College, PA 16801, USA
10 School of Optometry and Vision Science, Faculty of Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada
11 Faculty of Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada
* Correspondence: dr.m.poongodi@gmail.com

Abstract: In mobile edge computing (MEC), it is difficult to recognise an optimum solution that can perform in limited energy by selecting the best communication path and components. This research proposed a hybrid model for energy-efficient cluster formation and a head selection (E-CFSA) algorithm based on convolutional neural networks (CNNs) and a modified k-mean clustering (MKM) method for MEC. We utilised a CNN to determine the best-transferring strategy and the most efficient partitioning of a specific task. The MKM method has more than one cluster head in each cluster to lead. It also reduces the number of reclustering cycles, which helps to overcome the energy consumption and delay during the reclustering process. The proposed model determines a training dataset by covering all the aspects of cost function calculation. This training dataset helps to train the model, which allows for efficient decision-making in optimum energy usage. In MEC, clusters have a dynamic nature and frequently change their location. Sometimes, this creates hurdles for the clusters to form a cluster head and, finally, abandons the cluster. The selected cluster heads must be recognised correctly and applied to maintain and supervise the clusters. The proposed pairing of the modified k-means method with a CNN fulfils this objective. The proposed method, existing weighted clustering algorithm (WCA), and agent-based secure enhanced performance approach (AB-SEP) are tested over the network dataset. The findings of our experiment demonstrate that the proposed hybrid model is promising in aspects of CD energy consumption, overhead, packet loss rate, packet delivery ratio, and throughput compared to existing approaches.

Keywords: mobile edge computing; IoT; deep learning; cluster head; K-means; energy efficient algorithm

Citation: Bisen, D.; Lilhore, U.K.; Manoharan, P.; Dahan, F.; Mzoughi, O.; Hajjej, F.; Saurabh, P.; Raahemifar, K. A Hybrid Deep Learning Model Using CNN and K-Mean Clustering for Energy Efficient Modelling in Mobile EdgeIoT. *Electronics* **2023**, *12*, 1384. https://doi.org/10.3390/electronics12061384

Academic Editors: Martin Reisslein and Juan-Carlos Cano

Received: 15 January 2023
Revised: 16 February 2023
Accepted: 25 February 2023
Published: 14 March 2023

1. Introduction

MEC is an intelligent technique. Smartly examining partial computational offloading can decrease the energy usage of the client device (CDs) and quality service delay. It

is accomplished by breaking up a single task into multiple tasks under MEC. It enables practical data analysis of large amounts of information and acceleration with real-time, latency-free processing. Due to the popularity of IoT techniques, it is widely used in all intelligent applications. In these devices, all the components are connected via the internet and equipped with sensors, empowering them to detect real-time data from the surrounding environment. This mechanism has ultimately resulted in the fascinating idea of the IoT, where all intelligent objects, including connected vehicles, smart watches, notebooks, monitoring devices, and advanced manufacturing and relevance components, are linked through a group of channels and equipped with predictive analytics, forever evolving the way we function, reside, and perform [1].

As the IoT has grown, new challenges are also increasing. IoT systems utilise battery-operated devices to gather procedures and examine all the meaningful data. To enhance data integrity, the intelligent system connects all the pairing objects and sensors on multiple clusters, which leads to high energy demands [2].

The design and analysis of edge computing-based architectures have significant consequences for the future advancement of the IoT infrastructure. By utilising efficient clustering algorithms, the lifetime of the network and energy consumption can be improved. To accomplish these challenges, researchers are widely using AI-based IoT techniques, i.e., deep learning, machine learning, and fog computing, to generate more robust, flexible, productive, and precise solutions [3].

Mobile computing creates a discrete, non-centralised interconnected environment. This environment utilises various vital components, such as smartphones, IoT, sensors, cloud infrastructure, data processing, and storage infrastructure. Edge computing is very similar to IoT communication and mainly focuses on delivering the best connectivity, high data processing, and data transfer services to all the close-end nodes, primarily found at the network's boundary. Similarly, the MEC technique integrates the key features of mobile and edge computing. Due to the limited processing power and storage, the services of MEC systems can be affected, and it causes high energy utilisation and less reliability [4].

A MEC technique emerged as an advanced cutting-edge solution to many critical problems with cloud computing. In this technology, the computation server and individual applications are located near edge servers to improve the system performance, maximise the bandwidth, and achieve better reliability, high throughput, and less energy utilisation in client devices. It also enhances the computational power of client devices. Due to the complicated and power-intensive applications and services, the client devices have a restricted capacity for computational power and rechargeable batteries capacity. At the edge of wireless communication, MEC offers highly distributed computing resources and storage to CDs [5].

It is challenging for a CD with local storage and productive computational capacity to fulfil the demands of such high computational application domains; thus, in MEC, the CDs transfer the workload of such requests to the mobile edge server (MES) via data transfer to overcome computational delay and energy consumption of CDs. Similar to batteries, CDs' primary limitation is battery capacity. Even with native cloud services, CDs might not offer better customer service [6].

It is challenging for a CD to fulfil the demands of high computational applications with limited local storage and computational processing. A computational delay and high energy consumption became the biggest challenge in these applications. To deal with these issues, a MEC-based system allows all the CDs to transfer their high computing requests to the mobile edge server (MES) via a data transfer service. MEC systems have also restricted battery capacity [7].

Computational loading procedures in the MEC environment are classified into two types: (a) total loading and (b) partial loading. Total loading offloads the entire job to MES for implementation and operation. In contrast, partial loading divides the job into separate components, with some elements accomplished natively on CD and others mounted to MES for completion. Despite its benefits, the MEC system encounters several issues, e.g.,

data privacy and security, deployment protocol selection, energy consumption, and task scheduling. In past decades, researchers have given great attention to expanding IoT and mobile devices and substantial requests for sensitive areas, i.e., speech recognition, virtual reality, immersive gaming, Google glass, video progression, and object recognition [8].

Appliances with limited resources can encounter poor reliability and a terrible consumer experience. Users mainly utilise AI-based automation systems and MEC techniques to deal with such issues. With data security methods, these models can work in limited battery capacity to make precise forecasts and judgments. In MEC and IoT communication, a mass number of overloaded links can be the cause of the bottleneck. Deep learning and machine learning methodologies are vital in dealing with such issues [9]. In MEC systems, energy consumption is the most significant issue.

The primary goal of this research is to deal with energy issues in cluster head selection and cluster formation by establishing the most effective decision policies for MEC. To achieve the above, this research proposed a hybrid deep learning model based on CNN and the modified k-mean clustering (MKC) method for MECs called the "Energy-efficient cluster formation and head selection (E-CFSA) algorithm". The proposed hybrid model considers the partitioning by using a partial loading method, which determines the expense for each potential partitioning and loading strategy, and afterwards chooses the optimum.

In the proposed system, a cluster head selection and cluster formation process involves a rotational-based method to resolve the self-organisation and high availability characteristics. A master cluster head block directs a successful team to transmit the information to the base station efficiently. The proposed method also utilised a modified k-mean clustering method (MKM) to select the balanced cluster head and to choose more than one CH in a cluster to lead the group. The CHs selection depends on the length and timeliness of cluster nodes and allows more than two clusters to show in the group. It helps to reduce the reclustering cycles and achieves better energy and time results. Experimental analyses were performed on proposed and existing methods, i.e., WCA and AB-SEP. This research also allows IoT and mobile devices to discover pooled forecasting jointly.

The article is organised as follows: Section 2 discusses the literature on energy-efficient cluster head selection for MEC. Section 3 discusses the material and methods. It also covers the working of the proposed hybrid model. Section 4 covers the simulation results, discussion, analysis, and comparison. Section 5 discusses the conclusion and future work.

2. Literature Review

A natural energy optimisation problem is discussed in [10]. This research includes the performance review of energy utilisation, the transforming model for energy transformation, and improving the MEC system's power quality. The proposed techniques deal with energy challenges using collaborative block descent and fuzzy linear programming concepts. As of now, significant research initiatives have been dedicated to constructing offloading schemes for MEC networks.

Deep learning (DL) techniques need a lot of processing and memory to store training data and vast training models. A novel deep learning model is developed in [11]. The proposed model functions better on edge devices by implementing shallow features with limited processing capacity IoT equipment. A novel model is discussed in [12] when precise decisions must only be considered. The proposed deep learning model speeds up edge devices' knowledge acquisition and convolution layers. It also decreases the width of the components for multiclass classification [13]. An early disappearance of features in the MEC system with limited learning outcomes is always challenging [14]. The research proposed an enhanced CNN model using a modified layering technique for edge devices.

To predict the optimum computational and storage transferring techniques, three components-based models are discussed in [15]. The proposed model utilises state variables, decision behaviour, and scheme utility. The number of state variables helps to split the MEC process into static and dynamic transferring. It also utilises intelligent terminals, "I-Devices", which are more suitable for network infrastructure programs and services [16].

MEC systems have the higher processing power and deal with large amounts of data processing. Subsequently, article [17] discussed a deep reinforcement method for adaptive MEC. The proposed model maximises the MES data sampling and enhances the transfer rate. A comprehensive replica learning strategy to reduce MD service delays is discussed in [18]. A cost function of the proposed model deals with the service process and offers better communication and fewer service delays. Applying practical deep learning-based computing with a limited training dataset for MEC systems is difficult. The proposed model also suggested a high-rate binary transferring process with dynamic MEC networks [19]. This research deals with energy uncertainty issues by using CNN. To reduce the weight value of energy usage and latency, the proposed model uses a binary task allocation method based on time-varying workflows and different analysis features.

An energy-efficient method based on a deep learning technique was introduced in [20]. This research suggested a gradient-based deterministic strategy in MEC systems to solve the global optimisation problem. This article assesses a dense decentralised cellular network with multiple users, servers, and activities. The authors also considered MES and CD mobility for designing a region parallel task transferring model that enables reliable communication for low latency application areas. A comprehensive investigation of the emerging multimedia IoT is described in [21]. It also promotes several novel applications that enhance the overall quality of life by linking all the connected devices via emerging technological solutions [22].

The primary objective of the proposed model [23] was to emphasise the outline of MEC and its significant applications. MEC introduces edge devices among node sources and the cloud to prolong cloud services. This research also deals with the energy issues in MEC by enhancing the capabilities and optimising the load [24].

Table 1 discuss a comparison review of numerous existing research based on various parameter in the field of MEC energy consumption.

Table 1. Comparative analysis of existing research.

References	Method	Energy Consumption Model	Cluster Head Formation	Service Delayed	Partitioning of Task	Use of Multiuser and Mutiserver	Hybrid Deep Learning Model
[10]	Edge intelligent energy-efficient model	Y	N	Y	N	N	N
[11]	Hierarchical energy-efficient mobile-edge computing	Y	N	Y	N	Y	N
[12]	UAV-assisted mobile edge computing	Y	N	Y	N	N	N
[13]	Joint computation and communication cooperation	Y	N	Y	N	Y	N
[14]	Intelligent task prediction and offloading in less energy	Y	N	Y	N	Y	N
[15]	Energy-based routing	Y	N	Y	N	N	N
[16]	Offloading based on the reliability model	Y	N	Y	N	Y	N

Table 1. *Cont.*

References	Method	Energy Consumption Model	Cluster Head Formation	Service Delayed	Partitioning of Task	Use of Multiuser and Mutiserver	Hybrid Deep Learning Model
[17]	Energy efficient model using e-harvest	Y	N	N	N	Y	N
[18]	Offloading-based cost function	Y	N	N	Y	Y	N
[19]	Machine learning-based energy-saving model	Y	N	Y	N	N	N
[20]	AI-based cluster head selection	N	Y	Y	N	N	N
[21]	DNN based method	N	Y	Y	N	N	N
[22]	Energy-efficient routing protocol	Y	N	Y	N	N	N
[23]	Energy-aware mobile edge computing	N	Y	Y	N	N	N
[24]	Distributed deep learning-based task offloading	N	Y	Y	N	N	N
Proposed Hybrid Model	CNN with modified k-mean clustering	Y	Y	Y	Y	Y	Y

3. Materials and Methods

This section describes the features of the proposed and existing methods and covers the essential parameters and database specifications.

3.1. Proposed Hybrid Model

We proposed an "Energy-efficient model" E-CFSA for MEC in this research. The proposed hybrid model utilises CNN and modified k-mean clustering (MKC). Figure 1 shows the architecture of the proposed model.

The proposed model includes several functions, which provide for different phases. A detailed description of these phases is covered in the following subsections.

(A) Network initialisation phase is responsible for network variable declaration and initialisation and splits the network into small subgroups.
(B) Evolution of nodes: This is responsible for node selection. This phase calculates the trust among nodes. It utilises a modified version of k-means clustering to create the best-fit clusters.
(C) Cluster head selection: This phase utilises the CNN method for the best cluster head selection to save energy.
(D) Data transmission: This is the last phase of the proposed model and is responsible for data transmission.

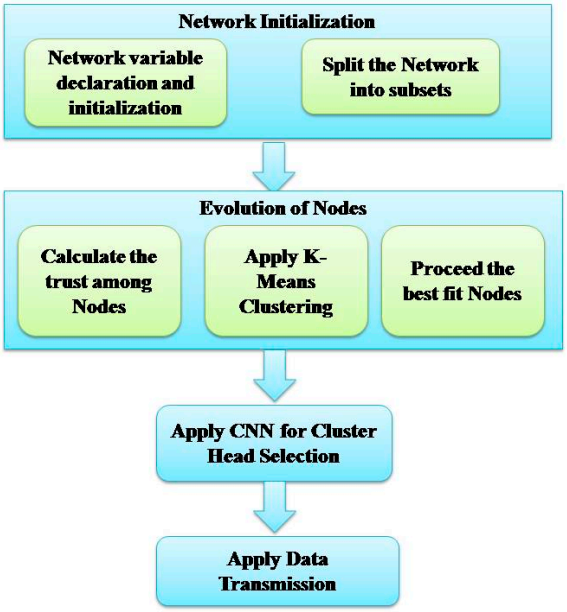


Figure 1. The architecture of the proposed hybrid model.

3.2. CNN Model

The CNN architecture is presented in Figure 2. In the proposed hybrid model, a CNN model uses three convolutional layers and a similar number of active, fully connected layers. In addition to the output layer, each layer in the CNN model of the proposed system is accompanied by a ReLu activation function (rectified linear function) over the sigmoid. The ReLu utilises a function (np. Heaviside (x, 1)).

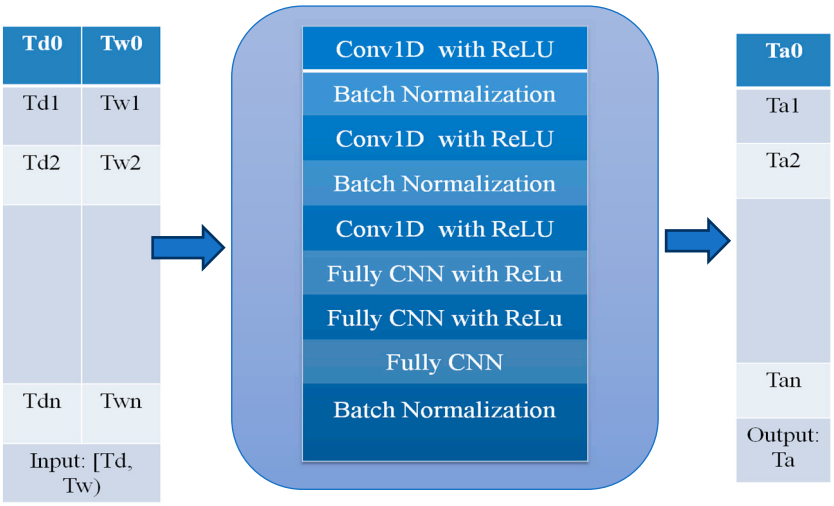


Figure 2. CNN model in Proposed Hybrid Model.

A sigmoid function decreases the output response to a frequency comparable to 0 or 1 [25]. Through block-by-block inspection, CNN gathers the related data for T_w and T_c and concentrates on regional content. We consider a responsive MEC task sequence of events in which the task weight T_w is changeable, and the task workloads Td can modify individually to reduce energy consumption.

In the CNN model, we use batch normalisation (BN) to resolve internal correlation coefficient shift patterns in feature diagrams to avoid overfitting the current model by correcting gradient movement and enhancing network generalisation. Table 2 shows the parameters of the CNN model.

Table 2. Parameters of CNN Model in Proposed Hybrid Model.

Layer Used	Activation Function (AF)	Size	Batch Normalisation
Fully CNN-1	ReLu AF	21	NA
Fully CNN-2	ReLu AF	64	NA
Fully CNN-3	Sigmoid AF	10	10
Conv1D	ReLu AF	16	16
Conv2D	ReLu AF	16	16
Conv3D	ReLu AF	3	NA

3.2.1. CNN-Based Cluster Head Formation

In the proposed hybrid model, the CNN model utilises forward pass and backward phases for weight distribution, which further helps cluster head formation.

Forward Pass

The CNN model predicts all the possible outcomes after receiving the input data (xi) and weight training into the input layer. Equation (1) shows how to determine the net input. Net input ($NT_{input-data}$) depends on the weight parameter (w_{ij}). Equations (2) and (3) show the net input calculation for Layers 1 and 2. In the below equation, x represents the input data, and w represents the weight.

$$NT_{inputdata} = \sum(w_{ij} * x_{ij}) \text{ i and j} > 0 \quad (1)$$

$$NT_{inputdata1} = (w_{11} * E_{Nij}) + (w_{33} * ND_{Nij} + w_{55} * M_{Nij}) + (w_{77} * PD_{Nij}) \quad (2)$$

$$NT_{inputdata2} = (w_{22} * E_{Nij}) + (w_{44} * ND_{Nij}) + (w_{66} * M_{Nij}) + (w_{88} * PD_{Nij}) \quad (3)$$

The inputs are squashed by applying the logistic function. It generates the new output represented in Equations (4) and (5).

$$out_{output1} = \left(\frac{1}{1 + e^{-NT_{input1}}} \right) \quad (4)$$

$$out_{output2} = \left(\frac{1}{1 + e^{-NT_{input2}}} \right) \quad (5)$$

The output from the neurons in the hidden layers is utilised as new input variables in a subsequent iteration of this procedure for better products for the CNN layer.

Determining the Total Error

A mean squared error operation function is applied to determine the error for each output variable, and based on these results; a total cumulative error is measured. In Equation (6), the mean squared error formula is represented [26].

$$En_{total} = \sum \frac{1}{2} (\text{target\_value} - \text{output\_value})^2 \quad (6)$$

The sum of the measured errors is represented in Equation (7); the total error for this CNN is as follows:

$$En_{total} = (En_{output1} + En_{output2}) \quad (7)$$

where:

$$En_{output1} = \frac{1}{2} (\text{target\_value}_{o1} - out_{output1})^2$$

and

$$En_{output2} = \frac{1}{2} (\text{target\_value}_{o2} - out_{output2})^2$$

Calculating the values of $E_{output1}$ and $E_{output2}$ in Equation (8)

$$En_{total} = \frac{1}{2}\left(target\_value_{o1} - out_{output1}\right)^2 + \frac{1}{2}\left(target\_value_{o2} - out_{output2}\right)^2 \quad (8)$$

Backward Pass

The backpropagation (BP) method maintains the network weight information and ensures total performance. A backpropagation method minimises the error rate across each output unit and the network if the absolute error exceeds the desired value. The BP algorithm takes the derivative of the inner function product of En_{total} concerning the specified weights [27].

Consider the weight training (W_1, W_2, W_3, and W_4). Using the convolutional layers from Equation (9) as the chain rule's application location:

$$\frac{\partial En_{total}}{\partial w_9} = \frac{\partial En_{total}}{\partial En_{total}} * \frac{\partial out_{output1}}{\partial net_{input1}} * \frac{\partial net_{input1}}{\partial w_9} \quad (9)$$

After repeating the above equation for weight w_{10}, the hidden layer is described in Equation (10):

$$\frac{\partial En_{total}}{\partial w_n} = \frac{\partial En_{total}}{\partial En_{total}} * \frac{\partial out_{output1}}{\partial net_{input1}} * \frac{\partial net_{input1}}{\partial w_n} \quad (10)$$

After repeating the similar process for additional weight training at a hidden layer in which n = (1, 2, 3, . . . , 8), now, the total error change for the output is:

$$En_{total} = \frac{1}{2}\left(target_{o1} - out_{output1}\right)^2 + \frac{1}{2}\left(target_{o2} - out_{output2}\right)^2 \quad (11)$$

The output's variation concerning its cumulative estimated input is then calculated below.

$$out_{output1} = \frac{1}{1 + en^{-net_{input1}}} \quad (12)$$

$$out_{output2} = \frac{1}{1 + en^{-net_{input2}}} \quad (13)$$

After partially differentiating Equation (14), $\partial out_{output1}$

$$\frac{\partial En_{total}}{\partial out_{output1}} = -\left(target_{o1} - out_{output1}\right) \quad (14)$$

Partially differentiating Equation (15) concerning ∂net_{input1}

$$\frac{\partial out_{output1}}{\partial net_{input1}} = \left[out_{output1}\left(1 - out_{output1}\right)\right] \quad (15)$$

A variation in the network weights for the cumulative estimated input of output1 can be determined by (16).

$$\frac{\partial net_{input1}}{\partial w_n} = out_{output1} \quad (16)$$

The total error can be determined by putting the values of Equation (11) into (16). We have also calculated the outcome for a fraction $[\frac{\partial E_{total}}{\partial w_n}]$ for each weight category from n = 1 to 10. After that, a new error function is determined to overcome the total error rate. The new weight can be the difference in the outcome of a fraction $[\frac{\partial E_{total}}{\partial w_n}]$ and the actual weight as described in Equation (17).

$$w_n += w_n + \mu * \frac{\partial E_{total}}{\partial w_n} \quad (17)$$

where: Wn—New weights Wn and μ learning rate.

The algorithm's hidden and output layers utilise the ReLu activation function. Neural network models use backpropagation to train individuals and automatically update all the weights in response to input datasets. This helps to determine the possible errors in output and hidden neurons.

Rather than using the traditional gradient descent stochastic method, the interpolation method is employed. It determines the active learning rate, aiding computational efficiency and cutting learning costs. It is deemed that the error can be reduced to an acceptable level. The model has been accurately trained even when the final output (Y) equals the network node predicted values [28].

Figure 3 shows the training time graph (training error vs testing error). In each epoch testing and training, data points are represented by a curve that shows the model's inconsistency (one epoch = one cross on the entire dataset). A testing error demonstrates the model's resilience towards the extracted features. The model's loss results significantly decrease for the first 50 epochs; this is encouraging. Ultimately, the medium-sized dataset causes some training and validation curve fluctuations.

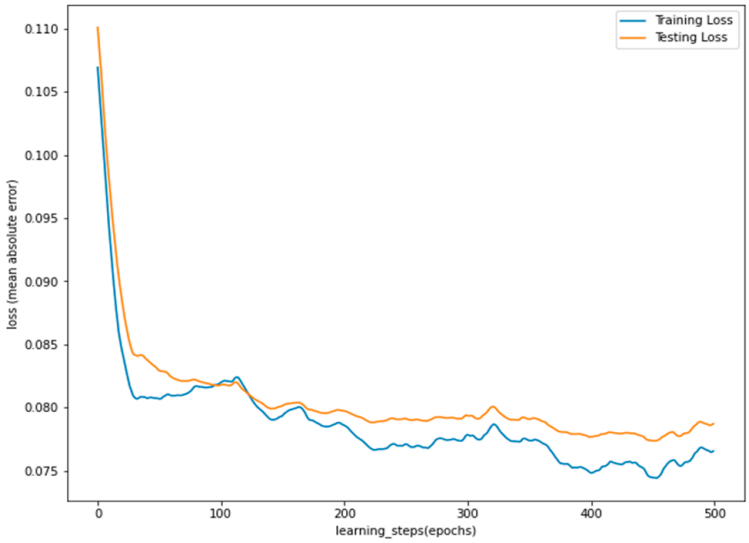


Figure 3. The outcome of the Learning curve.

However, the prototype is outstanding at forecasting future undiscovered samples because the experiment loss is nearer to the training loss. In the first 100 epochs of its 500 epochs of training, the CNN learned very quickly; however, as it neared the end of its activity, the slope began to flatten. The model has achieved better statistics in the simulation, but noise signals can affect the performance. As a result, testing loss outcomes are more significant than training loss development.

The simulation graph clearly shows that after the training phase, the proposed model more precisely forecasted the scores within 0.085. The proposed model generated a rating between 0 and 1 for each sample. Nodes with the best precision and lowest error were selected as cluster head terminals. In Figure 4, this is represented by the "black star".

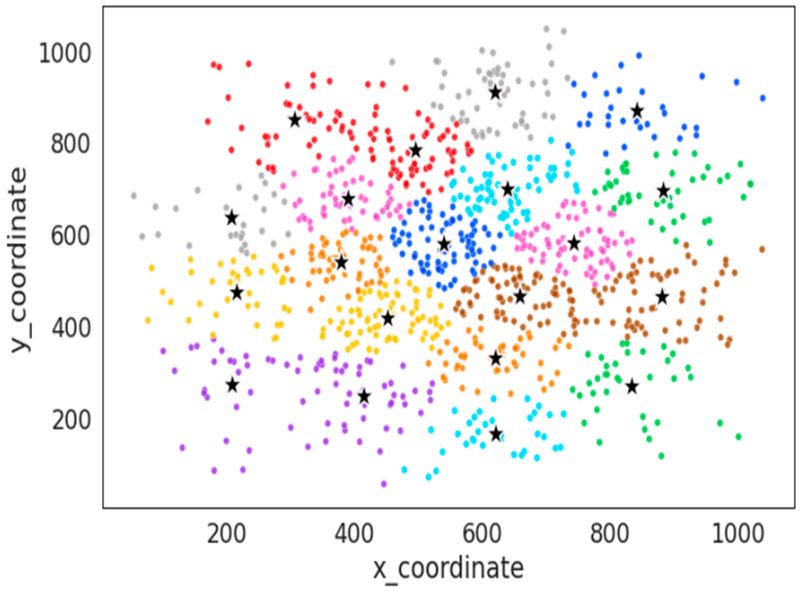


Figure 4. Representation of cluster head by black stars.

3.3. Modified K-Means in Cluster Formation Procedure

We modified the proposed model's existing k-means clustering method [29] for cluster head formation. It is a vector quantisation approach and divides 'n' observations data into 'k' groups. Each observation is closely related to a particular cluster.

In this algorithm, the variable 'k' refers to a dataset's number of nodes/ clusters. All the communication signals are placed into cluster centres assuming the k value. Equation (18) shows the formula for the "n-dimensional centroid point" (ND-CP) within k n-dimensional space:

$$\mathrm{NDCP}(X_{d1},\ X_{d2},\ Xd_3 \ldots\ldots X_{dn}) = \left(\frac{\sum_{i=1}^{n} X_{d1^{st}}}{k1}, \ldots\ldots\ldots,\ \frac{\sum_{i=1}^{n} X_{dn^{th}}}{kn}\right) \tag{18}$$

After this step, the next node's distance toward the cluster centre's coordinates is estimated. Once the model's training is completed, the Euclidean distance is calculated towards the network's x and y coordinates (19).

$$d(i,j) = \sqrt{\left|X_{i_1} - X_{j_1}\right|_2 + \left|X_{i_2} - X_{j_2}\right|_2 + \left|X_{i_3} - X_{j_3}\right|_2 + \ldots + \left|X_{i_n} - X_{j_n}\right|^2} \tag{19}$$

The node point with the lowest distance towards the cluster centroid point is directly merged in the calculated new cluster. This process is iteratively repeated to check each cluster. After the end of the process, a new cluster is formulated. We also utilise the popular elbow analysis technique to recognise all the best clusters (Ck).

Figure 5 shows that the graph starts to flatten knowingly when the number of clusters (k) is 10 and 20; at this point, the graph looks like an elbow. It means the optimal number of clusters can be taken as any value from 10 to 20 for a given dataset under which k-means perform well.

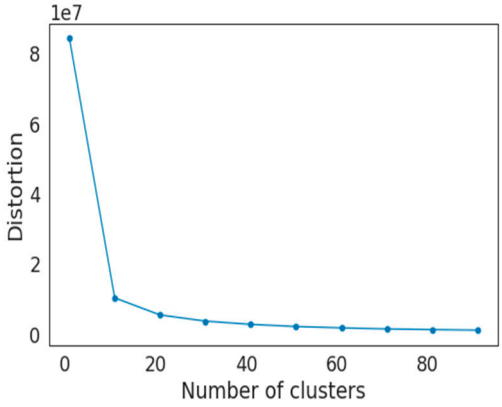


Figure 5. Elbow analysis.

A graph was plotted among the number of cluster nodes and distortion results, as shown in Figure 6. Equation (20) shows the formula to determine the distortion value.

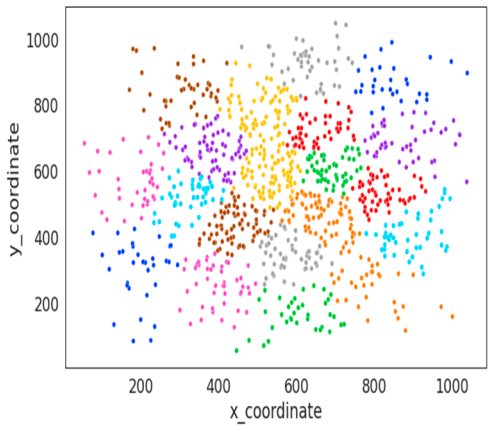


Figure 6. Number of clusters.

The variable Ck shows the possible clusters in p_x (number of points), and variable D_x shows the sum of distances among cluster points for cluster x. The number of groups formed is k. Furthermore, the most efficient node is selected as cluster head within each cluster by using a CNN based on four features: node degree value (NDN), node speed (NS), energy consumption (ECN), and packet drop (PDN).

$$N_{ck} = \sum_{x=1}^{ck} \frac{1}{p_x} D_x \tag{20}$$

3.4. Dataset Description

This research utilises an online network dataset generated on a network simulator. The dataset contains 16 attributes, including node number, x-coordinate, y-coordinate, no. of the packet received, no. of a packet sent, no. of the packet forwarded, no. of packet drop, no. of neighbours, initial energy (constant), remaining energy, node speed, pause time, energy consumption, simulation time (constant), transmission range (constant), and optimal node reliability factor [30] as the target value.

In NS simulator 2.35, the dataset scenario was generated. An experimental setup with 1000 endpoints was positioned inside a terrain area of (1100 × 1100) meters [31]. A random mobility way-point model was used to set up each node, giving it a maximum speed range of 0 to 35 m/s, a data transmitting range of 300 m and a preliminary energy capacity of 300 joules. The data transmission packet size was 512 bytes, and a constant bit

rate-based UDP traffic was used to produce the data traffic pattern. On this basis, various performance features have been recorded for each node during the simulation. The target value represents the node reliability factor calculated during the simulation.

The dataset used in the analysis includes 1034 data samples of selected 11 features, from which 70% of samples were considered for training and 30% for testing.

3.5. Data Preprocessing

The following steps have been defined under data preprocessing [32].

3.5.1. Statistical Analysis and Visualisation of Data

Tables 3 and 4 represent the statistical data analysis in quantile and descriptive statistics, respectively. They are about analysing the data and variables to produce meaningful information.

Table 3. Quantile Statistics.

Attribute	Min.–Max. Value	5th Percentile	Q1	Median	Q3	95th Percentile	Range	Interquartile Range
X-Coordinate	54–1040	218.95	413	569	713.25	903.05	986	300.25
Y-Coordinate	53–1046	198.9	406.75	551	700.25	902.05	993	293.5
Packet Received	150–349	158	198	249	302	339	199	104
Packet Sent	50–199	58.95	90	126	164	192	149	74
Packet Forwarded	150–199	152	163	175	186	197	49	23
Packet Drop	0–149	8	35	72	110	140	149	75
No. of Neighbours	1–9	1	3	5	7	9	8	4
Remaining Energy	80.00–99.98	80.92	85.14	90.37	95.34	99.09	19.99	10.21
Node Speed	1.01–24.98	2.31	6.85	12.80	18.77	23.77	23.97	11.93
Energy Consumption	0.02–19.98	0.92	4.66	9.64	14.87	19.09	19.98	10.21
The Optimal Node Reliability Factor	0.06–1	0.17	0.32	0.51	0.71	0.89	0.93	0.39

Table 4. Descriptive Statistics.

Attribute	Standard Deviation	Coefficient of Variation	Kurtosis	Mean	MAD	Skewness	Sum	Variance	Memory Size
X-Coordinate	205.51	0.36	−0.604	564	150	−0.027	564,329	42,235.63	15.6 KB
Y-Coordinate	205.41	0.37	−0.566	552.42	146	−0.024	552,420	42,194.19	15.6 KB
Packet Received	58.74	0.23	−1.241	249.43	52	−0.032	249,431	3451.50	15.6 KB
Packet Sent	42.92	0.34	−1.204	125.95	37	−0.031	125,952	1842.22	15.6 KB
Packet Forwarded	14.33	0.082	−1.161	174.77	12	−0.054	174,770	205.63	15.6 KB
Packet Drop	43.10	0.59	−1.21	72.851	38	0.040	72,851	1858.37	15.6 KB
No. of Neighbours	2.56	0.511	−1.21	5.01	2	0.011	5010	6.578	15.6 KB
Remaining Energy	5.80	0.064	−1.182	90.23	5.07	−0.098	90,236.00	33.679	15.6 KB
Node Speed	6.84	0.53	−1.197	12.87	5.97	0.033	12,875.68	46.877	15.6 KB
Energy Consumption	5.803	0.59	−1.182	9.76	5.07	0.098	9763.99	33.67	15.6 KB
The Optimal Node Reliability Factor	0.2302	0.438	−1.091	0.525	0.195	0.033	525.369	0.0530	15.6 KB

Quantile statistics refers to dividing a probability distribution into areas of equal probability (four equal parts); it includes minimum and maximum value, 5th Percentile, Q1, median, Q3, 95th percentile, ranges, and interquartile range for variables.

Descriptive statistics are also vital for analysing the data when raw data is challenging to visualise.

Moreover, it summarises data meaningfully and shows a more accessible and straightforward interpretation. It includes standard deviation, coefficient of variation, kurtosis, mean, median absolute deviation (MAD), skewness, sum, variance, and memory size. These metrics help to quickly understand and visualise the data during preprocessing [33].

The histogram and box plot of each feature is shown in Figures 7 and 8, respectively, to visualise essential elements from the dataset. These provide a quantitative understanding to summarise the distribution of variables and also help to identify data density, patterns, and outlier points in datasets. In histogram plots, data samples are displayed in a bar chart where the x-axis generally gives intervals or discrete bins for the observations. The y-axis shows the dataset's frequency or count of adherence [33].

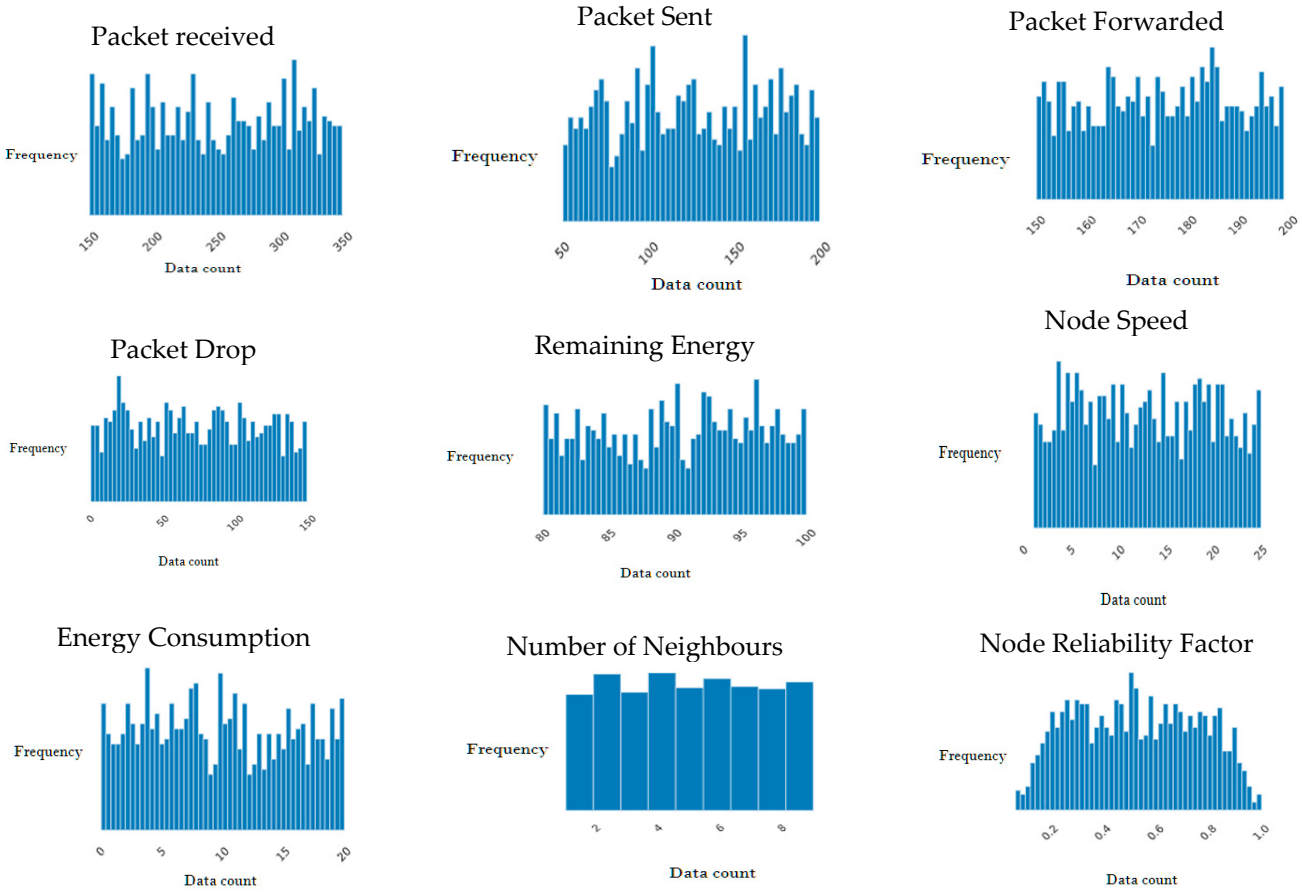


Figure 7. Histogram of features.

Figure 8 represents the blue box for the middle 50% of the data, within that black line for the median, the end lines for the whiskers that summarise the range of sensible data, and finally, dots for the possible outliers.

Another representation is that the box plot also helps to observe the skewness, spread, and outlier points in which the x-axis is defined to represent the data sample, and the y-axis shows the observation values. For each attribute, single boxplots have been drawn that summarise the middle 50% of the data and create a box that starts with the observation at the 25th Percentile (Q1) and ends with the 75th Percentile (Q3), known as the interquartile range. The 50th Percentile (Q2) shows the median represented by a line. Whisker lines extend from both ends of the box, demonstrating the expected range of sensible data in the distribution (minimum and maximum value of data) [34].

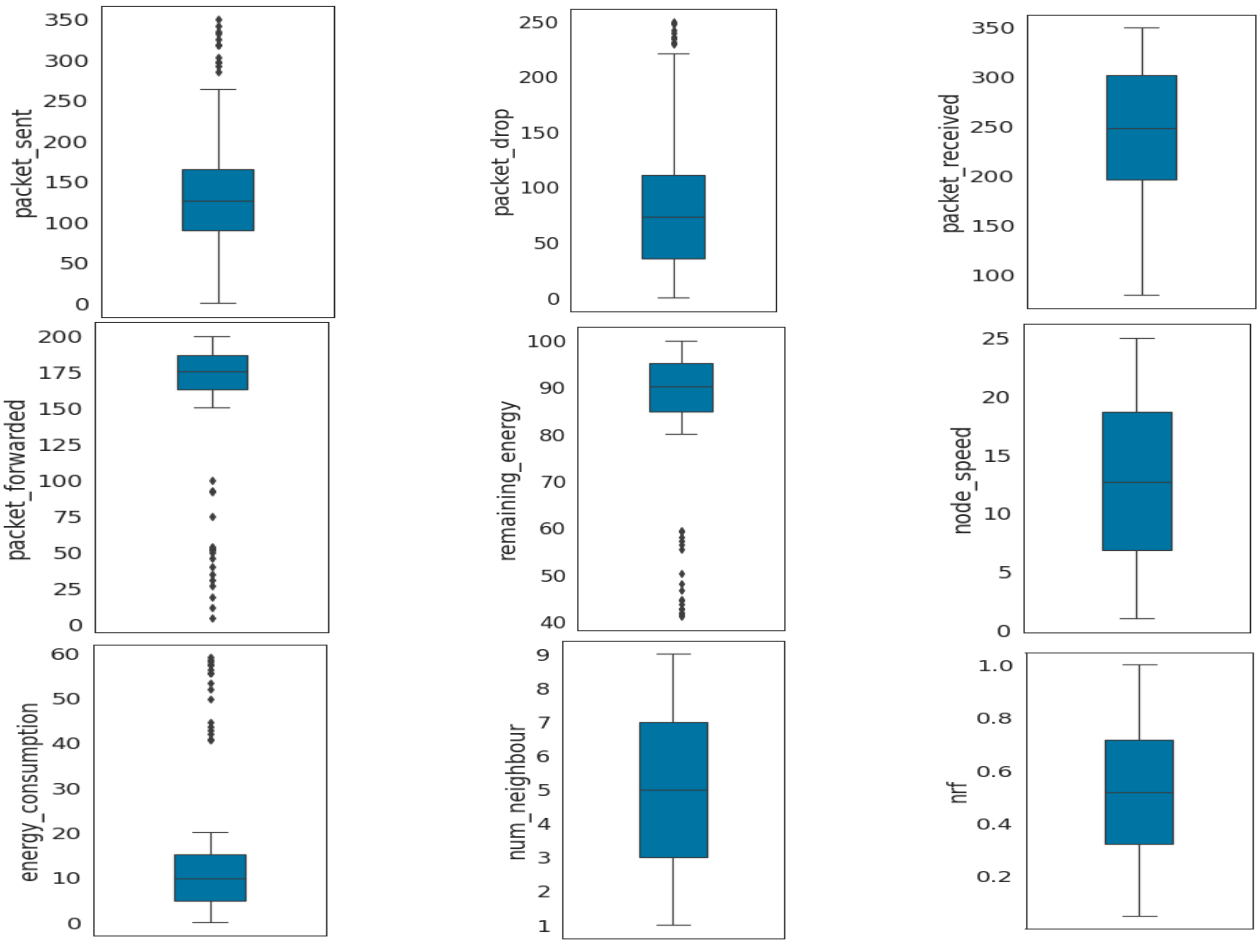


Figure 8. Boxplot representation of features.

Observations outside the whiskers might be outliers and are drawn with small circles. Mathematically, this is represented as:

$$InterQuartileRange(IQR) = Q_3 - Q_1 \tag{21}$$

The expected range is $[(Q_1 - 1.5 \times IQR),\ (Q_3 + 1.5 \times IQR\ ]$; any data point outside the given range will be considered an outlier.

3.5.2. Normalisation of Data and Feature Selection

The previous sections discussed the comprehensive details of the dataset. This section applies the preprocessing data feature. This phase first involves data cleaning to remove null records and outliers. It utilises a z-normalization process to standardise feature values by putting them on the same scale. This phase eliminates the missing value and noise from the dataset and normalises the dataset. It helps the model training process and also improves the overall accuracy. Mathematically, Z-score normalisation represents Equation (22)

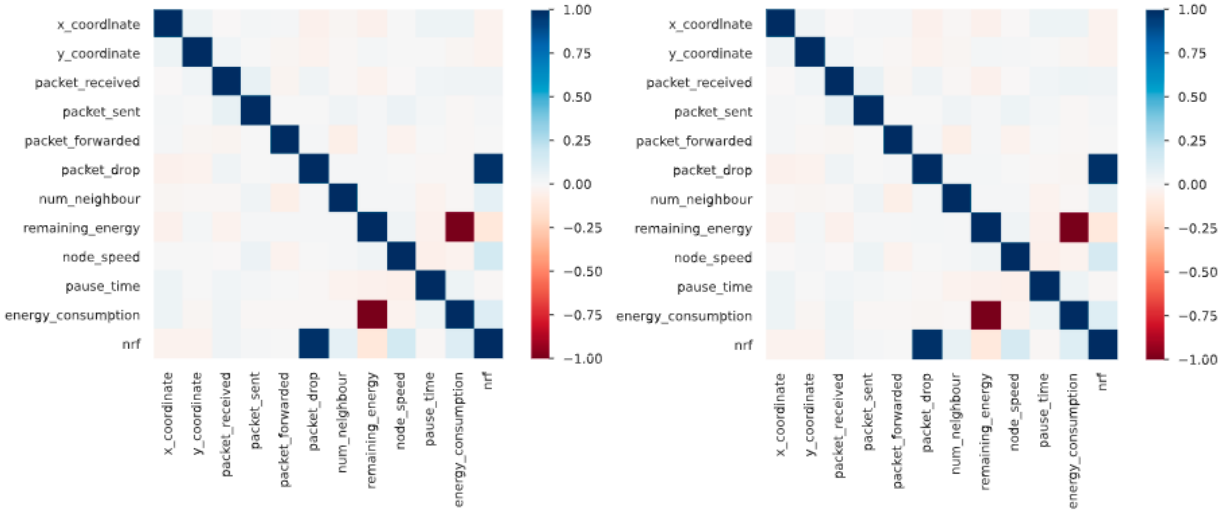
$$Z - \text{score} = (X_i - \mu)\frac{1}{\sigma} \tag{22}$$

Xi, μ, and σ represent the original mean and standard deviation samples. Equation (23) represents the σ value.

$$\sigma = \sqrt{\frac{\sum (X_i - \mu)^2}{\text{no. ofsample}}} \tag{23}$$

Figure 9 shows the Pearson correlation and Spearman correlation matrix. This help to quantify the relationship between features and measure the strength of how they strongly correlate. Mathematical formulas are shown in Equation (24) for pairs of random elements (X, Y).

$$\text{Pearson\_correlation\_coefficien}\ (X,\ Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{24}$$



(**a**) (**b**)

Figure 9. Pearson (**a**) and spearman (**b**) correlation matrix.

When calculating covariance cov(X, Y) to determine the direction of the relationship between features by capturing the variance between X and Y, it may be given a positive or negative result. Here, σ_X and σ_Y show the standard deviation of X and Y (25). Finally, the result of $\rho(X,\ Y)$ always represents a value between -1 and 1.

$$\text{cov}(X,\ Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x) \times \left(y_i - \mu_y\right) \tag{25}$$

where a spearman correlation coefficient $\rho(rg_X, rg_Y) = r_S = \frac{\text{cov}(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_y}}$ and, $d_i = rg(x_i) - rg(y_i)$ and, $r_S = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$.

Here initially, Xi and Yi converted into ranks variables rg_X and rg_Y, respectively, and calculate $\rho(X,\ Y)$ between rank variables. $\text{cov}(rg_X, rg_Y)$ is covariance, and $\sigma_{rg_X}\sigma_{rg_y}$ are the standard deviation of the rank variables. The correlation coefficient matrix gives evidence about correlated features. These are measured on a scale of -1 to 1. In Figure 7, the feature with the value -1 (dark maroon) and 1 (dark blue) or nearly that means it has highly corrected to each other. This analysis identifies four features: energy consumption, a neighbour of node, node speed, and packet drop, which are positively correlated to the target value (optimal node reliability factor).

4. Experimental Results and Discussions

This section covers experimental analysis and discussion. The proposed E-CFSA model and the existing AB-SEP method [2] and WCA [3] are compared based on performance measuring parameters.

4.1. Network Setup

This research utilised a multi-hop network connection with varying homogeneous sensor nodes. Experimental modelling was performed over NS-2 and Python [35]. Many grids were used to split the rectangle region under deep consideration. This is consistent with the idea that the implemented network is rectangular and supported by most existing studies.

A rectangular pattern of sensor networks has been used to send and receive packets. The BS was situated either toward the middle or near any of the connected edges in the simulation. Constant power communication is used as a power transmission process [36].

The proposed hybrid model considers the partitioning procedure in a partial loading method which determines the expense for each potential partitioning and loading strategy and chooses the optimum solution. The attributes of key simulation attributes are displayed in Table 5.

Table 5. Simulation parameters used.

Simulation Parameters	Values
Nodes	Sim 1: 20 to 100 and Sim 2: 100 to 1000 nodes
Total simulation duration	200 s
Terrain	Sim 1: 500×500 m and Sim 2: 1000 × 1000 m
Mobility model	Random way-point model
Node speed	0 m/s to 25m/s (random way)
Primary node energy	0 to 200, Joule (random way)
Data traffic	CBR with UDP
Number of CBR and load	CBR: 10 pairs and packet size: 512 bytes
Communication channel	Wireless
Location of a base station	Node: 20, 40, . . . , 100
Load partitioning	Partial loading method

4.2. Performance Measuring Parameters

The specifications that evaluate the performance of both the traditional and the proposed E-CFSA are presented in this subsection. Experiments were conducted to compare the E-CFSA with the conventional approach to determine the cluster head's routing overhead, throughput, packet delivery ratio, and stability period [37].

- Packet delivery ratio (PDR): The PDR is the data packets the destinations receive to those the sources generate. Mathematically, it can be defined by Equation (26). S1 is the number of packets sent, and S2 is the number of packets received for nodes.

$$\text{PDR} = \left[\frac{\text{S1}}{\text{S2}} * 100 \right] \tag{26}$$

- Throughput (Th): It is defined as the fraction of the sum of delivered packets (from the source) and the total simulation time by Equation (27).

$$\text{Th} = \left[\frac{\text{Number of received packets}}{\text{Simulation time}} \right] \tag{27}$$

- Routing or network overhead (RO): It is defined as the number of control and routing packets required for communication in the network, as described in Equation (28).

$$\text{RO} = \left[\frac{\text{number of routing packets}}{\text{packets received}} \right] \tag{28}$$

- Cluster head stability time (CHST): It is defined as the total period for which a network node works as a cluster head. The average of that period is known as the average stability time.

- Energy consumption (EC): The cumulative energy the system uses for data transformation, communication, and confirmation, as described in Equation (29).

$$EC = \frac{\text{Energy used in communication}}{\text{Total Energy}} \quad (29)$$

4.3. Simulation Results and Discussion

As described in Table 6, the proposed E-CFS algorithm and existing methods AB-SEP and WCA were evaluated in different scenarios with different parameters, i.e., the number of grids and clusters and the number of nodes. The number of grids ranges depending on the start conditions, varying from six to one hundred. Moreover, the node sample size ranges depending on the network region size. It differs from 20 to 100 (in Scenario-1) and 200 to 1000 (in Scenario-2).

Table 6. Experimental scenarios and specifications.

Scenario	Terrain	Grid Size	Number of Rounds	Number of Nodes
Scenario-1	500 × 500	4 × 4	100–2000	20 to 100
Scenario-2	1000 × 1000	10 × 10	100–2000	200 to 1000

4.3.1. Scenario One

The first scenario was implemented with 20 to 100 nodes, a grid size of (4 × 4), and a terrain of (500 × 500) meters. Experiments were performed to measure the performance of the proposed E-CFSA hybrid deep learning model. Moreover, a comparative analysis was performed with existing AB-SEP and WCA methods [38]. Table 7 shows the simulation results (impact of network size), and Table 8 shows the results (effect of node speed).

Table 7. Simulation results (Impact of Network Size).

Nodes	PDR (%)			Throughput (in kbps)			Routing Overhead			Average Stability Time (in a sec)			Energy Consumption (in Jules) for CHs and Non-CHS All the Nodes		
Frequency	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA
20	83.42	82.45	75.87	126	110	98	0.42	0.41	0.62	22	15	10	0.0134	0.027	0.041
40	87.13	84.16	77.49	157	124	110	0.51	0.61	0.81	50	40	30	0.0141	0.025	0.041
60	88.91	85.20	69.92	176	135	138	0.62	0.71	0.91	75	60	40	0.0145	0.312	0.054
80	91.95	87.28	69.88	220	180	149	0.82	0.91	0.98	88	75	55	0.0152	0.341	0.059
100	87.81	86.70	74.17	255	210	172	0.96	1.0	1.23	110	95	65	0.0187	0.387	0.060

Table 8. Simulation results (impact of Node speed).

Node Speed (m/s)	PDR (%)			Throughput (in kbps)			Packet Loss Rate (%)			Average Stability Time of CH's (in a sec)		
Frequency	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA
5	94.83	89.52	80.89	115.83	100.65	95.81	12.57	16.12	20.11	40.37	35.74	25.78
10	93.72	90.76	80.78	113.32	98.74	93.92	17.45	19.24	25.27	35.74	31.56	24.85
15	90.21	89.80	77.75	102.98	99.48	92.18	16.32	19.21	29.87	28.88	27.37	20.96
20	89.76	87.90	70.96	100.48	98.76	85.28	22.12	24.77	29.89	25.38	24.89	21.56
25	88.17	86.45	71.47	98.74	96.64	78.34	25.34	26.98	34.55	22.95	22.55	15.77

The simulation results based on the network size and node speed are presented in Figures 10 and 11 for the proposed E-CFSA and existing AB-SEP and WCA. In Figure 8, when the number of nodes varies from 20 to 100, the PDR % of the proposed model is best for node 80; similar to 100 nodes, the throughput is 255, and similar routing overhead,

average stability time (in a sec), and energy consumption (for all CHs and non-CHS nodes) results for the proposed method are best as compared to the existing WCA and AB-SEP.

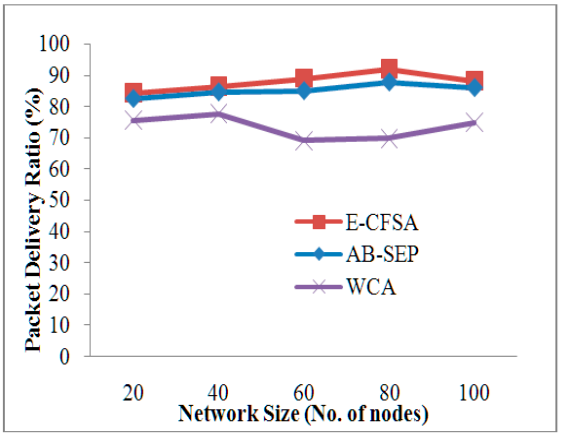


Figure 10. Graph PDR Vs. Network size.

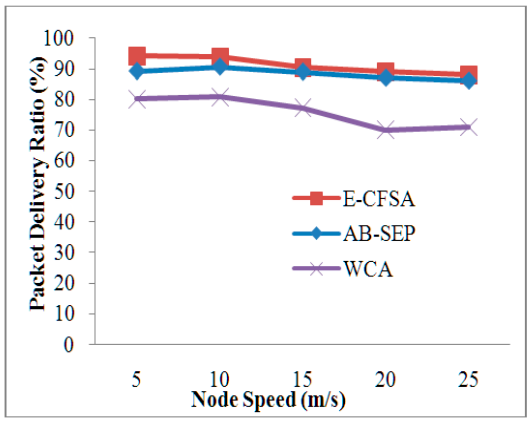


Figure 11. Graph PDR Vs. Node speed.

Similar to Figure 11, simulation results for the impact of node speed clearly show that the proposed method achieves better PDR, throughput, packet loss rate, and average stability time of CHs compared to the existing AB-SEP and WCA methods [39].

Figures 12 and 13 show the throughput simulation results for the proposed E-CFS and the existing WCA and AB-SEP methods with varying node speeds and network node count. The experimental findings demonstrate the higher data rate of E-CFSA speeds during both the trials and variability.

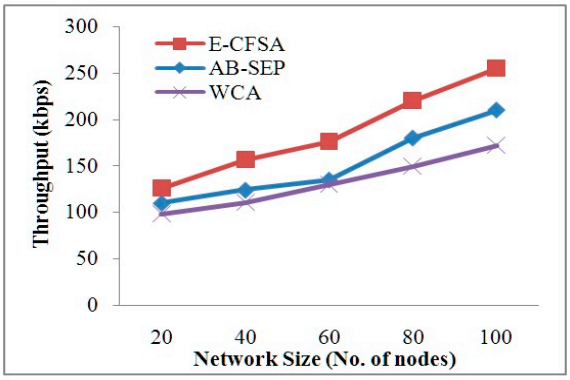


Figure 12. Graph throughput Vs. Network.

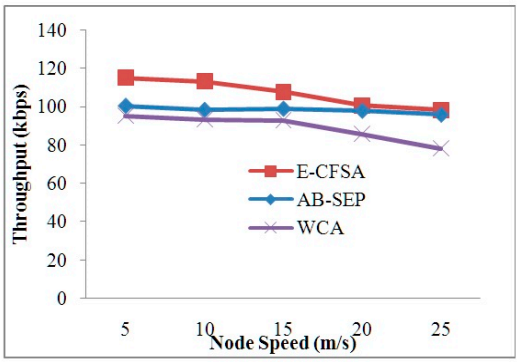


Figure 13. Graph throughput Vs. Node speed.

In this simulation, the network's node speed varied from 5 m/s to 25 m/s, and the number of nodes varied from 20 to 100. These outcomes also show that WCA chose the shortest route but did not consider the node reliability factor. As a result, there is a regular variation in network sizes during routing, which reduces throughput. The experimental results of routing overhead with changing network size (number of nodes) and packet loss ratio with changing node speed are shown in Figures 14 and 15, respectively. These two experiments are critical in evaluating and contrasting the efficiency of the proposed E-CFSA with the existing WCA and AB-SEP methods. The proposed method reduces routing overhead and packet drop ratio.

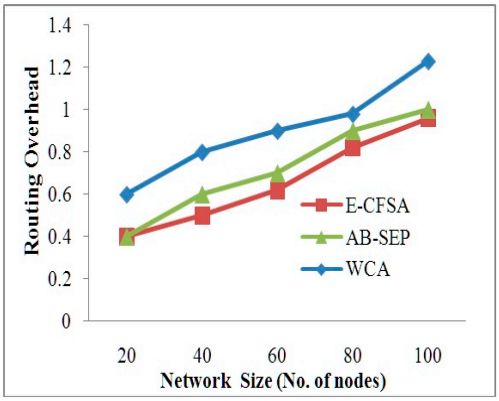


Figure 14. Graph Routing overhead Vs. Network size.

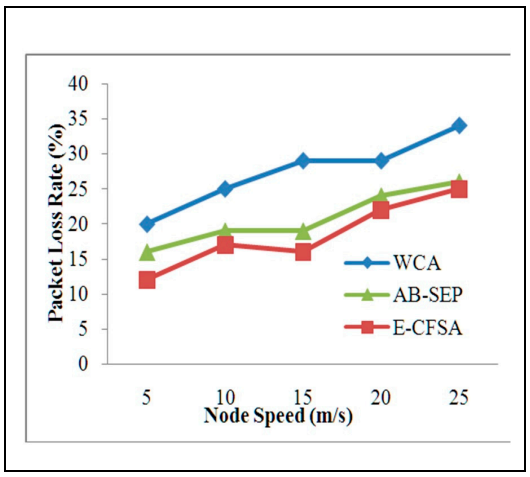


Figure 15. Graph Packet loss ratio Vs. Node speed.

Figures 16 and 17 show the comparison results calculated for a lifetime for the proposed and existing methods. These results were calculated with dynamic network size and network node speed. The results of the experiments clearly show that the proposed E-CFSA achieve lower stability times and keeps all variations in the network size and node speed. The innovative collaborations enable the proposed E-CFSA to outperform traditional approaches. The above experimental findings also suggest that k-means assists the proposed E-CFSA in cluster head selection, enabling this outclass over existing methods.

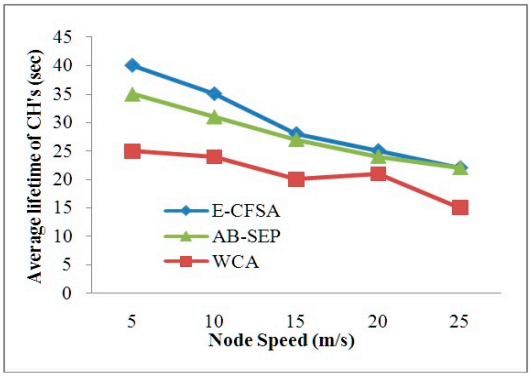


Figure 16. Graph Average Stability Time Vs. Node speed.

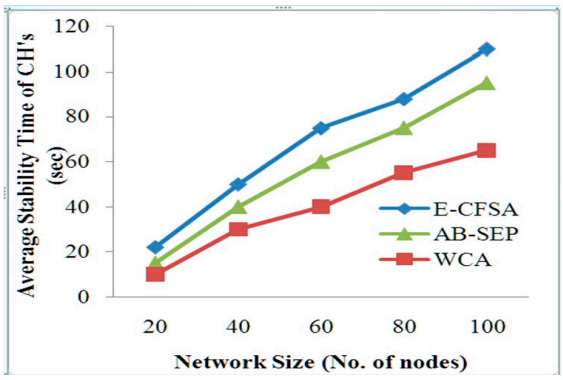


Figure 17. Graph Average Stability Time of CHs Vs. Number of nodes.

4.3.2. Scenario-Two

The second scenario was implemented with 200 to 1000 nodes, a grid size of (10 × 10), several rounds of 100 to 2000, and a terrain of (1000 × 1000) meters. Experiments were performed to measure the performance of the proposed E-CFSA. Then, a comparison was made with the current state-of-the-art methods. Table 9 shows the results of the effect of the network size.

Figures 18–21 show the simulation results of scenario two for the proposed E-CFSA, the existing WCA, and AB-SEP with variations in the number of nodes and node speed in the network. The proposed method has a better PDR % of 98.99 % for 20 nodes, which is best compared to the WCA and AB-SEP methods. Similarly, the proposed method achieves 94.89 % throughput for a node speed of 5 m/s, which is the best result. Similar to Figures 18 and 19, the proposed method performs a network lifetime of 107 s and an average stability time of CH of 97 s, which is best compared to existing WCA and AB-SEP methods.

Table 9. Simulation results (Impact of Network Size).

Number of Nodes	Packet Delivery Ratio (%)			Throughput (kbps)			Routing Overhead			Average Stability Time of CH's (sec)			Energy Consumption (J) in Both CHs and Non-CHS		
Frequency	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA	E-CFSA	AB-SEP	WCA
200	86.91	86.98	78.18	197	127	107	0.518	0.498	0.688	44	35	25	0.0174	0.031	0.047
400	88.41	86.76	79.67	187	136	122	0.491	0.667	0.892	75	38	47	0.0154	0.035	0.049
600	89.91	87.84	78.87	196	155	147	0.678	0.787	0.974	85	54	60	0.0165	0.378	0.055
800	91.65	85.84	77.75	235	194	166	0.858	0.934	0.968	90	89	78	0.0157	0.308	0.057
1000	92.88	86.98	75.34	278	217	184	0.963	1.1	1.1	122	98	89	0.0178	0.344	0.069

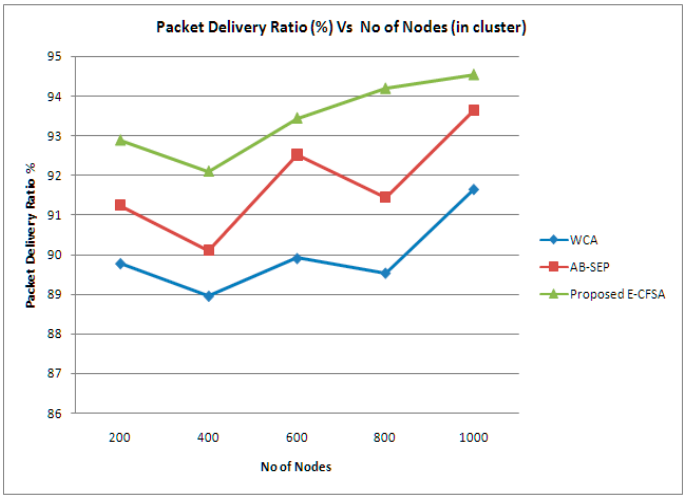


Figure 18. Graph PDR Vs. No. of Nodes.

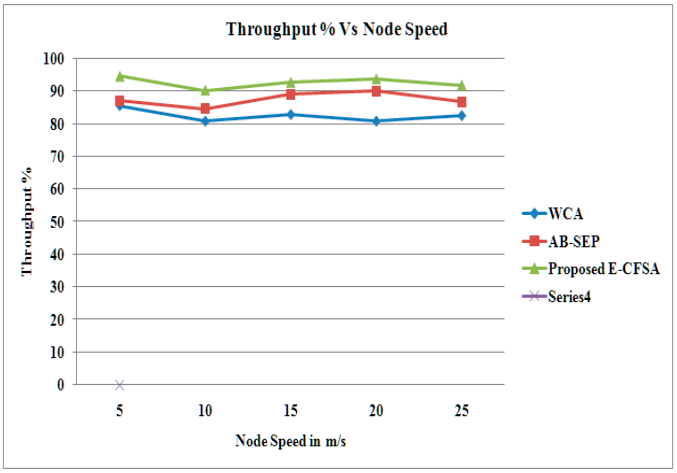


Figure 19. Graph Throughput Vs. No. of Nodes.

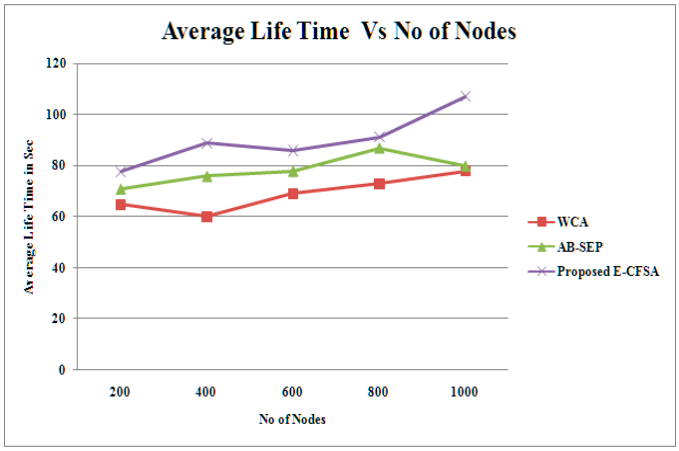


Figure 20. Graph Avg Lifetime Vs. No. of Nodes.

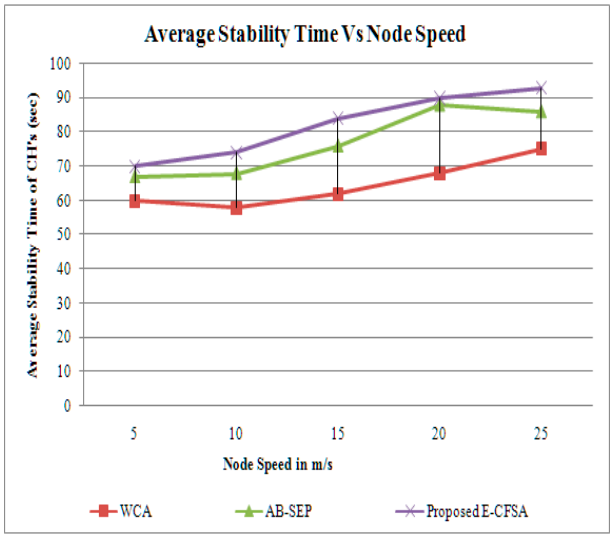


Figure 21. Graph Avg Stability Vs. Node Speed.

5. Conclusions

This research proposes an energy-efficient cluster formation and head selection algorithm (E-CFSA) using a CNN and modified k-mean clustering for a MEC environment. This research develops an efficient way to make clusters with stable cluster heads using machine learning in which nodes form clusters using the k-means algorithm. A CNN was trained to select an efficient cluster head. Data collection has been performed through network simulation to build training and test data, data analytics applied to analyse the data, and feature selection used to select the best model. The trained model predicted scores with an error of $+/-0.075$ on the test dataset. This procedure has reduced the repetitive passage of cluster heads with more extended stability and a lifetime of member nodes in a cluster that is analysed through the cluster head stability time parameter. Finally, performance is examined under the best model regarding overhead, packet delivery ratio, and throughput with the network size and node speed variation. The model has shown better results with less overhead, packet loss rate, and a higher throughput and packet delivery ratio than the existing WCA and AB-SEP methods.

The proposed model can be extended in future work using bioinspired methods, as they offer enticing concepts; selecting cluster nodes can be considered an intelligent and optimisation problem for more complex and heterogeneous networks.

Author Contributions: This research specifies the following individual contributions: Conceptualization, D.B. and U.K.L.; Data curation, P.M. and U.K.L.; Formal analysis, F.D. and U.K.L.; Funding acquisition, O.M. and K.R.; Investigation, F.H.; Methodology, P.S.; Project administration, K.R.; Resources, U.K.L.; Software, F.H.; Supervision, U.K.L. and P.M; Validation, P.S.; Visualization, D.B.; Writing-original draft, D.B. and U.K.L.; Writing-review & editing, D.B. and U.K.L. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank Prince Sattam Bin Abdulaziz University project number (PSAU/2023/R/1444). The authors thank Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R236), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Data Availability Statement: Data will be provided by the authors on demand.

Conflicts of Interest: The authors affirm that they do not have any conflicting priorities related to the research.

Abbreviations

Abbreviations	Details
MEC	Mobile edge computing
CDs	Client devices
E-CFSA	Energy-efficient cluster formation and head selection
CNNs	Convolutional neural networks
MKM	Modified k-mean clustering
IoT	Internet of things
CH	Cluster head
DL	Deep learning
CNN	Convolution neural networks
MES	Mobile edge server
BN	Batch normalisation
WCA	Weighted clustering
AB-SEP	Agent-based secure enhanced performance approach

References

1. Song, S.; Ma, S.; Zhao, J.; Yang, F.; Zhai, L. Cost-efficient multi-service task offloading scheduling for mobile edge computing. *Appl. Intell.* **2022**, *52*, 4028–4040. [CrossRef]
2. Zhou, W.; Chen, L.; Tang, S.; Lai, L.; Xia, J.; Zhou, F.; Fan, L. Offloading strategy with PSO for mobile edge computing based on cache mechanism. *Clust. Comput.* **2022**, *25*, 2389–2401. [CrossRef]
3. Irshad, A.; Chaudhry, S.A.; Ghani, A.; Mallah, G.A.; Bilal, M.; Alzahrani, B.A. A low-cost privacy-preserving user access in mobile edge computing framework. *Comput. Electr. Eng.* **2022**, *98*, 107692. [CrossRef]
4. Zhao, F.; Chen, Y.; Zhang, Y.; Liu, Z.; Chen, X. Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 2154–2165. [CrossRef]
5. Al-Shuwaili, A.; Simeone, O. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 398–401. [CrossRef]
6. Yang, Z.; Bi, S.; Zhang, Y.-J.A. Dynamic offloading and trajectory control for UAV-enabled mobile edge computing system with energy harvesting devices. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 10515–10528. [CrossRef]
7. Han, T.; Zhang, L.; Pirbhulal, S.; Wu, W.; de Albuquerque, V.H.C. A novel cluster head selection technique for edge-computing based IoMT systems. *Comput. Netw.* **2019**, *158*, 114–122. [CrossRef]
8. Wang, T.; Qiu, L.; Sangaiah, A.K.; Xu, G.; Liu, A. Energy-efficient and trustworthy data collection protocol based on mobile fog computing in Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3531–3539. [CrossRef]
9. Lin, Y.; Cavallaro, J.R. Energy-efficient convolutional neural networks via statistical error compensated near threshold computing. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
10. Zhou, Z.; Shojafar, M.; Abawajy, J.; Yin, H.; Lu, H. ECMS: An edge intelligent energy efficient model in mobile edge computing. *IEEE Trans. Green Commun. Netw.* **2021**, *6*, 238–247. [CrossRef]
11. Wang, Q.; Tan, L.T.; Hu, R.Q.; Qian, Y. Hierarchical energy-efficient mobile-edge computing in IoT networks. *IEEE Internet Things J.* **2020**, *7*, 11626–11639. [CrossRef]
12. Tun, Y.K.; Park, Y.M.; Tran, N.H.; Saad, W.; Pandey, S.R.; Hong, C.S. Energy-efficient resource management in UAV-assisted mobile edge computing. *IEEE Commun. Lett.* **2020**, *25*, 249–253. [CrossRef]

13. Cao, X.; Wang, F.; Xu, J.; Zhang, R.; Cui, S. Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet Things J.* **2018**, *6*, 4188–4200. [CrossRef]
14. Wu, G.; Miao, Y.; Zhang, Y.; Barnawi, A. Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading. *Comput. Commun.* **2020**, *150*, 556–562. [CrossRef]
15. Zhang, K.; Mao, Y.; Leng, S.; Zhao, Q.; Li, L.; Peng, X.; Pan, L.; Maharjan, S.; Zhang, Y. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. *IEEE Access* **2016**, *4*, 5896–5907. [CrossRef]
16. Zhang, L.; Lai, S.; Xia, J.; Gao, C.; Fan, D.; Ou, J. Deep reinforcement learning based IRS-assisted mobile edge computing under physical-layer security. *Phys. Commun.* **2022**, *55*, 101896. [CrossRef]
17. Ning, Z.; Huang, J.; Wang, X.; Rodrigues, J.J.P.C.; Guo, L. Mobile edge computing-enabled Internet of vehicles: Toward energy-efficient scheduling. *IEEE Netw.* **2019**, *33*, 198–205. [CrossRef]
18. Ale, L.; Zhang, N.; Fang, X.; Chen, X.; Wu, S.; Li, L. Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 881–892. [CrossRef]
19. Lyu, X.; Tian, H.; Ni, W.; Zhang, Y.; Zhang, P.; Liu, R.P. Energy-efficient admission of delay-sensitive tasks for mobile edge computing. *IEEE Trans. Commun.* **2018**, *66*, 2603–2616. [CrossRef]
20. Guleria, K.; Prasad, D.; Lilhore, U.K.; Simaiya, S. Asynchronous Media Access Control Protocols and Cross Layer Optimizations for Wireless Sensor Networks: An Energy Efficient Perspective. *J. Comput. Theor. Nanosci.* **2020**, *17*, 2531–2538. [CrossRef]
21. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Haq, N.U.; Umar, A.I.; Shuja, J.; Ahmad, Z.; Ben Dhaou, I.; Alsharekh, M.F. LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Clust. Comput.* **2022**, *26*, 99–117. [CrossRef]
22. Lilhore, U.K.; Khalaf, O.I.; Simaiya, S.; Tavera Romero, C.A.; Abdulsahib, G.M.; Kumar, D. A depth-controlled and energy-efficient routing protocol for underwater wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2022**, *18*, 15501329221117118. [CrossRef]
23. Trinh, H.; Calyam, P.; Chemodanov, D.; Yao, S.; Lei, Q.; Gao, F.; Palaniappan, K. Energy-aware mobile edge computing and routing for low-latency visual data processing. *IEEE Trans. Multimed.* **2018**, *20*, 2562–2577. [CrossRef]
24. Mukherjee, M.; Kumar, V.; Lat, A.; Guo, M.; Matam, R.; Lv, Y. Distributed deep learning-based task offloading for UAV-enabled mobile edge computing. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 1208–1212.
25. Simaiya, S.; Lilhore, U.K.; Pandey, H.; Trivedi, N.K.; Anand, A.; Sandhu, J. An Improved Deep Neural Network-Based Predictive Model for Traffic Accident's Severity Prediction. In *Ambient Communications and Computer Systems*; Springer: Singapore, 2022; pp. 181–190.
26. Ali, Z.; Abbas, Z.H.; Abbas, G.; Numani, A.; Bilal, M. Smart computational offloading for mobile edge computing in next-generation Internet of Things networks. *Comput. Netw.* **2021**, *198*, 108356. [CrossRef]
27. Lilhore, U.K.; Simaiya, S.; Kaur, A.; Prasad, D.; Khurana, M.; Verma, D.K.; Hassan, A. Impact of Deep Learning and Machine Learning in Industry 4.0: Impact of Deep Learning. In *Cyber-Physical, IoT, and Autonomous Systems in Industry 4.0*; CRC Press: Boca Raton, FL, USA, 2021; pp. 179–197.
28. Chen, Z.; He, Q.; Liu, L.; Lan, D.; Chung, H.M.; Mao, Z. An artificial intelligence perspective on mobile edge computing. In Proceedings of the 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, 9–11 August 2019; pp. 100–106.
29. Sangaiah, A.K.; Medhane, D.V.; Han, T.; Hossain, M.S.; Muhammad, G. Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics. *IEEE Trans. Ind. Inform.* **2019**, *15*, 4189–4196. [CrossRef]
30. Lilhore, U.K.; Imoize, A.L.; Lee, C.-C.; Simaiya, S.; Pani, S.K.; Goyal, N.; Kumar, A.; Li, C.-T. Enhanced convolutional neural network model for cassava leaf disease identification and classification. *Mathematics* **2022**, *10*, 580. [CrossRef]
31. Kathiroli, P.; Selvadurai, K. Energy efficient cluster head selection using improved Sparrow Search Algorithm in Wireless Sensor Networks. *J. King Saud Univ. -Comput. Inf. Sci.* **2022**, *34*, 8564–8575. [CrossRef]
32. Arbi, A.; Cao, J.; Es-Saiydy, M.; Zarhouni, M.; Zitane, M. Dynamics of delayed cellular neural networks in the Stepanov pseudo almost automorphic space. *Discret. Contin. Dyn. Syst.-S* **2022**, *15*, 3097–3109. [CrossRef]
33. Arbi, A.; Cao, J.; Alsaedi, A. Improved synchronization analysis of competitive neural networks with time-varying delays. *Nonlinear Anal. Model. Control* **2018**, *23*, 82–107. [CrossRef]
34. Guo, Y.; Ge, S.S.; Arbi, A. Stability of traveling waves solutions for nonlinear cellular neural networks with distributed delays. *J. Syst. Sci. Complex.* **2022**, *35*, 18–31. [CrossRef]
35. Abrar, M.; Ajmal, U.; Almohaimeed, Z.M.; Gui, X.; Akram, R.; Masroor, R. Energy efficient UAV-enabled mobile edge computing for IoT devices: A review. *IEEE Access* **2021**, *9*, 127779–127798. [CrossRef]
36. Chen, Y.; Zhang, N.; Zhang, Y.; Chen, X.; Wu, W.; Shen, X.S. Energy efficient dynamic offloading in mobile edge computing for internet of things. *IEEE Trans. Cloud Comput.* **2019**, *9*, 1050–1060. [CrossRef]
37. Zhang, D.-G.; Chen, L.; Zhang, J.; Chen, J.; Zhang, T.; Tang, Y.-M.; Qiu, J.-N. A multi-path routing protocol based on link lifetime and energy consumption prediction for mobile edge computing. *IEEE Access* **2020**, *8*, 69058–69071. [CrossRef]

38. Simaiya, S.; Gautam, V.; Lilhore, U.K.; Garg, A.; Ghosh, P.; Trivedi, N.K.; Anand, A. EEPSA: Energy efficiency priority scheduling algorithm for cloud computing. In Proceedings of the 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 7–9 October 2021; pp. 1064–1069.
39. Liao, L.; Lai, Y.; Yang, F.; Zeng, W. Online computation offloading with double reinforcement learning algorithm in mobile edge computing. *J. Parallel Distrib. Comput.* **2023**, *171*, 28–39. [CrossRef]