International Journal of Computational Intelligence Systems (2024) 17:229
https://doi.org/10.1007/s44196-024-00631-4

RESEARCH ARTICLE

Optimized Tiny Machine Learning and Explainable AI for Trustable and Energy-Efficient Fog-Enabled Healthcare Decision Support System

R. Arthi[1] · **S. Krishnaveni**[2]

Received: 28 May 2024 / Accepted: 13 August 2024

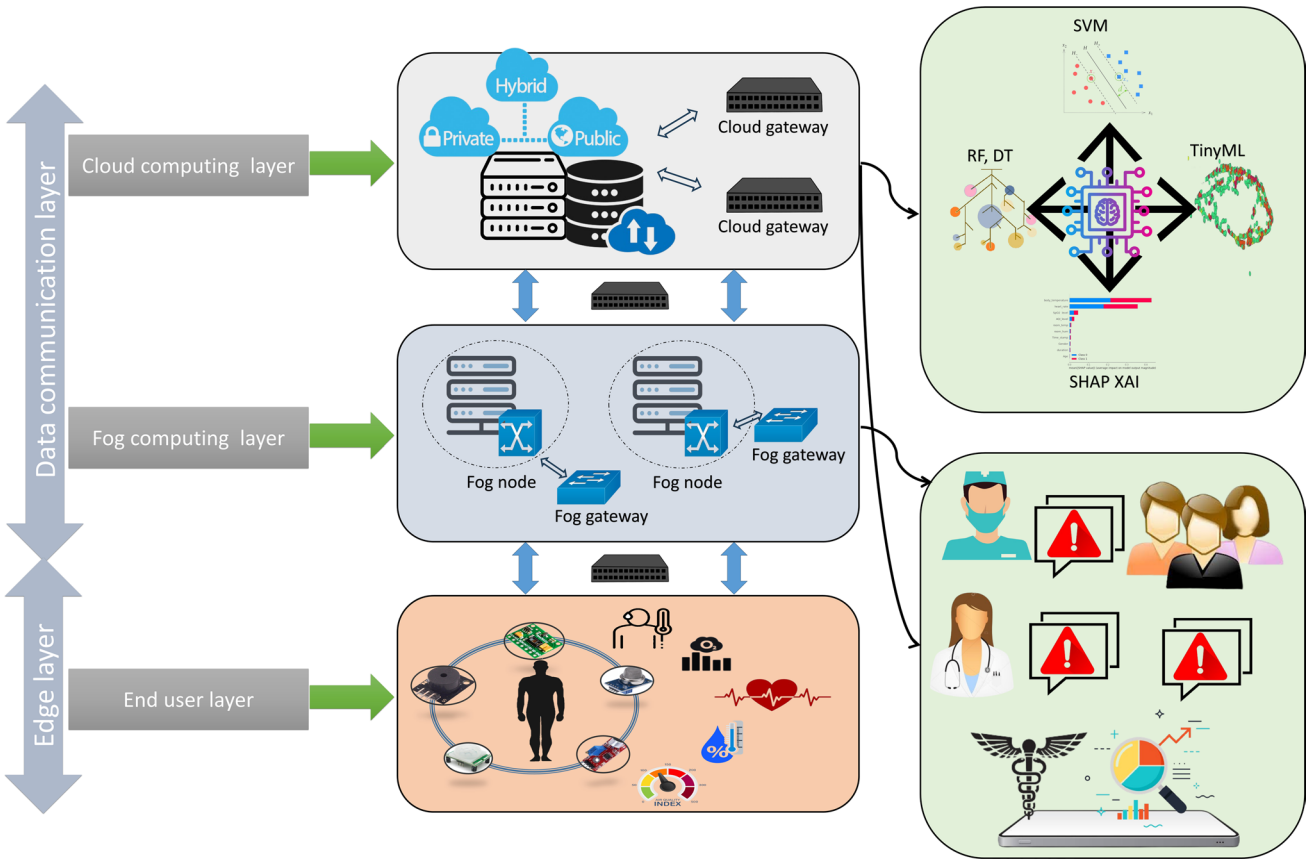
Abstract

The Internet of things (IoT)-based healthcare decision support system plays a crucial role in modern medicine, especially with the rise in chronic illnesses and an aging population necessitating continuous remote health monitoring. Current healthcare decision support systems struggle to deliver timely and accurate decisions with minimal latency due to limited real-time healthcare data and inefficient computational resources. There is a critical need for an optimized, energy-efficient machine learning model that reliably supports remote health monitoring within IoT and fog computing environments. Our study proposes an Optimized Tiny Machine Learning (TinyML) and Explainable AI (XAI) binary classification model for a trustable and energy-efficient healthcare decision support system, leveraging fog computing to optimize performance. The fog-based approach improves response times and enhances bandwidth usage, addressing critical needs such as reduced latency, higher bandwidth utilization, and decreased packet loss. To further improve efficiency, we incorporate the innovative mLZW data compression technique, significantly enhancing data communication efficiency and reducing response time to critical health alerts. However, limited real-time healthcare data records challenge machine learning classification performance. By implementing a TinyML algorithm, our system demonstrates superior performance to other machine learning models. The proposed optimized TinyML model achieves an impressive F1 score of 0.93 for health abnormalities detection, emphasizing its robustness and effectiveness. This paper highlights the potential of TinyML and XAI in delivering robust, trustworthy, and energy-aware healthcare solutions, making significant contributions toward effective remote health monitoring and decision support in fog-enabled IoT networks.

Extended author information available on the last page of the article

Published online: 02 September 2024

Graphical abstract



Keywords Healthcare · Fog computing · Internet-of-things · TinyML · Explainable artificial intelligence

1 Introduction

An IoT-based healthcare decision support system is paramount in today's healthcare domain. With the increasing prevalence of chronic diseases and the aging population, continuously and remotely monitoring patients' health becomes crucial. IoT devices collect real-time data on vital signs, medication adherence, and lifestyle habits, enabling healthcare providers to make informed decisions swiftly. This system enhances patient outcomes by allowing early detection of potential health issues and timely interventions. Moreover, it reduces the burden on healthcare facilities by minimizing hospital visits and enabling efficient resource management.

A noteworthy healthcare decision support system provides decisions with less latency and high accuracy. The current research on fog-based healthcare decision support shows improved response time with less latency and optimized bandwidth usage. This integrates the advantages of fog computing and cloud computing, offering a robust framework for managing, processing, and analyzing massive amounts of healthcare data efficiently and securely.

Fog computing extends cloud services to the edge of the network, closer to the data source. This proximity reduces latency, ensuring real-time data processing and analysis, which is crucial for time-sensitive healthcare applications such as continuous patient monitoring and emergency response. By processing data locally on fog nodes, the system quickly detects anomalies in vital signs and triggers alerts, allowing for prompt medical interventions. Additionally, fog computing reduces the bandwidth required to transmit data to the cloud, alleviating network congestion and improving overall system performance.

The fog nodes perform initial data processing, filtering, and real-time analytics, while the cloud handles in-depth analysis, long-term storage, and integration with other health

information systems. This hierarchical approach not only enhances data security by minimizing sensitive data transmission, but also ensures that healthcare providers have timely access to critical information for decision-making.

Despite the advantages depicted in current research, IoT fog-based healthcare systems still face research gaps as depicted in Table 1. Compared to existing studies, the proposed system uniquely incorporates a complete suite of monitoring features and optimizations. It integrates pulse rate, oxygen level, and sleep monitoring comprehensively, whereas the current studies [1–3] only focus on one or a few of these parameters. The proposed system also ensures the calculation of power and memory consumption, which current studies in [1, 2, 4, 5] do not address. Hence, there is a need for a robust trustable energy-aware healthcare decision support system.

The proposed hardware setup analyzes overall human health parameters with Raspberry Pi as a fog layer. The collected dataset comprises features such as heart rate, oxygen level, human body temperature, room temperature, room humidity, and environment air quality index. The proposed decision support system uses fog-based communication which shows less latency compared to conventional cloud communication. The bandwidth of the network channel is further improved with the proposed optimized data compression technique. The response time of the proposed decision support is less compared to the current research. The number of real-time healthcare data records collected from the proposed system is less for better analysis by the standard machine learning models. Machine learning algorithms require huge data for better classification performance. Hence, the performance measure of standard ML classification shows deprived values in our research. The current research [8–13] shows that TinyML (tiny machine learning) algorithms depict better performance compared to the other machine learning models in handling limited records of IoT datasets.

Applying TinyML for IoT healthcare datasets with limited data records offers several advantages, particularly in resource-constrained environments. TinyML, which refers to machine learning algorithms optimized for tiny, low-power devices, enables IoT devices to perform real-time data analysis and decision-making. This capability is crucial for healthcare applications where timely interventions can significantly impact patient outcomes.

Healthcare providers can use TinyML to detect anomalies in vital signs monitoring by deploying models on wearable devices. These models continuously monitor patient data such as heart rate, blood oxygen levels, and sleep patterns, detecting irregularities that may indicate potential health issues and triggering immediate alerts for early interventions without constant data transmission to centralized servers. The application of TinyML in our proposed system shows significant improvement in classification performance.

After the design and development of a robust healthcare monitor system, there is a major research gap in the current studies [1, 2, 4, 5, 7, 14–19] on proving the trustability of the proposed model. Proving the trustability of the ML model is crucial to ensure reliable, accurate, and unbiased outcomes, particularly in healthcare. Trustworthy models enhance patient data safety and decision-making efficacy.

Our research uses Matthews correlation coefficient (MCC) statistical analysis, SHAP XAI (Shapley Additive exPlanations) feature dependency analysis, and Wasserstein distance between the features in the generated dataset. The MCC values a balanced evaluation of model performance, considering true and false positives and negatives. MCC is particularly valuable for imbalanced datasets, ensuring accurate and comprehensive model validation. SHAP is a powerful tool in Explainable AI (XAI)

Table 1 Study of research gap of the current IoT fog-based healthcare research

	[5]	[3]	[1]	[6]	[4]	[2]	[7]	Proposed system
Pulse rate	✖	✖	✖	✖	✓	✓	✓	✓
Oxygen level	✖	✓	✓	✖	✓	✓	✖	✓
Sleep monitor	✖✖	✖	✖	✖	✖	✖	✓	✓
Room temperature and toxic gas monitor	✓	✖	✖	✓	✖	✓	✖	✓
Fog-based decision support system	✓	✓	✖	✓	✓	✓	✓	✓
Calculation of power consumption	✓	✖	✖	✖	✖	✖	✖	✓
Calculation of memory consumption	✓	✖	✖	✖	✖	✖	✖	✓
optimization of bandwidth	✖	✓	✖	✖	✖	✖	✓	✓
Use of mobile application	✖	✖	✖	✓	✓	✓	✓	✓
ML/DL classification	✓	✓	✓	✓	✓	✓	✓	✓
TinyML classification	✖	✖	✖	✖	✖	✖	✖	✓
Calculation of network packet loss	✖	✓	✓	✓	✓	✖	✖	✓
Model trustability analysis	✖	✓	✖	✖	✖	✖	✖	✓

that visualizes feature dependencies in machine learning models for reliability analysis. It calculates the contribution of each feature to predictions, offering insights into model behavior. Feature dependency plots generated by SHAP illustrate how changes in input features impact model outcomes, aiding in understanding the model's reliability and performance across different scenarios. These visualizations are invaluable for identifying which features significantly influence predictions, potentially uncovering biases or unexpected correlations. Wasserstein distance measures the difference between distributions of features in machine learning models, crucial for reliability analysis. It quantifies how much transformation is needed to align feature distributions, aiding in assessing model robustness and generalization across varying data inputs. Thus, Wasserstein distance informs the stability and consistency of ML predictions.

With these applications in our research, the proposed IoT fog-based healthcare framework proves significant classification performance. Additionally, the proposed model proves to be trustworthy.

1.1 Contributions

The key contributions of our work are:

- Developed an IoT-based healthcare decision support system incorporating the innovative mLZW data compression technique, significantly improving data communication efficiency and reducing response time to critical health alerts.
- Designed and developed the Optimized TinyML (O-TML) binary classification model using TensorFlowLite, outperforming traditional ML models such as decision trees, random forest, and SVM, as well as existing TinyML frameworks in healthcare dataset analysis.
- Conducted comprehensive statistical analysis and evaluated the proposed model's trustability and performance in handling class imbalances using the Matthews correlation coefficient (MCC), demonstrating superior reliability and effectiveness compared to conventional ML models.
- Employed the SHAP XAI algorithm to analyze feature importance and assess model reliability. This enhanced model transparency and trustworthiness by examining feature dependency rates, force plot rankings, and calculating the Wasserstein distance between features.
- Implemented the optimized TinyML and XAI model within a fog-enabled IoT network, improving response times, optimizing bandwidth usage, and addressing critical challenges such as reduced latency, improved bandwidth utilization, and decreased packet loss, achieving an F1 score of 0.93 for health abnormalities detection.

1.2 Paper Organization

Section 2 provides a literature review of the proposed system with the current systems. Section 3 provides the proposed methodology and the components required for the research. Section 4 provides the results obtained from the proposed work and a discussion about the performance of the proposed work. Section 5 provides the conclusion and future works of the research.

2 Related Works

There are various IoT-based healthcare monitor systems proposed for critical patients. Table 2 describes a brief of current research on IoT-based healthcare systems with TinyML application. The recent researches on fog-based healthcare care systems are briefed below:

In [5], to implement a healthcare solution in real-world scenarios, the author has developed and implemented a unique design that combines deep learning with IoT devices. To evaluate the effectiveness of the proposed architecture, the author utilizes Fog Bus, a fog-enabled cloud framework. Through Fog Bus, various performance metrics such as resource usage, network throughput, congestion, precision, and runtime are measured. Furthermore, the model can be configured to operate in different modes to maximize quality of service (QoS) or accurately forecast outcomes in various fog computing settings tailored to different user requirements. This flexibility enables the architecture to adapt and deliver optimal results in different scenarios. The model utilizes Fog Bus, which showcases promising results in terms of resource utilization, network throughput, congestion management, precision, and runtime. Its ability to operate in different modes allows for customization and optimization, ensuring high QoS and accurate predictions in diverse fog computing environments and user-specific scenarios.

The integrated Federated Learning model proposed in [3] included a distributed edge–fog–cloud architecture specifically designed for the IoT smart healthcare industry. The results show that, in every measurable category, the edge-based deployment performs better than the fog and cloud approaches. The edge-based deployment specifically shows improvements of 0.3% in energy consumption, 2% in network utilization, 15% in cost, 11% in execution time, and 3% in latency when compared to fog. The edge-based deployment exhibits even greater benefits as compared to the cloud: 1.6% less energy use, 31% less network usage, 41% less cost, 24% less execution time, and 85% less latency.

Table 2 Comparative study of current IoT-based healthcare monitor system

Refs.	Description	Type of signal	Target device	Generated real-time dataset	Include latency	Includes attacks	Includes tiny ML	ML/DL algorithm	Limitations
[20]	Hearing aid speech enhanced with RNN, pruning, and integer quantization	Audio data	STM32F746VE MCU	✖	✓	✖	✓	Recurrent neural network	The complexity of implementing RNN over the health hardware used is high, which makes it difficult to deploy in a real-time scenario
[8]	Auditory signals for r from Parkinson's disease	Accelerometer signals	ATMega2560 microcontroller	✖	✓	✖	✖	K-nearest neighbors, decision tree, random forest, AdaBoost, SVM	The size and dimension of the dataset used are very small with only two patients considered
[9]	CNN algorithm to detect ECG abnormalities	Single lead electrocardiogram	Chip nRF52 with an ARM's CortexM4 processing core	✓	✓	✓	✖	Convolutional neural network	The applied algorithm performance on the standard dataset is high However, the same algorithm performance on low-power sensor data is reduced
[10]	Wearable device for Epilepsy patients	Brain activity via Electroencephalography	STM32L476 ARM cortex-M4	✖	✖	✖	✖	Random forest	The data wearable device is not analyzed with machine learning algorithms
[11]	Ultrasound imaging for automated liver failure diagnosis	Images obtained via Ultrasound Imaging	CAD transferred to a small memory footprint	✓	✖	✖	✓	Deep neural network	The proposed validation model requires more number of data entry which may lead to network memory leaks
[12]	Emotion Detection with physiological signals	Physiological signals	Shimmer GSR +	✓	✓	✓	✖	Random forest, SVM, logistic regression	
[13]	Heart disease prediction at fog and cloud layer	Bioelectrical signal	Chip nRF52 with an ARM's CortexM4 processing core	✓	✓	✖	✖	K-nearest neighbor	The performance of the multimodal algorithm is less for sensor data
Our work	Healthcare decision support	Accelerometer signals	BCM2711, Quad-core Cortex-A72 (ARM v8) pi processor	Yes	✓	✖	✓	RF, SVM, decision tree	The proposed framework does not consider attack packets

The geographical temporal recurrent neural network proposed in [1] forecasts the encephalitis epidemic outbreak in Bihar. The self-organized mapping (SOM) technique is paired with the T-RNN model to improve the geographical visualization of outbreaks. By gathering AES data, the trilogical IoT–fog–cloud (TIFC) model facilitates spatiotemporal monitoring and epidemic control. Time-series granules at different timestamps are formed by the connections between distinct events created by spatiotemporal patterns. The author uses the FCM model to determine the patient's category. The architecture uses a spatiotemporal-based prediction model to help users make informed decisions related to their health and to give them pertinent information. This strategy demonstrates the effective utilization of medical resources.

In [6], the author developed a fog-level warning system to help drivers when they are driving. To issue alarm messages, the author acquired fog-level data using the NetSim simulator. The neighboring cars were first grouped, following the suggested approach to create a fog cloud. The vehicles grouped around the driver's vehicle receive an alert message in the event of an emergency if the driver's behavior or condition is inappropriate. A virtual fog layer was constructed to receive notifications when the vehicles in the vicinity were not covered by the fog node that had been created. It would be challenging to detect adjacent cars and issue alert messages in real-world situations. These real-world challenges in grouping the surrounding cars must be taken into account by the author.

In the event of a patient emergency, [4] and [2] proposed a fog-level alert system for medical professionals and personal caretakers. Based on the blood sugar level, temperature, and ECG, the author in [4] suggested a J48 graft classifier to categorize the patient's health status as normal or critical. To avoid and forecast COVID-19 patients, the author in [2] offered several machine learning algorithms, including decision trees, random forests, and naïve base methods. The temperature and oxygen saturation level of the patient were deemed noteworthy metrics by the author. The approach did not take into account the real-time IoT hardware configuration. Furthermore, there was no explanation of how an alert system in real-time scenarios was developed.

A fog-level healthcare monitor system was proposed by the author in [7] to identify hypertension instances, notify the physician, and seal the circle under emergency patients. Using patient blood pressure data, the author employed multiple Machine Learning (ML) models to forecast emergencies. For accuracy, sensitivity, and response speed, Artificial neural networks (ANN) performed better than other machine learning techniques. When compared to cutting-edge techniques, the suggested solution demonstrated effective bandwidth usage and decreased latency. Only the hypertension parameter was taken into account by the author when estimating patient death from cardiovascular disease. The other factors were disregarded, including the patient's lifestyle, sleep habits, and surrounding circumstances.

3 Proposed Methodology

Figure 1 shows our research workflow. The architecture of the proposed fog-based decision support system consists of three phases: (1) data collection from the patient health monitor; (2) the fog-based decision support system to deliver emergency alerts to caretakers and doctors through a mobile application; (3) store and analyze the collected data in the tinyML platform. Below is the detailed proposed architecture.

3.1 Data Collection

We propose a hardware-based human healthcare decision support system with the following components in the edge layer.

The MAX30102 sensor is a non-invasive pulse rate and oxygen level monitor system. The sensor runs on a 5v supply from the microcontroller. The red and infrared LED present in the sensor indicates it is working. The integrated glass cover over the sensor protects it from light interference from the external environment. The DHT11 sensor measures the humidity and temperature of the patient's room. The three-pin sensor gets its power from the 5 V supply from the controller board. It covers a humidity range of 20–90% and a room temperature range of 0–50 °C. The sensor provides a resolution of 16-bit for both temperature and humidity measurements.

The Mq-135 gas sensor detects toxic gas near the patient, such as carbon monoxide, methane, hydrogen sulfide, etc., from fire fumes and explosives. Also, it measures the air quality of the room [18]. The sensor attracts oxygen and free electrons from the atmosphere. When introducing a toxic gas, the poisonous gas breaks the oxygen–electron bond and produces heat, predicting toxicity (Fig. 2).

As shown in Fig. 3, the proposed hardware setup uses an Arduino Uno R3 microcontroller to integrate all the sensors into the fog node [14, 21] and the Wi-Fi module. All sensors are connected with a 5v supply from the microcontroller board. The board captures the sensor data and communicates with the fog and cloud layer. The testbed consumes a power of 100–200 mW with a maximum memory usage of 15 kb, since it uses tiny sensors connected to the controller.

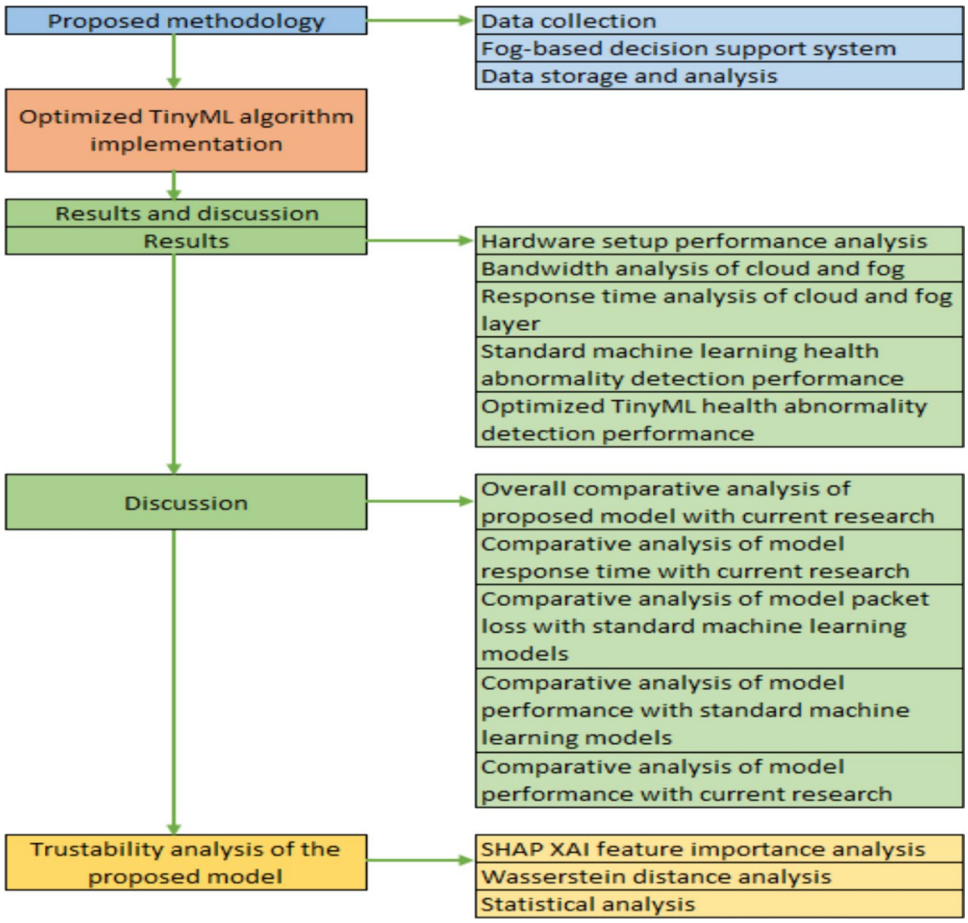


Fig. 1 Overall workflow of the proposed system

3.2 Fog-Based Decision Support System

The data from the controller board reaches the cloud storage through the fog layer. Our work uses Raspberry Pi 3 [17] to set up the fog nodes between the edge and cloud layer. When the patient's health data deviates from the threshold value mentioned in Table 3, a notification reaches five closed people of the patient through the mobile application from the Raspberry Pi 3 node.

The Raspberry Pi 3 microprocessor provides high data processing and communication capability. Hence, it is used as a fog node in the virtual fog layer [16]. It receives and temporarily stores the data from the microcontroller. The microprocessor is connected to the mobile application to send the notifications as shown in Fig. 4. The mobile application used in our work is designed through the MIT app inventor with the following widgets: Human General Health Report; the threshold value of the human body temperature, oxygen level, pulse rate, room temperature, and humidity; the contact number of the doctor, nurse, and three other personal caretakers to whom the alert needs to be sent [22, 23]; decision support notification; health status button to monitor the person's current health as and when required.

3.3 Data Storage and Analysis

The data from the edge layer reaches the cloud storage to visualize the data and analyze any deviation from the threshold value.

ESP8266 Node MCU Wi-Fi module connects the IoT cloud platform with the proposed hardware [24–28]. For the ESP8266 to connect with the Wi-Fi module, the SSID network name and password are provided in the Arduino IDE software and are activated using the ESP8266 library function. Our proposed work uses the Thing Speak IoT cloud platform to store and visualize the data. Figure 5 shows the visualization of the created channel named "health monitor system". The Thing Speak library is uploaded, and the "write API" key from the channel is copied into the Arduino

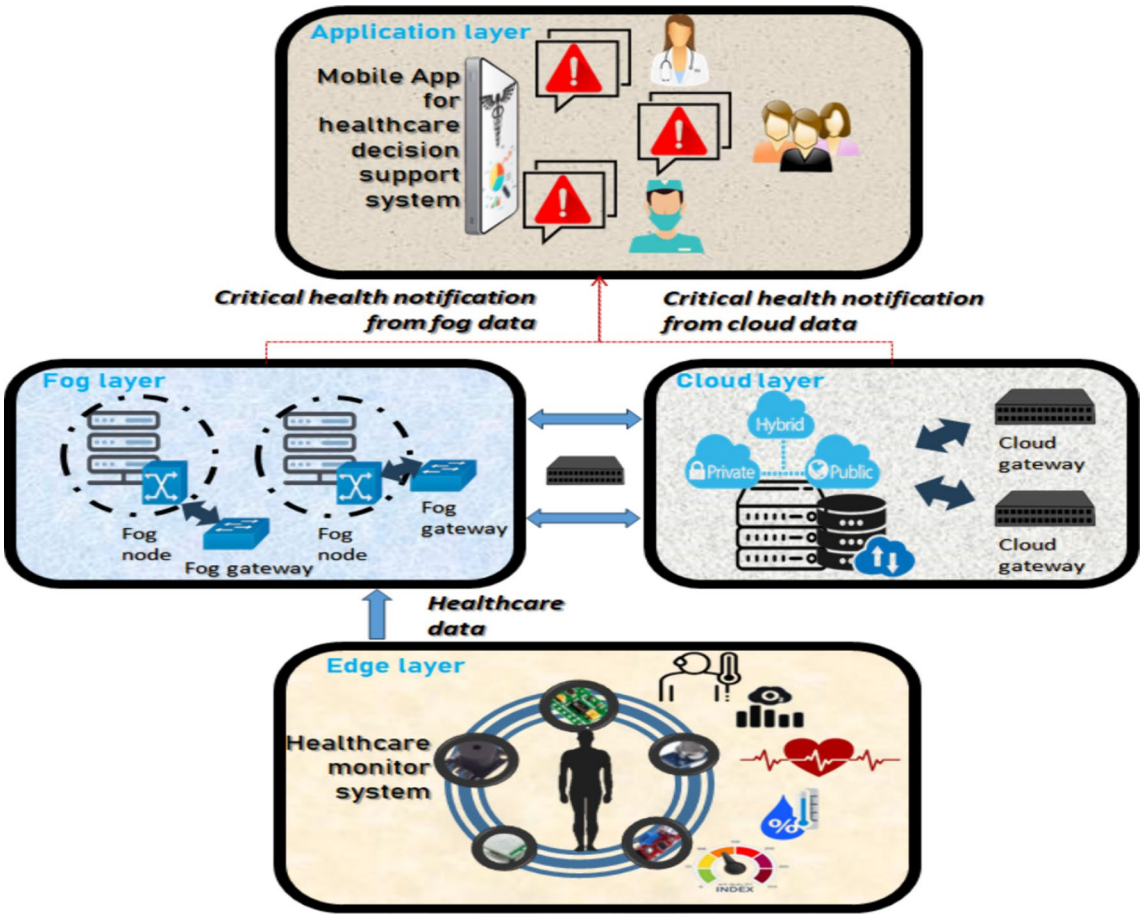


Fig. 2 The architecture of the proposed fog-enabled healthcare decision support system

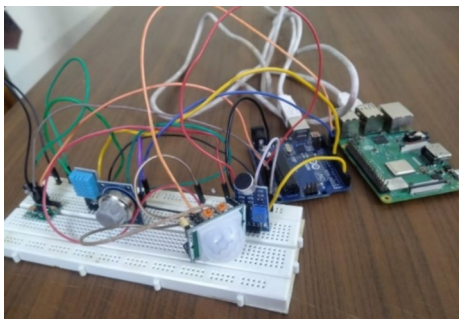


Fig. 3 Hardware setup: edge and fog layer

IDE software to collect the edge data. The.csv file with the collected data is downloaded for data analysis.

Table 3 Human health parameters (threshold values)

S. no	Health parameters	Threshold value
1	Pulse rate	60–100 beats per minute (bpm)
2	Oxygen level	88–94 oxygen saturation (SpO2)
3	Body temperature	35.6–37.4 °C
4	Room temperature	22–26 °C
5	Room humidity	30–60%
6	Room Air Quality Index (AQI)	0–100 AQI

Since the data is collected directly from the proposed hardware setup, the number of entities in the data is significantly less (1100 entities). The experiment includes implementing machine learning models such as SVM, decision tree, and random forest algorithms. The proposed O-TML approach is preferable for the collected dataset from the sensors.

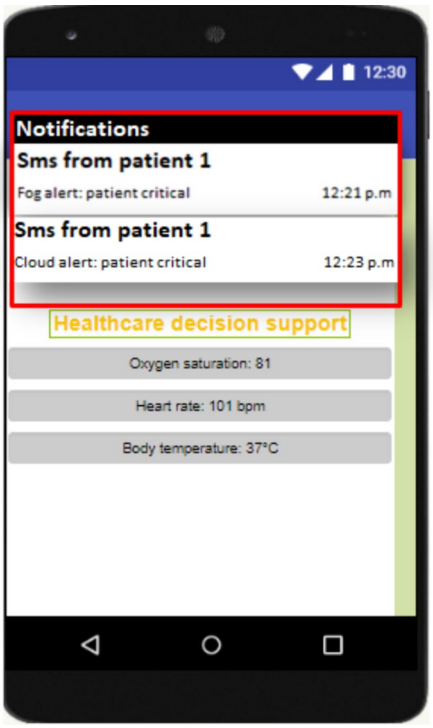


Fig. 4 Mobile app notification from fog and cloud environment

4 Optimized TinyML Algorithm Implementation

The proposed methodology leverages Edge Impulse tinyML software and an Optimized Tiny Machine Learning (O-TML) model for classification tasks within IoT healthcare systems. Our approach involves several critical steps, each contributing to the final model's accuracy, efficiency, and trustability. This section presents a detailed workflow and the mathematical foundations for each phase of the methodology.

4.1 Data Acquisition

To begin, a project is created in the Edge Impulse web interface, where we establish the data type and provide necessary project details such as name, description, and settings. For data acquisition, live categorization is not utilized; instead, the software ingests pre-stored data uploaded from the cloud in CSV format. The configuration involves setting the classification mode as the learning block and raw data as the processing block, with a frequency of 1 Hz and a window size of 1000 ms. This setup ensures that data is segmented into manageable portions, facilitating efficient processing and analysis.

4.2 Preprocessing of Data

Data preprocessing is a crucial step to ensure that the input data is standardized, correctly formatted, and ready for efficient model deployment and training. Edge Impulse provides built-in digital signal processing (DSP) features for signal filtering, which helps remove noise and artifacts from the sensor data. Filters such as band-pass, low-pass, high-pass, and notch filters are applied to enhance the signal of interest and eliminate unwanted frequencies. Normalization is performed to scale disparate sensor data ranges to a common scale using configurable scaling settings, z-score normalization, and min–max scaling techniques. This step ensures that each feature or sensor channel has a comparable range, preventing biases during the model training process.

4.3 Feature Extraction

Feature extraction transforms raw sensor data into meaningful representations that the model can use for effective learning. Edge Impulse offers various feature extraction methods, including Fourier transforms, statistical moments, wavelet transforms, and time-domain signal analysis. These techniques capture essential characteristics of the data, improving the model's performance by focusing on relevant features. The figures provided in the study illustrate the feature extraction process for the generated dataset and the feature explorer for training and testing of healthcare datasets across different epochs (Table 4).

Figure 6 shows the feature visualization of the training dataset. The figure depicts the memory usage and training time of the TinyML model. Figure 7 depicts the feature explorer of the training and testing healthcare dataset for different epochs.

4.4 Dimensionality Reduction and Segmentation

High-dimensional sensor data can pose challenges such as overfitting and increased computational complexity. To address this, principal component analysis (PCA) is employed to reduce the dimensionality of the data while preserving crucial information. PCA involves computing the eigenvectors and eigenvalues of the covariance matrix, allowing us to project the data onto a lower-dimensional subspace that retains the most variance.

Additionally, windowing and segmentation techniques are used to divide long sequences into more manageable segments. This approach helps the model capture local patterns and dependencies within the data, enhancing its ability to learn temporal correlations and improving overall model performance.

Fig. 5 Cloud data visualization of the generated dataset

Table 4 TinyML feature extraction parameters

Input dimension	Learning rate	No. of epochs	Number of dense layers	Number of neurons	Activation function	Output function
10	0.0005	50	2	20, 10	ReLu	Sigmoid
10	0.0005	75	2	20, 10	ReLu	Sigmoid
10	0.0005	100	2	20, 10	ReLu	Sigmoid
10	0.0005	100	3	20, 10, 10	ReLu	Sigmoid

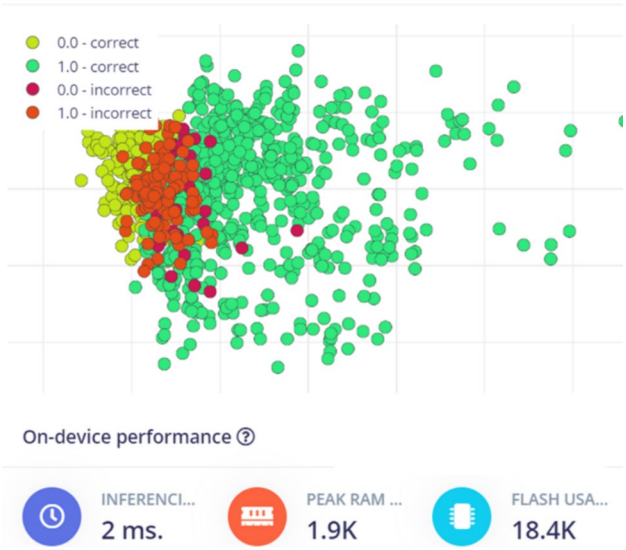


Fig. 6 TinyML training data explorer including inferencing time (model training time) = 2 ms; RAM usage = 1.9 kb; flash memory usage = 18.4 kb

4.5 Model Training and Optimization

The core of our methodology involves training the Optimized Tiny Machine Learning (O-TML) model. For each layer in the trained model, if it is the last layer, model features are extracted. Feature selection is performed using TensorFlow Lite, where statistical features are selected and used to train the model.

The random forest (RF) model is defined and trained using TensorFlow Sequential. The trained model is then converted to TensorFlow Lite format, which involves a series of steps to ensure compatibility and optimization for deployment on edge devices. The TensorFlow Lite model is optimized using default optimization settings, and an interpreter is loaded to run the model.

4.6 Model Validation and Performance

The TensorFlow Lite model's predictions are printed to verify output data, and the model is validated using a radial basis function (RBF) kernel. This validation step ensures the robustness and trustability of the model in real-world scenarios.

Our methodology integrates advanced data processing, feature extraction, dimensionality reduction, and machine learning techniques to develop a robust and efficient TinyML classification model. By leveraging Edge Impulse and TensorFlow Lite, we ensure that the model is optimized for deployment in IoT healthcare systems, capable of providing accurate and reliable classification results with minimal latency and computational overhead.

4.7 Mathematical Basis

The core of our methodology involves Optimized Tiny Machine Learning (O-TML) model leverages neural network architectures with dense layers. The neural network is trained using backpropagation, which involves computing the gradient of the loss function with respect to the network's weights and updating the weights to minimize the loss. Specifically, we employ gradient descent, loss function, ReLU activation function, and principal component analysis (PCA) to develop our approach. The preprocessing steps involve filtering and normalizing the data, which are fundamental operations in signal processing. These steps ensure that the data fed into the neural network is clean and standardized, thereby improving model performance.

We have characterized our methodology with the following mathematical formulae:

(a) **Objective function**:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x_i;\theta), y_i + \lambda R(\theta), \tag{1}$$

where $L(\theta)$ represents the loss function, L is the individual loss for each prediction, $(f(x_i;\theta), y_i)$ is the true label, λ is the regularization parameter, and $\lambda R(\theta)$ represents the regularization term.

(b) **Gradient descent update rule**:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t). \tag{2}$$

In this formula, θ_{t+1} and θt are the parameters at iterations $t+1$ and t, respectively, η is the learning rate, and $\nabla_\theta L(\theta_t)$ is the gradient of the loss function with respect to the parameters.

(c) **Activation function (e.g., ReLU)**:

$$f(x) = \max(0, x). \tag{3}$$

(d) **Output prediction**:

$$\hat{y} = f(Wx + b), \tag{4}$$

where $\hat{y}$ is the predicted output, W represents the weights, x is the input, and b is the bias term.

4.8 Feature Extraction

Feature extraction transforms raw data into a set of features that are more meaningful for the learning algorithm. This involves several techniques such as filtering, normalization, and dimensionality reduction.

Signal filtering: To remove noise and artifacts from sensor data, we apply digital signal processing (DSP) techniques.

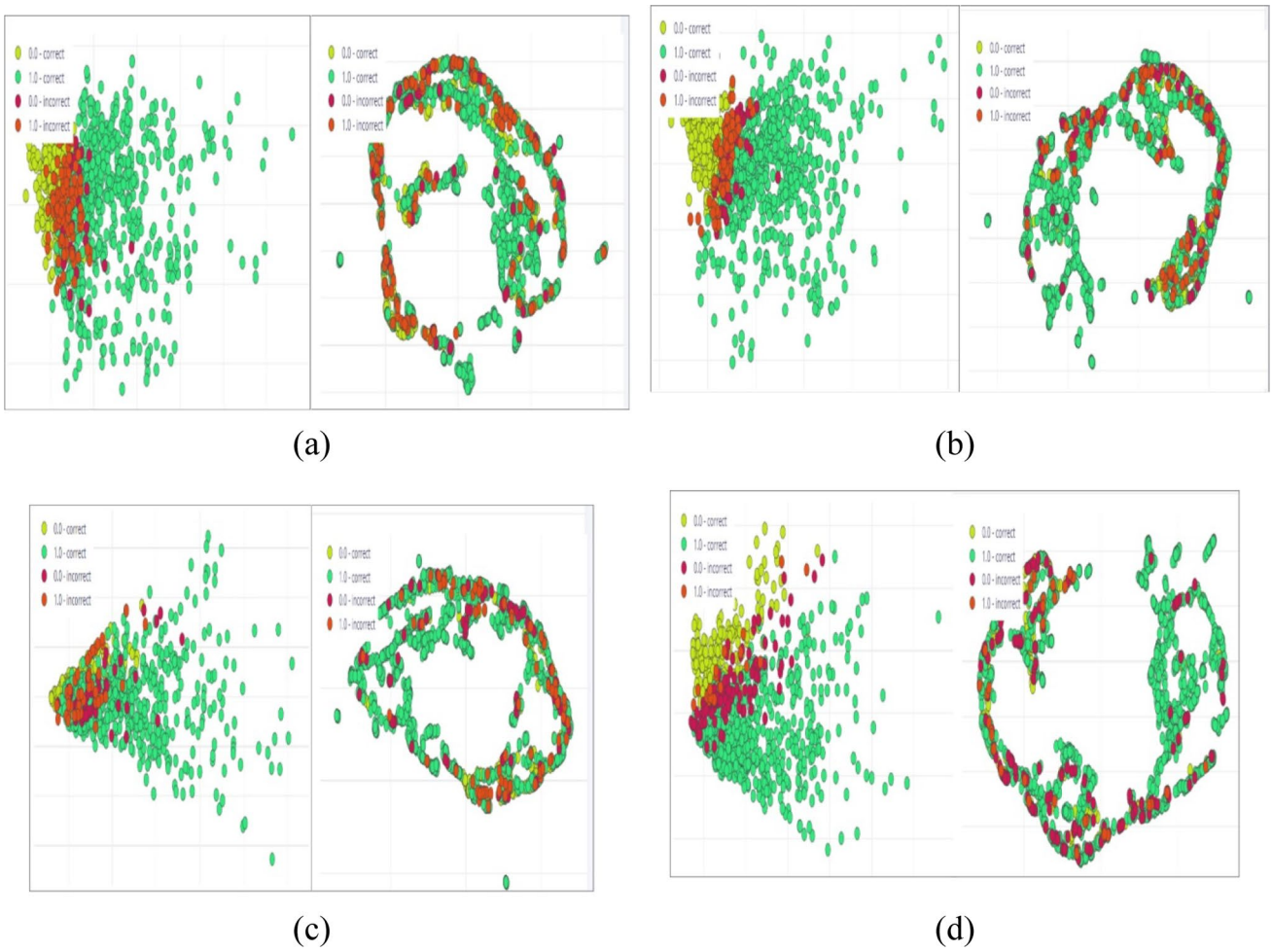


Fig. 7 TinyML feature cluster distribution (green and yellow (correct) and red and purple (incorrect) indicate prediction accuracy): **a** epoch = 50; **b** epoch = 75; **c** epoch = 100, dense layer = 2; **d** epoch = 100, dense layer = 3

For example, a band-pass filter can be mathematically represented as:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\mathrm{d}\tau, \tag{5}$$

where $x(t)$ is the input signal, $h(t)$ is the impulse response of the filter, and $y(t)$ is the filtered output.

Normalization: Normalization scales the data to a common range. One common method is z-score normalization:

$$z = \frac{x-\mu}{\sigma}, \tag{6}$$

where x is the data point, μ is the mean, and σ is the standard deviation.

Feature extraction methods, such as Fourier transforms and statistical moments, derive meaningful representations from raw sensor data. The extracted features (F) are used to improve model performance:

$$F = \text{Transform}(x).$$

Dimensionality reduction (PCA): Principal component analysis (PCA) reduces the dimensionality of the data by projecting it onto a lower-dimensional subspace that maximizes variance.

Mathematically, this involves computing the eigenvectors ($\mathbf{v}$) and eigenvalues (λ) of the covariance matrix $\mathbf{C}$:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}. \tag{7}$$

The transformed data $X_{\text{transformed}}$ is obtained by projecting the original data X onto the selected eigenvectors:

$$X_{\text{transformed}} = X\mathrm{W}, \tag{8}$$

where $\mathbf{W}$ is the matrix of selected eigenvectors.

Algorithm for proposed O-TML classification:

Input: Raw healthcare data
Output: Trained and optimized TinyML classification model

Initialize:
Pandas as pd
Numpy as num
TensorFlow as tf
Call lite function from TensorFlow

for each input **do**
for each model layer **do**
Predicted label ← call forward pass (in, target, features)
call calculate loss (Predicted features, features)
call adam optimizer
call optimization backward Pass() → To reduce loss
end for
end for
trained model ← {i, model layer, wf, bf}
for the model layer in the trained model **do**
if the model layer is n − 1 then
model features ← {i}
end if
train_df = call TENSORFLOWLITE FEATURE SELECTION (train statistical features)
for features in train_Df() **do**
Train RF model = tf.keras.Sequential([tf.keras.layers.Input(shape=(X_train.shape[1],)),
tf.keras.layers.Dense(units=1)
convert lite.TFLiteConverter.from_keras_model(RF model)
convert tflite to tflite. optimize.DEFAULT
load tflite_model interpreter
print (model_precition of output data)
validate tflite_model (RBF kernel)
end for
end for
end for
Output: Trained and optimized TinyML classification model

5 Results and Discussion

The evaluation of the proposed healthcare system takes into account multiple important factors. First, it closely examines precise and prompt patient data from health monitoring. Second, the efficacy of the proposed mLZW data compression technique for enlarged bandwidth and reduced response time is analyzed. Third, the notification efficacy is evaluated, with a focus on the alert triggers' accuracy and response to important conditions. Fourth, the comparative evaluation of the proposed OH-TinyML classification model with the current research is performed. Finally, the proposed system is checked for trustability through various metrics such as model specificity and sensitivity; model statistical analysis; SHAP XAI feature importance analysis; and features Wasserstein distance calculation.

Table 5 The detection range of the sensors used

Sensor used	Detection range	Working voltage (V)	Connected pin to the controller
MQ135	10–1000 ppm	2.5–5.0	D10
MAX30102	40–85 °C	3.3–5.5	A0
KY-037		3.3–5.5	D2
DHT11 (tem)	0–50 °C	3.3–5.5	D7
DHT11 (hum)	20–90%	3.3–5.5	D7

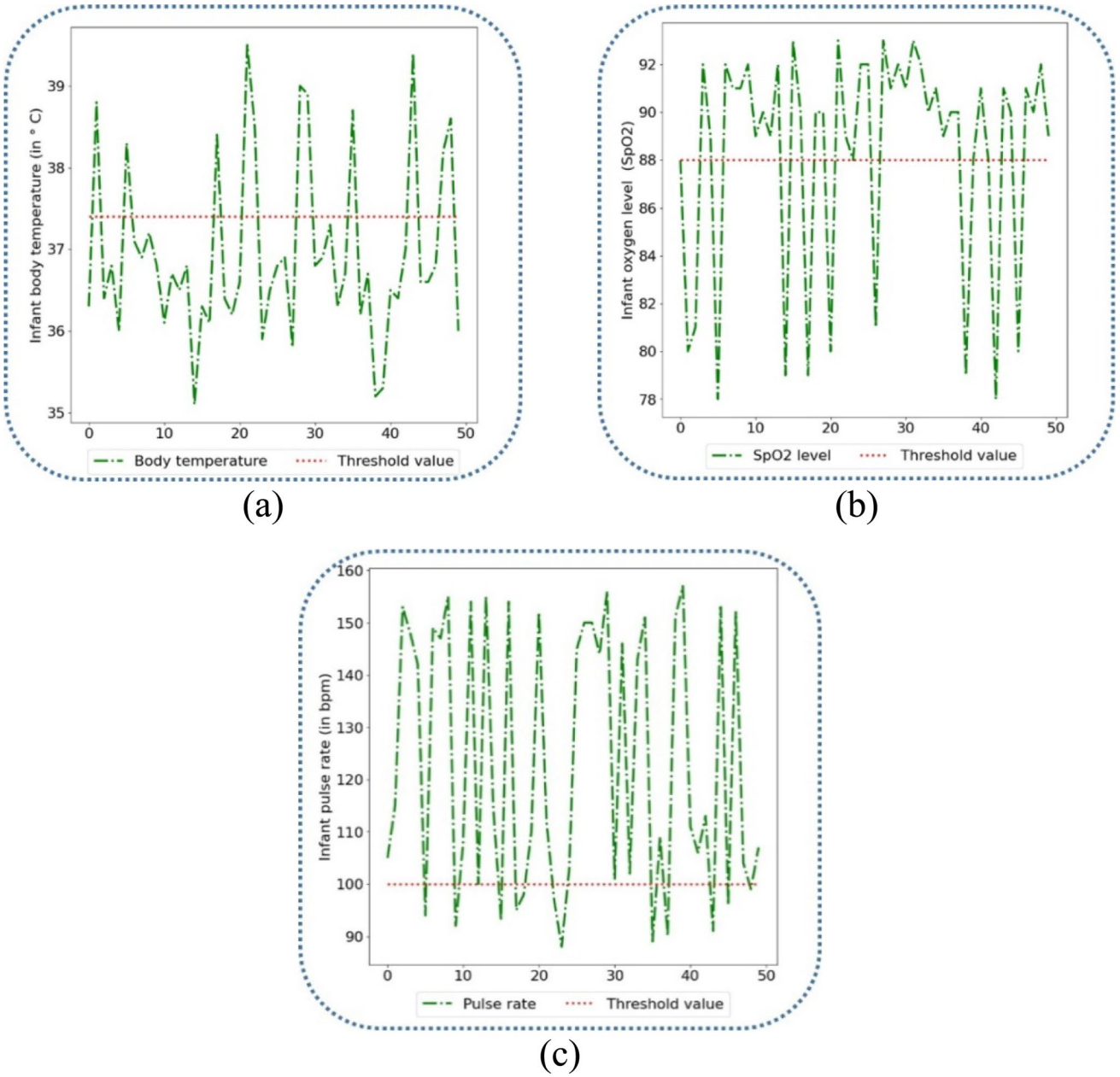


Fig. 8 Parameter analysis of the healthcare decision support system based on: **a** body temperature, **b** oxygen level, and **c** pulse rate

5.1 Results

This section depicts the analysis of the fog-based decision support system, bandwidth, response time, and performance comparison of optimized TinyML with standard machine learning models.

5.1.1 Hardware Setup Performance Analysis

As shown in Table 5, since the sensors used in the proposed system have a high detection range, the overall performance of the proposed hardware setup for continuous health monitor is high. The Raspberry Pi 3 used as a fog node is time sensitive and requires less computation power. It is easily adaptable with the Arduino Uno R3 microcontroller and ESP8266 Wi-Fi module. Also, the Raspberry Pi 3 processor is compatible with the mobile application. The values from these sensors are monitored remotely by healthcare workers and personal caretakers.

Figure 8 shows the patient body temperature, oxygen level, room temperature, and humidity variation from the threshold values as visualized in the Thing Speak cloud

platform. The notification received from the fog layer to the mobile application is programmed based on this threshold value of the patient's health status.

When the human body temperature goes beyond 37.4, the person suffers from fever, which may lead to fits if unattended. Hence, the health workers and caretakers immediately receive a notification when the body temperature increases. The SpO2 level indicates the oxygen breathability of the human. In general, the SpO2 level lies between 88 and 94. If the MAX30102 sensor senses the oxygen level below 85, the person tends to suffer from breathlessness. In this case, the health workers receive a notification. The pulse rate indicates the heart rate, which generally measures from 60 to 100 bpm. If the pulse goes below 60 bpm, this shows that the person's heart is not functioning well and may lead to death or coma. Also, a pulse rate of more than 100 bpm says that the person is restless, or the heart muscles are too fragile to function due to some infectious virus. In both cases, a notification is sent to provide immediate medical service.

5.1.2 Bandwidth Analysis of Cloud and Fog Layer

In our work, we execute a modified LZW (mLZW) data compression technique, which takes a series of symbols, strings them together, and finally turns the strings into codes. The technique uses CHAR and STR to perform the compression. A set of one or more characters is stored in STR, while a single character, or a single byte value between 0 and 255, is stored in CHAR. Each character in the STR has a single byte. Reading bytes from the input file once again and saving them in the CHAR creates a data table. To find out if a code has already been assigned to the string and character combination, this table is examined. This table has a total list size of 2N strings and characters. The series of symbols is encoded by the algorithm using a fixed length code, taking advantage of the N-bit index in the table for this purpose. If the bit length used to encode the sequence is 12 bits, the index of this combined list with an 8–12 bit symbol sequence is encoded into 12 bits.

Several indicators are used in current research to analyze the effectiveness of data compression algorithms. This study

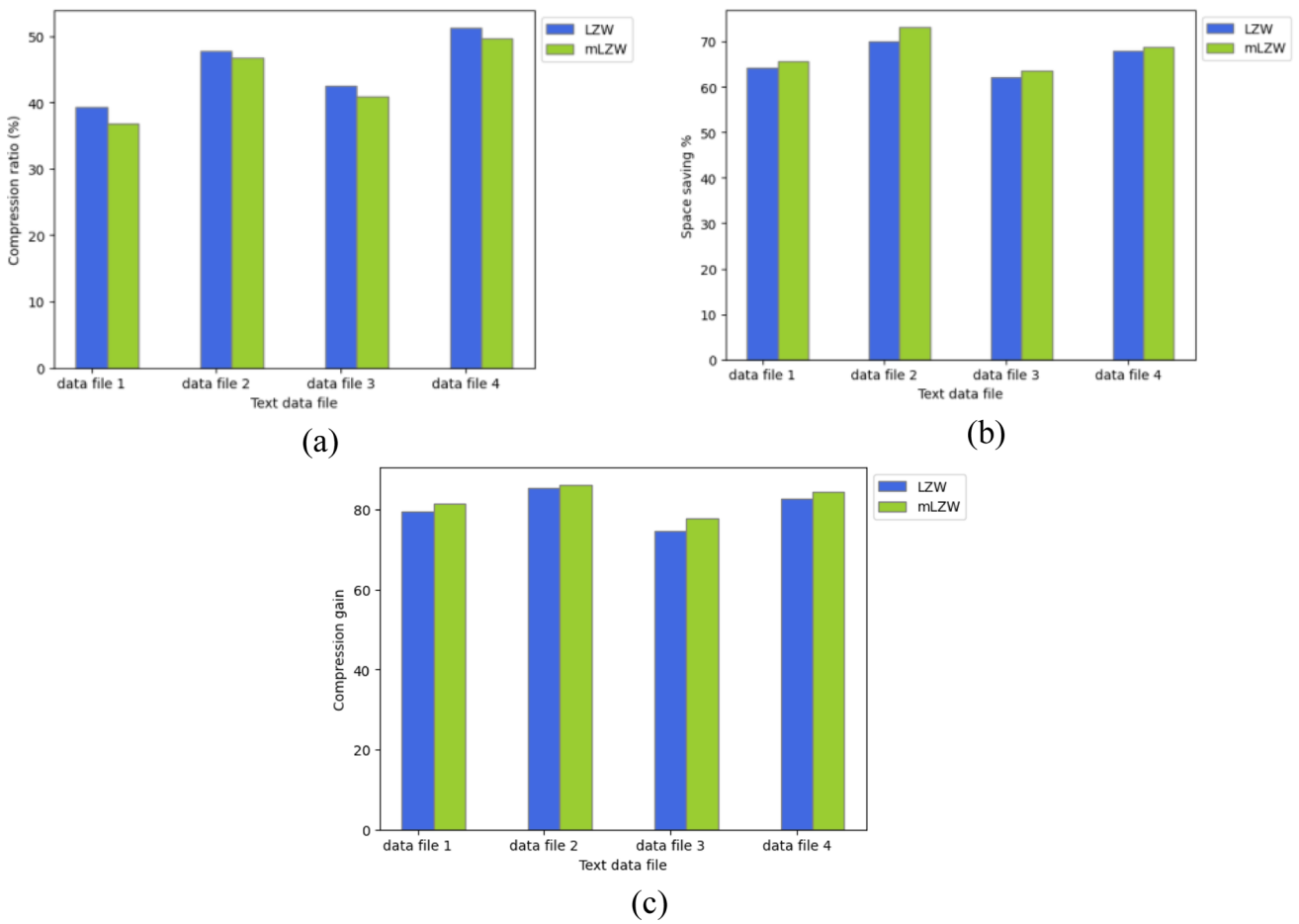


Fig. 9 Performance comparison of mLZW data compression technique based on **a** bandwidth compression ratio, **b** space saving percentage, and **c** bandwidth compression gain

makes use of the compression ratio, compression gain, and percentage space savings. The compression ratio describes the average number of bits needed to store the compressed data.

$$rmData \text{ compression ratio (CR)} = \frac{N_c}{N_{unc}} = \frac{S_c}{S_{unc}}, \tag{9}$$

where N_c and N_{unc} stand for the number of bits in the compressed and original data, respectively, and S_c is the size of the compressed data and S_{unc} is the size of the original data. It is essential to assess compression strategies based on the amount of space they save, because they enable the most effective use of storage. Data space saving percentage measures the reduction in data size achieved through compression relative to the original size. Therefore, percentage space savings (SS%) are also considered. This index, which shows the reduction in file size relative to its initial size, is expressed by Eq. (3). The analysis also considers the compression gain (CG) of each technique.

$$\text{Data compression space saving percentage(SS\%)} = \left(\frac{S_{unc} - S_c}{S_{vnc}}\right) \times 100\%, \tag{10}$$

$$\text{Data compression gain(CG)} = 100 \log_e\left(\frac{S_{wnc}}{S_c}\right) = 100 \log_e(Cf). \tag{11}$$

It is observed that mLZW had the lowest CR in Fig. 9a. This indicates that employing this approach to store the compressed file will take fewer bits. Furthermore, mLZW exhibited a maximum CR of 49.6 according to Fig. 9a analysis, which represents a bandwidth of 18kbps. This suggested that it might result in a 50% reduction in data size. It is clear from Fig. 9b that the mLZW algorithm provides superior space-saving.

5.1.3 Response Time Analysis of Cloud and Fog Layer

The response time of the proposed system depends on the latency caused by the fog and cloud layer to send the notification message. The latency produced by the fog and cloud layer to transmit the notification message determines how quickly the proposed system responds. The data communication from the sensor nodes to the mobile device, fog layer, and cloud layer; (1) data propagation through the network channels during communication; (2) data processing during notification and analysis; (3) miscellaneous factors such as data lag from malfunctioning sensor nodes, data queuing, and other wiring delays all contribute to the latency in the proposed healthcare system.

The data communication is given by the summation of uplink communication (C_u) and downlink communication (C_d), which is the time the data takes to reach the destination and the time taken for the response data to reach the source or the monitor system.

$$\text{Uplink communication} \quad (C_u) = (1 + DU_f) * (DU_a/DU_t), \tag{12}$$

where DU_f is the data failure rate during uplink communication, DU_a is the amount of transmitted data, and DU_t is the data transmission rate.

$$\text{Downlink communication} \quad (C_d) = (1 + DD_f) * (DD_a/DD_t), \tag{13}$$

where DD_f is the data failure rate during downlink communication, DD_a is the amount of transmitted data, and DD_t is the data transmission rate.

$$\text{Data\_communication} = (1 + DU_f) * (DU_a/DU_t) + (1 + DD_f) * (DD_a/DD_t). \tag{14}$$

The uplink communication delay (DUP) includes data transfer from sensor nodes to the mobile (DC_{sm}) of the health workers and caretakers, then to the fog layer for processing (DC_{mfp}); the data is transferred next from the fog processing layer to the fog transmitting layer (DC_{fpft}) and finally to the cloud layer (DC_{ftc}).

$$DUC = DC_{sm} + DC_{mfp} + DC_{fpft} + DC_{ftc}. \tag{15}$$

The downlink communication delay of our proposed work can happen in two ways: (1) alert message delay from the fog layer to the mobile device (DDP_{fm}) or (2) alert message delay from the cloud layer to the mobile device (DDP_{cm}).

$$DDC_{cm} = DDC_c + DDC_{fm}, \tag{16}$$

$$D_{com}(fog) = DC_{sm} + DC_{mfp} + DC_{fpft} + DC_{ftc} + DDC_{fm}, \tag{17}$$

$$D_{com}(cloud) = DC_{sm} + DC_{mfp} + DC_{fpft} + DC_{ftc} + DDC_{fm} + DDC_c. \tag{18}$$

The propagation delay (D_{prop}) is given by the delay caused due to the data propagation from sensor nodes to the cloud layer. This includes the data propagation from sensor nodes to the mobile (DP_{sm}) of the health workers and caretakers, then to the fog layer for processing (DP_{mfp}), the data next from the fog processing layer to the fog transmitting layer (DP_{fpft}), and finally to the cloud layer (DP_{ftc}).

$$D_{prop}(fog) = DP_{sm} + DP_{mfp} + DP_{fpft}, \tag{19}$$

$$D_{prop}(cloud) = DP_{sm} + DP_{mfp} + DP_{fpft} + DP_{ftc}. \tag{20}$$

The processing delay ($D_{proc)}$ in our proposed system is caused in two ways: (1) data processing delay at the fog

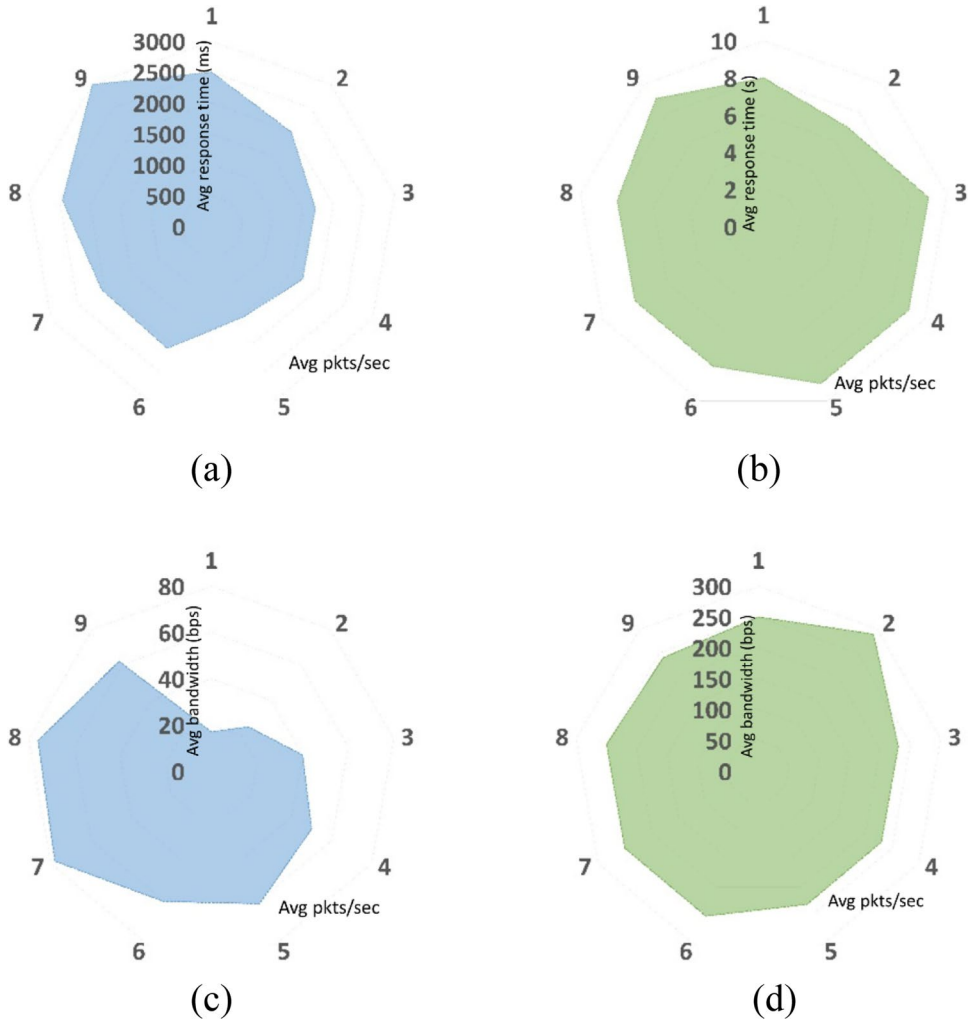


Fig. 10 **a** Response time of fog layer, **b** response time of cloud layer, **c** bandwidth of fog layer, and **d** bandwidth of cloud layer

layer ($\mathrm{DPR_f}$) and (2) data processing delay at the cloud layer ($\mathrm{DPR_c}$). The total processing delay of the fog layer ($\mathrm{DPR_f}(t)$)is due to fog level processing delay to determine the human health factors (from 1 to h factors—DPR_{fh}) and to send the alert message($\mathrm{DPR_f}(a)$).

$$\mathrm{DPR_f}(t) = \{(\mathrm{DPR}_{f1+}\mathrm{DPR}_{f2+}\mathrm{DPR}_{f3} \ldots \mathrm{DPR}_{fh}), \mathrm{DPR_f}(a)\}. \tag{21}$$

Therefore, the processing delay due to the fog layer is taken as the maximum of the total processing delay.

$$\begin{aligned}\mathrm{DPR_f} &= \max(\mathrm{DPR_f}(t)) = \max\{(\mathrm{DPR}_{f1} \\ &\quad + \mathrm{DPR}_{f2} + \mathrm{DPR}_{f3} \ldots \mathrm{DPR}_{fh}), \mathrm{DPR_f}(a)\} \\ \mathrm{DPR_c} &= \mathrm{DPR_f} + \mathrm{DPR_{ci}},\end{aligned} \tag{22}$$

$$D_{\mathrm{proc}}(\mathrm{fog}) = \max\{(\mathrm{DPR}_{f1+}\mathrm{DPR}_{f2+}\mathrm{DPR}_{f3} \ldots \mathrm{DPR}_{fh}), \mathrm{DPR_f}(a)\}, \tag{23}$$

$$D_{\mathrm{proc}}(\mathrm{cloud}) = \max\{(\mathrm{DPR}_{f1+}\mathrm{DPR}_{f2+}\mathrm{DPR}_{f3} \ldots \mathrm{DPR}_{fh}), \mathrm{DPR_f}(a)\} + \mathrm{DPR_{ci}}. \tag{24}$$

The overall delay of the proposed work is given by

$$\begin{aligned}\mathrm{Latency\,(fog)} &= D_{\mathrm{com}}(\mathit{fog}\ \mathrm{layer}) + D_{\mathrm{prop}}(\mathit{fog}\ \mathrm{layer}) \\ &\quad + D_{\mathrm{proc}}(\mathrm{fog\ layer}),\end{aligned} \tag{25}$$

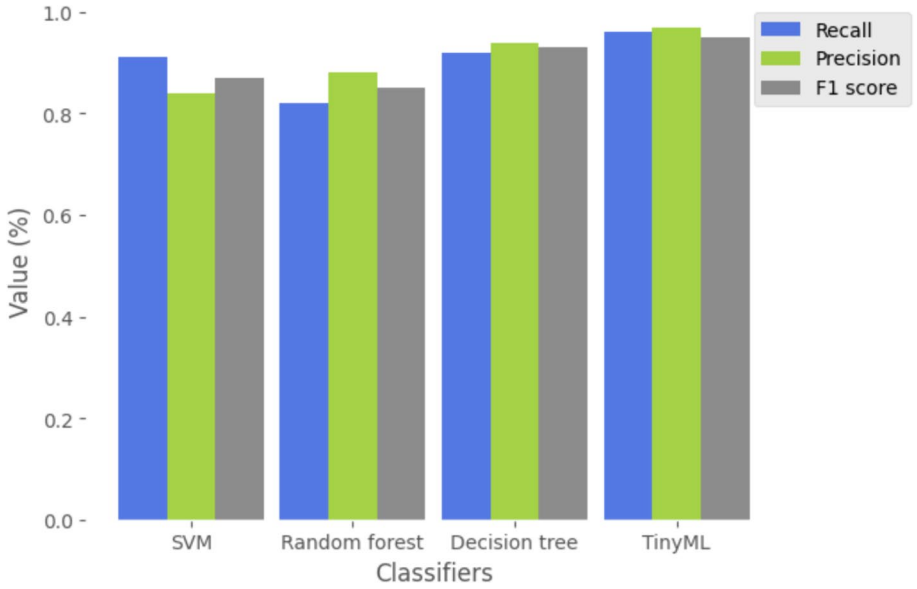


Fig. 11 Classification performance of various machine learning models

Table 6 Training performance of TenFlowLite TinyML and Edge Impulse TinyML

TinyML implementation	No. of epochs	No. of dense layers	Precision	Recall	F1 score	Accuracy	Loss
O-TML model (TensorFlowLite)	50	2	87.3	92.4	89.8	91	0.23
	75	2	91	89.9	90.4	92.5	0.19
	100	2	88.1	92.8	90.4	91.6	0.32
	100	3	88.1	95.4	91.6	92.4	0.24
Edge Impulse TinyML model (software)	50	2	88.9	93.2	91	92	0.23
	75	2	87.4	95.1	91.1	93.4	0.19
	100	2	90.2	94.6	92.3	92.9	0.32
	100	3	88.1	97.5	92.6	92	0.24

Table 7 Testing performance of TenFlowLite TinyML and Edge Impulse TinyML

TinyML implementation	No. of epochs	No. of dense layers	Precision	Recall	F1 score	Accuracy
O-TML model (TensorFlowLite)	50	2	83.2	85.1	84.1	86.54
	75	2	82.1	88.3	85.1	89.33
	100	2	85.4	92.6	88.9	89.98
	100	3	87.6	94.3	90.1	91.3
Edge Impulse TinyML model (software)	50	2	88.8	89.6	89.2	89.39
	75	2	85.7	94.3	89.8	91.82
	100	2	86.8	94.5	90.5	91.85
	100	3	88.43	91.8	90.8	90.47

$$\text{Latency} \quad (\text{cloud}) = D_{\text{com}}(\text{cloud layer}) + D_{\text{prop}}(\text{cloud layer}) + D_{proc}(\text{cloud layer}), \tag{26}$$

$$\Delta \text{ latency} \quad (\text{cloud - fog}) = \text{DDC}_{\text{c}} + \text{DP}_{\text{ftc}} + \text{DPR}_{\text{ci}}. \tag{27}$$

From Fig. 10,

average response time (fog) = 8.44 ms; average response time (cloud) = 2116.66 ms; average bandwidth (fog) = 246.66 bps; average bandwidth (cloud) = 52.11 bps latency = 27 ms.

Table 8 Overall comparative analysis of the proposed system with the current research

Refs.	Contribution	IoT device	Data storage	Communication technique	Visualization	Decision support system	ML application	TinyML application	Evaluation metrics	Trustability analysis	Trustability analysis tool
[30]	Personalized healthcare decision support system	IoT Sensor data	Local	Fog and cloud computing	Yes	Alert on Android application	J48Graft decision tree	No	Precision, recall, F1 score	No	No
[1]	Predict encephalitis disease in Bihar	Geographical visualization	Cloud	Fog and cloud computing	Geographical outbreak visualization	Yes	RNN	No	Reliability, Precision, recall, F1 score	Yes	From evaluation metrics
[6]	Driver assistance on the road	NetSim simulator	Cloud	Fog and cloud computing	Yes	Yes	No ML applied	No	No	No	No
[5]	Human critical health monitor system	IoT data	Cloud	Fog and cloud computing	Yes	No mobile application	Fog Bus	No	Precision, recall, F1 score	No	No
[3]	Patient ECG monitor system	Microservice-based IoT application	Cloud	Fog and cloud computing	Yes	No mobile application	Federated learning on edge devices	No	Accuracy	No	No
[4]	Emergency health monitor system	No	Cloud	Fog and cloud computing	Yes	Yes	Decision tree, naïve based, random forest	No	Reliability, Precision, recall, F1 score	Yes	From feature importance
[2]	Predict COVID attacks in patients	No	Local	Fog and cloud computing	Yes	Yes	No ML applied	No	No	No	No
[7]	Patient hypertension alert system	IoT data	Cloud	Fog and cloud computing	Yes	Yes	ANN	No	Accuracy	No	No
[10]	Activity monitor system	Sensor data	Cloud	Cloud computing	TinyML visualization	No mobile application	TinyML	Embedded TinyML	Precision, recall, F1 score	No	No
Our work	Reliable human healthcare decision support system	IoT Sensor data	Cloud	Fog and cloud computing	TinyML and XAI visualization	Yes	TinyML	TensorFlowLite, Edge Impulse	Reliability, Precision, recall, F1 score	Yes	Shap XAI

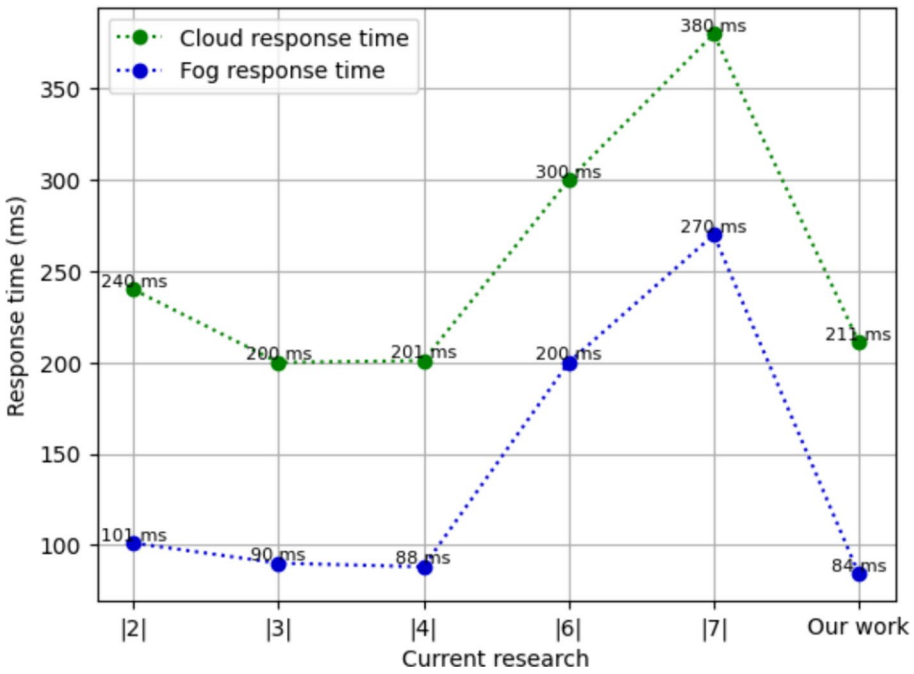


Fig. 12 Comparative analysis of the proposed model response time with current research

Equation 27 indicates the additional latency the cloud layer takes over the proposed fog layer. The response time depicted in 7 shows that the notification reaches the mobile application with significantly less latency when compared to the message received from the cloud layer. Hence, during emergencies, the notification from the fog layer is highly efficient in treating the patients before they attain a critical health status. Also, the fog layer provides more bandwidth for incoming messages than the cloud layer. Hence, the proposed system proves that the fog layer can handle more data efficiently than the cloud layer.

5.1.4 Standard Machine Learning Health Abnormality Detection Performance

The dataset collected from the proposed system is first tested with algorithms such as SVM, random forest, and decision tree. Then, it is applied to Edge Impulse tinyML [29] software. Next, the proposed O-TML model is applied. Figure 11 depicts the interpretation of the F1 score of different machine learning algorithms along with tinyML output. Figure 11 shows that the tinyML classification model outperforms the other ML models for the generated dataset. Table 9 explains the overall performance of the proposed tinyML model. Table 6 lists the training performance of tinyML for various epochs, and Table 7 lists the testing performance of tinyML for multiple epochs.

5.1.5 Optimized TinyML Health Abnormality Detection Performance

When evaluating a tinyML model's performance, several metrics are used to assess its effectiveness. From Tables 6 and 7, the model produces a high F1 score of 0.95 for 100 epochs and three dense layers: first with 10 neurons, second with 20 neurons, and third with 10 neurons. The training loss is 0.24. From Table 6, we infer that the model produces the same high F1 score with 100 and 75 epochs and two and three dense layers.

The above tables show that the O-TML model produces less data training loss for 75 epochs. With the increased epochs, when the number of dense layers is increased, there is a considerable decrease in the loss percentage. Also, with the increased number of dense layers, training, and testing data recall value is high.

5.2 Discussion

The result analysis of the proposed system is broadly discussed under two main categories as stated below. The comparative analysis of the proposed model provides a result comparison with the current research on various factors, model training and testing performance comparison, and loss analysis. The proposed model reliability analysis provides the application of XAI on our collected dataset to generate feature ranking and its impact on the classification result; model specificity and sensitivity; model statistical analysis; and features Wasserstein distance calculation.

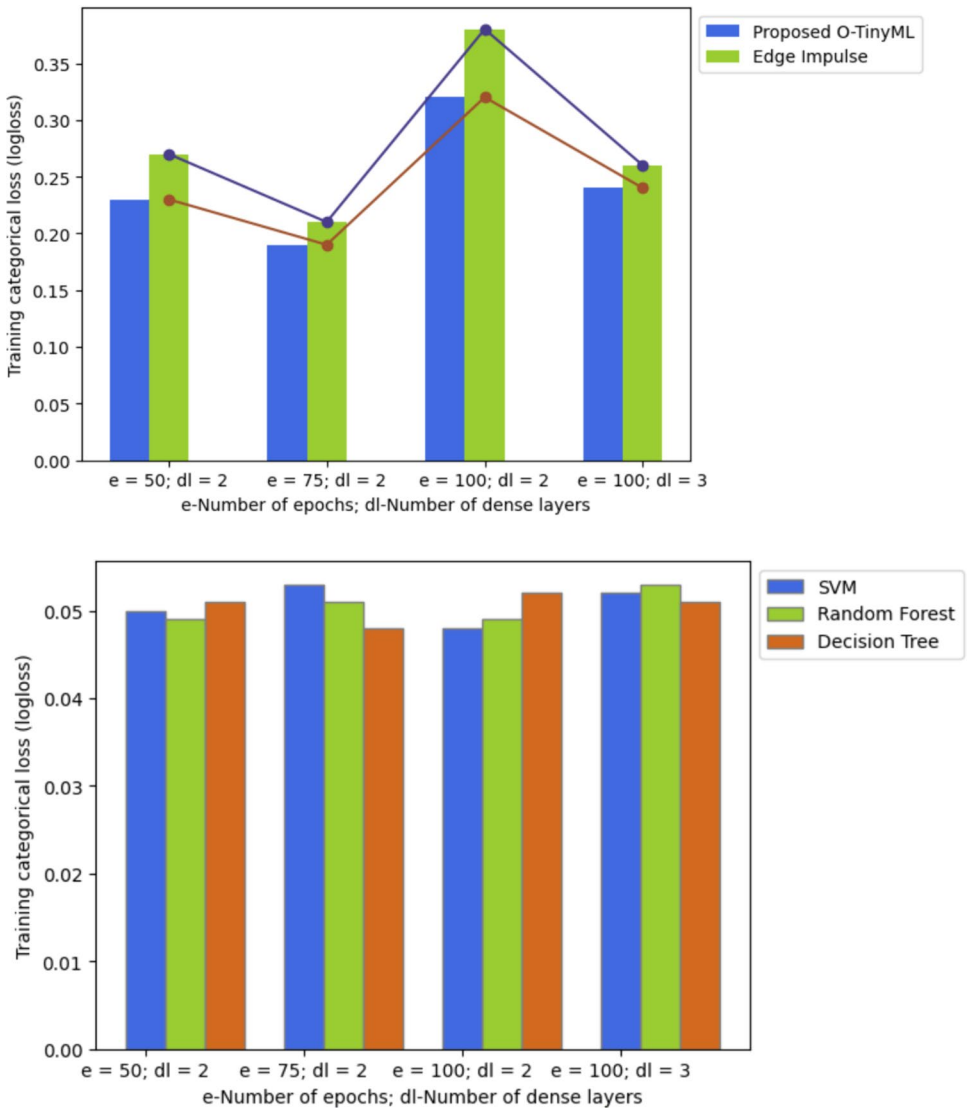
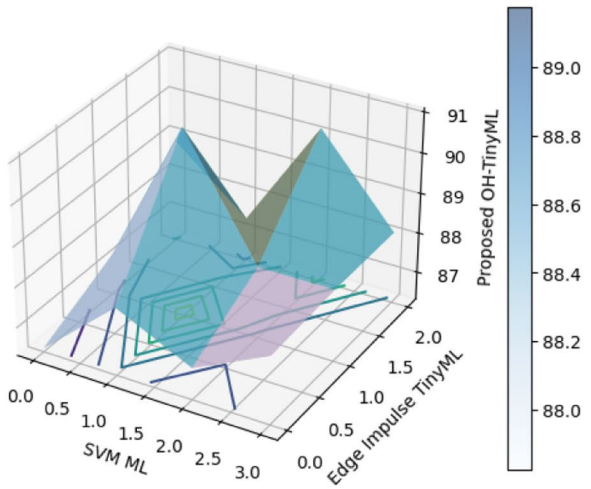


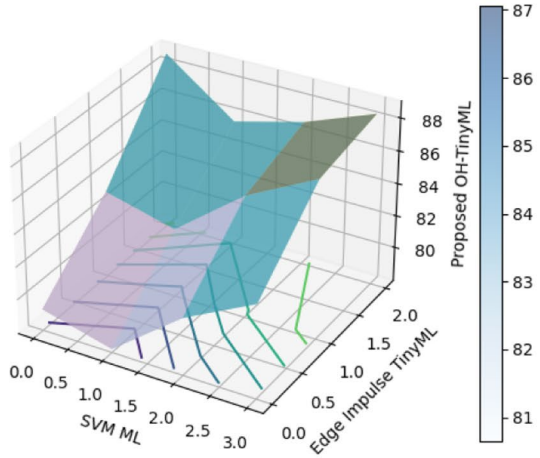
Fig. 13 Network training packet loss comparison calculation of **a** categorical loss in TinyML models and **b** categorical loss in ML models

5.2.1 Overall Comparative Analysis of the Proposed Model with Current Research

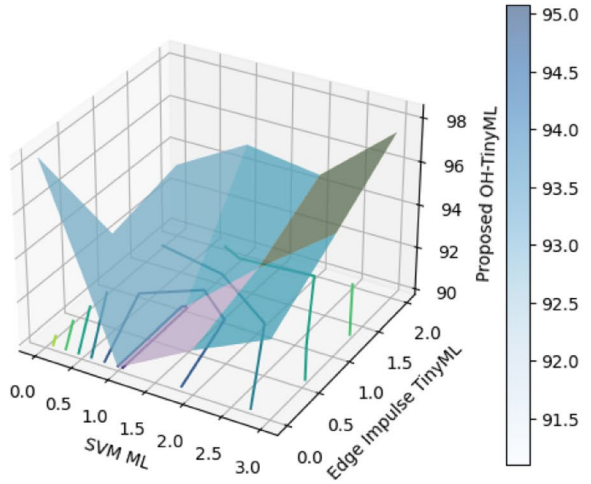
The proposed model provides various distinct elements compared to the current research. Table 8 shows that the proposed model uses a reliable IoT network in the healthcare decision support system. The first distinct element is the proposed monitor system with the monitor of human healthcare measures and environmental parameters. The second element is the enhanced data communication between the health monitoring device and the decision support mobile application, with improved bandwidth and reduced latency. The next distinct element is the application of O-TML to the collected dataset using the TensorFlowLite Python library and Edge Impulse software tool. Then, the model is evaluated for performance analysis and trustability analysis using various evaluation metrics and XAI derivation for the proposed model. The recent research that uses fog-level applications uses both cloud and fog computing as the data communication medium. Otherwise, the research that only analyses the sensor data with machine learning models uses only the cloud as communication technology. In these cases, the data is visualized for the evaluation metrics alone. The research with feature engineering uses data visualization through tools such as Plotly and other Python libraries. Our research additionally visualizes data through the SHAP XAI tool for feature ranking and feature importance over the proposed model. The research on the use of TinyML with healthcare decision support systems is very minimal. The paper taken for comparison [10] does not use fog-level communication. Also, there is no alert management considered. The current research with fog decision support system applies machine learning models such as SVM, RF, J48Graft decision tree, and federated learning method. However, the data directly taken from IoT sensors might be very little for training purposes. Hence, machine learning models


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

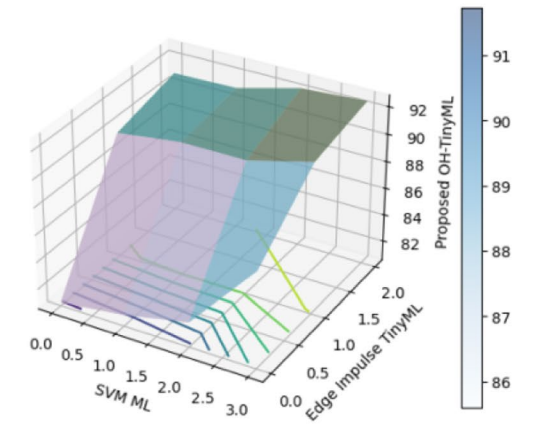
(a)


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

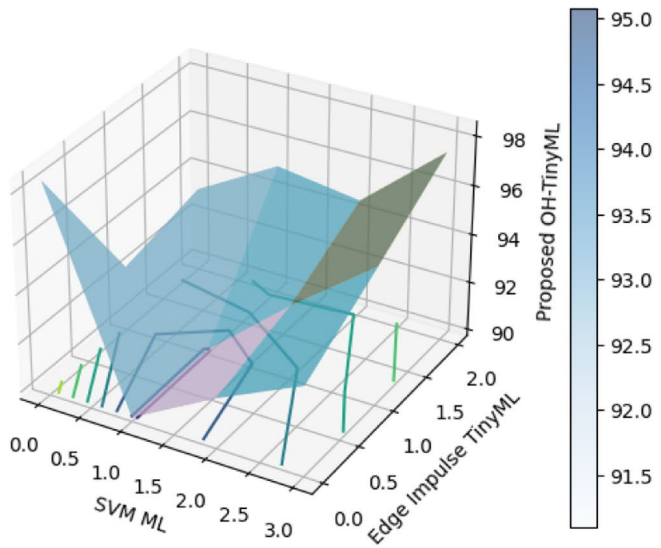
(b)


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

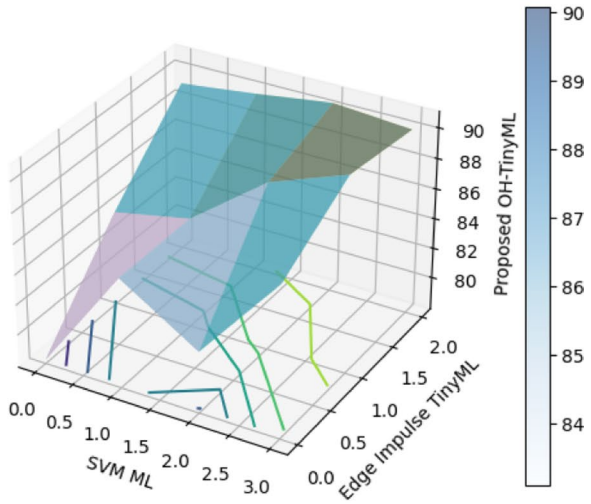
(c)


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

(d)


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

(e)


SVM ML
Edge Impulse TinyML
Proposed OH-TinyML

(f)

◂**Fig. 14** Comparative study of TinyML and ML model training and testing performance based on: **a** training precision, **b** testing precision, **c** training recall, **d** testing recall, **e** training F1 score, and **f** testing F1 score

Table 9 Performance comparison of the optimized TinyML model

Refs.	ML model	Precision	Recall	F1 score	Accuracy
[1]	Naive Bayes	89.6	84.8	88.6	91.8
	Random forest	92.1	92.5	91.2	94
	Fuzzy-CNN	93.4	93.6	94.4	93.9
	Proposed	94.7	95.8	95.2	95.9
[4]	MLP	73	70.22	76	77
	CNN	100	100	100	98
	Random forest	91	93.02	92	93
	Proposed	100	100	100	100
[6]	Bayesian network	82	82	82	Nil
	LCSS	85	86	85.49	Nil
	Markov predictor	87	83	84.95	Nil
	CNN	91	89	89.9	Nil
	Proposed	92	91	91.49	Nil
[5]	Cloud IoT HMS	Nil	Nil	Nil	94.9
	Cloud ML	Nil	Nil	Nil	94.5
	Proposed	Nil	Nil	Nil	96.2
[3]	Fedavg	Nil	Nil	Nil	94.1
	Edgefed	Nil	Nil	Nil	95
	Fed SDM	Nil	Nil	Nil	93.6
Our work	SVM	84	91	90.2	91.6
	RF	88.5	86.8	87.9	87.7
	DT	88.3	90.2	89.1	90.1
	Edge Impulse TinyML	88.43	91.8	90.1	90.47
	O-TML	87.6	94.3	90.1	91.3

can pose high training loss. Hence, along with the standard machine learning models, the proposed architecture applies the TinyML model using the TensorFlowLite library and Edge Impulse software tool. The trustability analysis of the proposed model assures that the model performs well in real-time scenarios. The current reliability test on IoT fog-level data is based on feature ranking and evaluation metrics such as accuracy and loss. Our work uses the SHAP XAI tool for reliability analysis, which examines feature importance in building the model and calculates Wasserstein distance between the two distinct features.

5.2.2 Comparative Analysis of Model Response Time with Current Research

The response time [30] of a fog-based decision support system can vary depending on several factors, including the system's design, network latency, computational capabilities of fog nodes, and the communication infrastructure. However, the primary advantage of fog computing in IoT systems is its ability to provide faster response times compared to cloud-based solutions by processing data closer to the network's edge. Figure 12 compares the response time of the proposed fog-based decision support system with the current research. With the application of the proposed mLZW data compression technique, the average bandwidth of the proposed method is 246.66 bps for fog-level communication and 52.11 bps for cloud-level communication. Since the fog layer bandwidth is very high, the response time of the proposed system is much less compared to the current research. Additionally, the usage of fog communication provides priorities over data. The proposed system uses only a few critical features such as heart rate, human body temperature, oxygen level, environmental air quality index, etc. Hence, the feature with a higher ranking receives a quicker response time. Since the proposed system is time sensitive, the less the response time, the higher is the system efficiency.

5.2.3 Comparative Analysis of Model Packet Loss with Standard Machine Learning Models

The ML model loss is the difference value between the actual predicted value and the expected value. The categorical loss function is used over the standard classification ML models to calculate the log loss. For N number of samples, y_{ij} predicted probabilities:

$$\text{Categorical Cross-Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}). \quad (28)$$

The predicted probability is 1 when it matches the expected value and 0 when there is a vast deviation. The log value of the predicted probability is reduced to provide maximum likelihood. From Fig. 13a, the training loss of the TinyML model using the TensorFlowLite library is slightly higher when compared to the model trained using the Edge Impulse tool. However, when compared to Fig. 13b, the categorical loss function of the other ML models such as random forest, SVM, and decision tree, the TinyML model is trained with less loss. This shows that for datasets with fewer entities, the TinyML model provides less loss during training and testing the model compared to the standard ML models. The lesser the loss, the higher is the accuracy.

Table 10 Overall comparison of the proposed framework with current approaches

Refs.	ML/DL algorithm	Bandwidth (kbps)	Accuracy (%)	Power consumption (mW)	Latency (ms)	Memory consumption (KB)	Includes model trustability tests
[8]	Recurrent neural network	NA	61	22.25	44.5	141	No
[20]	K-nearest neighbors, decision tree, random forest, AdaBoost, SVM	NA	93.58–97.65	308	39	400–500	No
[9]	Convolutional neural network	NA	78.40	55.73	34	195.6	No
[10]	Random forest	NA	93	205	NA	128	
[11]	Deep neural network	NA	88%	100–200	75	14,000	Enhanced ultrasound contrast imaging
[12]	Random forest, SVM, logistic regression	NA	69–73.08	100–300	10–200	50–200	No
[13]	Ensemble techniques	20	84	100–200	14–30	50–200	No
Our work	Random forest, SVM, decision tree, logistic regression, TinyML	18	92.60	100–200	27	max 150	MCC, feature dependency, Wasserstein distance

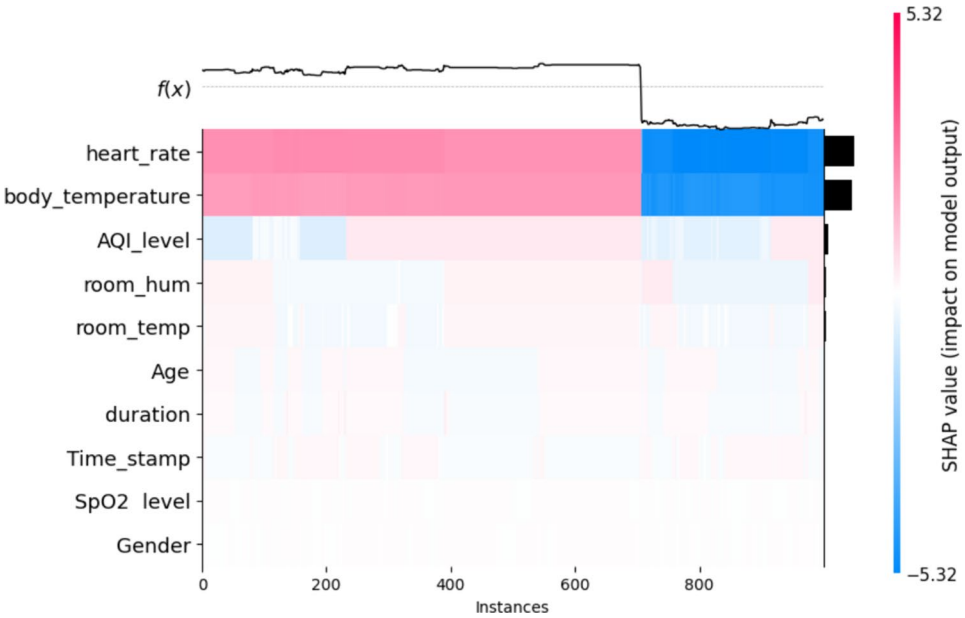


Fig. 15 SHAP XAI feature impact heat map

From Table 7 and Fig. 10, it is evident that TinyML provides higher accuracy. Hence, the loss function proves that the proposed model is efficient compared to current research.

5.2.4 Comparative Analysis of Model Performance with Standard Machine Learning Models

The proposed methodology uses the O-TML model and Edge Impulse TinyML software, an end-to-end workflow model for collecting, labeling, preprocessing, training, and deploying machine learning models on resource-constrained devices. The dataset with imbalanced values poses overfitting or underfitting issues. From Fig. 14, we observe that there is not much difference between the proposed model training performance and the testing performance. Also, the underlying features of the dataset are well learned by the proposed model. This indicates that there is a generalization in the unseen data for the generated dataset. When the model is away from overfitting and underfitting issues, in intern proves that it is reliable and generalized.

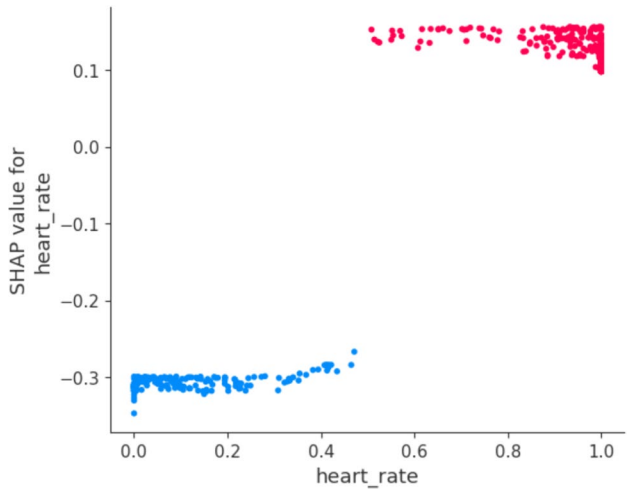


Fig. 16 SHAP XAI heart rate dependency plot

5.2.5 Comparative Analysis of Model Performance with Current Research

From Tables 8, 9, and 10, it is evident that the proposed O-TML outperforms the current algorithms in the F1 score. Recall measures the ability of a system to retrieve all relevant items or data points from a given set. The high recall value of the proposed model ensures that the system retrieves a large proportion of relevant items from the generated dataset. Also, the comparative table proves that the used tinyML model performs better for the generated dataset compared to the other standard machine learning models and current algorithms.

Table 11 Statistical comparative study of the proposed model

ML model	Sensitivity	Specificity	MCC value
SVM	0.92	0.89	0.81
Decision tree	0.91	0.87	0.78
Random forest	0.88	0.84	0.71
O-TML	0.94	0.9	0.84

6 Trustability Analysis of the Proposed Model

The proposed O-TML model for the healthcare decision support system is checked for reliability through various metrics such as model specificity and sensitivity; model statistical analysis; SHAP XAI feature importance analysis; and Features Wasserstein distance calculation.

6.1 SHAP XAI Feature Importance Analysis

The SHAP XAI feature ranking applied to the generated dataset states that the heart rate collected from the critical

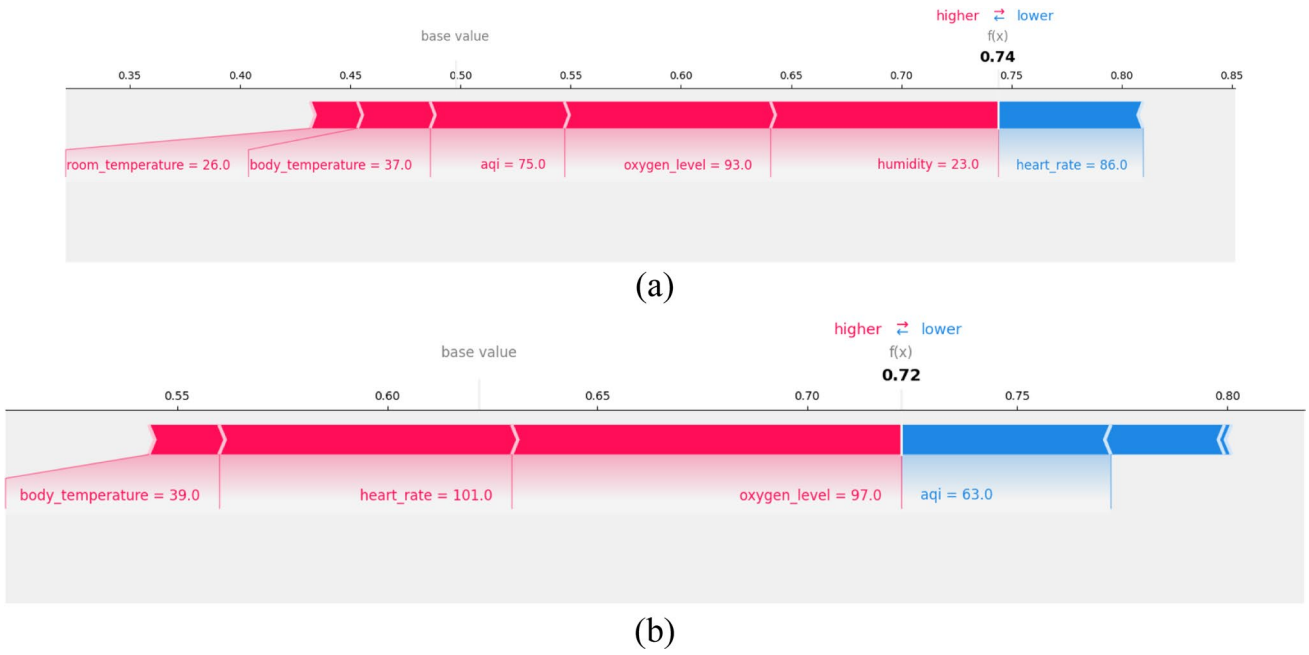


Fig. 17 SHAP explanation of: **a** normal health parameters and **b** critical health parameters

patient provides higher feature importance in the model generation. The main attribute of XAI is transparency to predict the impact of individual features on building the ML model. Figure 15 shows that heart rate, body temperature, and AQI level play important roles in machine learning health abnormality detection. The same set of features are depicted in the DNN feature extraction. This proves that the proposed model is reliable. The same results are achieved in the external XAI environment and the proposed algorithm.

Figure 16 analyzes the dependency of heart rate over the other SHAP values for individual instances. This scatter plot displays the SHAP values for the heart rate feature in a machine learning model. Each point represents an instance, with the x-axis showing the heart rate values and the y-axis showing the SHAP values. Red points indicate a positive impact on the model's prediction, while blue points indicate a negative impact. The plot provides a deeper impact of the heart rate over the model performance. A change in the heart rate provides a critical change in building the model. These plots give viewers an easy-to-understand visual aid that helps them recognize trends, comprehend non-linear relationships, and learn more about specific occurrences. By elucidating the decision logic, the combination of quantitative metrics and qualitative interpretations improves transparency, facilitates model modification, and builds confidence. All things considered, SHAP dependency charts enable users to understand and verify model behavior, promoting responsible AI deployment and well-informed decision-making.

From Fig. 17, we understand that examining SHAP force plots for heart rates that are normal and pathological offers important insights into how particular features affect model predictions in different health conditions. For a given heart rate, the force plot visualizations show the contribution of different features to the divergence from the average output of the model at each step. 0.74 and 0.72 SHAP values indicate that the selective features such as heart rate, body temperature, oxygen level, room temperature, room humidity, and AQI level contribute much toward the model output. The higher value proves that the proposed model has a higher prediction rate.

6.2 Wasserstein Distance Analysis

The Wasserstein distance indicates that the SHAP value distributions under comparison have a negligible difference. This can be understood as an indicator that the proposed model is consistent with the dependency of each feature over the other feature instances. For two distinct features, X and Y in the collected dataset,

$$\text{Wasserstein distance} = W_{\text{p}}(X,Y) = \left(\inf_{\gamma \in \Gamma(X,Y)} \int_{\mathcal{X}\times\mathcal{Y}} d(x,y)^p, \mathrm{d}\gamma(x,y) \right)^{\frac{1}{p}} \tag{29}$$

$$W_{\text{p},\epsilon}(X,Y) = \left(\inf_{\gamma \in \Gamma(X,Y)} \int_{\mathcal{X}\times\mathcal{Y}} d(x,y)^p \mathrm{d}\gamma(x,y) - \frac{1}{\epsilon} H(\gamma) \right), \tag{30}$$

where $d(x, y)$ provides the dissimilarity between the features and $\gamma(x, y)$ provides the set of distribution with the two features. $W_{\text{p},\epsilon}(X, Y)$ provides the regularized Wasserstein distance for the given dataset features. The calculated distance provides the relevance between the two features. The Wasserstein distance of the proposed model is 0.0011148.

The higher the Wasserstein distance, the more is the feature discrepancy. The proposed model provides a lesser distance, which means that the features relatively provide similar distribution. This proves that the proposed model is reliable and robust.

6.3 Statistical Analysis

Sensitivity measures the ratio of actual positive instances in the dataset to the number of true positive predictions [15]. The sensitivity value of the proposed O-TML model is 0.94. This high value says the model is highly sensitive to the generated dataset and has a critically low false negative rate.

Negative Predictive Value, or NPV is derived by dividing the total number of true negative results by the sum of true negatives and false negatives and provides information on how well a diagnostic test excludes ailments. The 0.946 NPV of our work shows that a negative test result accurately reflects the absence of the false condition.

In machine learning, determining the validity of model predictions requires managing the false discovery rate (FDR). FDR is the percentage of false positive identities among all positive identities produced by a test or technique. The 0.098 FDR value of our proposed system shows that we minimize the occurrence of false positives and enhance the reliability of positive predictions by managing the imbalance between precision and recall by establishing suitable thresholds (Table 11).

The ratio of accurately predicted negative cases, or accurate negative predictions, to the total number of real negative instances is known as specificity. It emphasizes the model's capacity to prevent false positives and serves as a supplement to recall.

$$\text{Specificity} = \text{(true negatives)}/\text{(true negatives + false positives)}. \tag{31}$$

The specificity of our model is 0.90, which shows that it is good at correctly identifying negative instances and avoiding false positives. Also, the model has a low tendency to classify positive instances as negative incorrectly.

The MCC, or Matthews correlation coefficient: The MCC metric is employed to evaluate the efficacy of binary classification models. To offer a fair assessment of the model's performance, it considers true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

$$\begin{aligned} \mathrm{MCC} = (\mathrm{TP} * \mathrm{TN} - \mathrm{FP} * \mathrm{FN})/\mathrm{sqrt}((\mathrm{TP} + \mathrm{FP}) \\ * (\mathrm{TP} + \mathrm{FN}) * (\mathrm{TN} + \mathrm{FP}) * (\mathrm{TN} + \mathrm{FN})). \end{aligned} \tag{32}$$

The model is trustable, as proven by the MCC value approaching 1 in Table 9. The model performs reasonably well and has a good balance of the predicted values, according to the 0.848 MCC score.

7 Conclusion

This study introduces an Optimized Tiny Machine Learning (TinyML) and Explainable AI (XAI) binary classification model tailored for trustable and energy-efficient healthcare decision support systems in fog-enabled IoT networks. The incorporation of the innovative mLZW data compression technique and fog computing significantly enhances data communication efficiency, reduces response times, and optimizes bandwidth usage. The proposed TinyML model, achieving an impressive F1 score of 0.93 for health abnormalities detection, outperforms traditional ML models, demonstrating its robustness and effectiveness. The integration of the SHAP XAI algorithm enhances model transparency and trustworthiness by providing valuable insights into feature importance and dependency. These advancements collectively address critical challenges in remote health monitoring, offering a robust, trustworthy, and energy-aware solution for modern healthcare needs. However, the proposed model does not include attack packet analysis through the network.

For future enhancements, further research could explore the analysis of network attack packets, and the integration of additional advanced data compression techniques to further optimize communication efficiency. Additionally, expanding the dataset to include more diverse and larger real-time healthcare records could enhance the model's generalizability and accuracy. Investigating the potential for incorporating edge AI capabilities alongside fog computing could provide even more rapid and localized decision support. Lastly, ensuring the system's adaptability to various healthcare environments and its scalability to support a broad range of health monitoring applications will be essential for widespread adoption and effectiveness.

Author Contributions AR: framework methodology design, data collection, and writing of the manuscript. KS: review of the manuscript, editing, and supervision.

Funding The authors received no specific funding for this study.

Data Availability The dataset collected and stored in the cloud platform is available on request from the corresponding author.

Declarations

Conflict of Interest The authors declare no competing interests.

Ethics Approval Not applicable.

References

1. Bhatia, M., Kumari, S.: A novel IoT–fog–cloud-based healthcare system for monitoring and preventing encephalitis. Cogn. Comput.. Comput. **14**(5), 1609–1626 (2022)
2. Singh, P., Kaur, R.: An integrated fog and artificial intelligence smart health framework to predict and prevent COVID-19. Glob. Transit. **2**, 283–292 (2020)
3. Rajagopal, S.M., Supriya, M., Buyya, R.: Fed SDM: federated learning based smart decision-making module for ECG data in IoT integrated Edge–Fog–Cloud computing environments. Internet of Things **22**, 100784 (2023)
4. Devarajan, M., Subramaniyaswamy, V., Vijayakumar, V., Ravi, L.: Fog-assisted personalized healthcare-support system for remote patients with diabetes. J. Ambient. Intell. Humaniz. Comput.Intell. Humaniz. Comput. **10**, 3747–3760 (2019)
5. Tripathy, S.S., Rath, M., Tripathy, N., Roy, D.S., Francis, J.S.A., Bebortta, S.: An intelligent health care system in fog platform with optimized performance. Sustainability **15**(3), 1862 (2023)
6. Alowish, M., Shiraishi, Y., Mohri, M., Morii, M.: Three-layered architecture for driver behavior analysis and personalized assistance with alert message dissemination in 5G envisioned fog-IOC. Future Internet **14**(1), 12 (2021)
7. Sood, S.K., Mahajan, I.: IoT-fog-based healthcare framework to identify and control hypertension attack. IEEE Internet Things J. **6**(2), 1920–1927 (2018)
8. Gokul, H., Suresh, P., Vignesh, B.H., Kumaar, R.P., Vijayaraghavan, V.: Gait recovery system for parkinson's disease using machine learning on embedded platforms. In: 2020 IEEE International Systems Conference (SysCon), pp. 1–8. IEEE, New York (2020)

9. Faraone, A., Delgado-Gonzalo, R.: Convolutional-recurrent neural networks on low-power wearable platforms for cardiac arrhythmia detection. In: 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 153–157. IEEE, New York (2020)
10. Zanetti, R., Aminifar, A., Atienza, D.: Robust epileptic seizure detection on wearable systems with reduced false-alarm rate. In: 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pp. 4248–4251. IEEE, New York (2020)
11. Căleanu, C.D., Sîrbu, C.L., Simion, G.: Deep neural architectures for contrast enhanced ultrasound (CEUS) focal liver lesions automated diagnosis. Sensors **21**(12), 4126 (2021)
12. Ayata, D., Yaslan, Y., Kamasak, M.E.: Emotion recognition from multimodal physiological signals for emotion aware healthcare systems. J Med. Biol. Eng. **40**, 149–157 (2020)
13. Tuli, S., Basumatary, N., Gill, S.S., Kahani, M., Arya, R.C., Wander, G.S., Buyya, R.: HealthFog: an ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments. Futur. Gener. Comput. Syst.. Gener. Comput. Syst. **104**, 187–200 (2020)
14. Daraghmi, E.Y., Wu, M.C., Yuan, S.M.: A multilayer data processing and aggregating fog-based framework for latency-sensitive IoT services. Appl. Sci. **11**(4), 1374 (2021)
15. Hong, Y.G., Hwang, S., Seo, J., Lee, J., Park, J.: Real-time implementation of distributed beamforming for simultaneous wireless information and power transfer in interference channels. ETRI J. **43**(3), 389–399 (2021)
16. Mukherjee, A., Ghosh, S., Behere, A., Ghosh, S.K., Buyya, R.: Internet of health things (IoT) for personalized health care using the integrated Edge–Fog–Cloud network. J. Ambient. Intell. Humaniz. Comput.Intell. Humaniz. Comput. **12**(1), 943–959 (2021)
17. Oğur, N.B., Al-Hubaishi, M., Çeken, C.: IoT data analytics architecture for smart healthcare using RFID and WSN. ETRI J. **44**(1), 135–146 (2022)
18. Ramraj, S., Arthi, R., Murugan, S., Julie, M. S.: Topic categorization of tamil news articles using pretrained word2vec embeddings with convolutional neural network. In: 2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE), pp. 1–4. IEEE, New York (2020)
19. Singla, J., Mahajan, R., Bagai, D.: An energy-efficient technique for mobile-wireless-sensor-network-based IoT. ETRI J. **44**(3), 389–399 (2022)
20. Fedorov, I., Stamenovic, M., Jensen, C., Yang, L.C., Mandell, A., Gan, Y., et al.: TinyLSTMs: efficient neural speech enhancement for hearing aids. arXiv preprint arXiv:2005.11138 (2020)
21. Arthi, R., Krishnaveni, S.: Design and development of IoT test bed with DDoS attack for cybersecurity research. In: 2021 3rd International Conference on Signal Processing and Communication (ICPSC), pp. 586–590. IEEE, New York (2021)
22. Ahmad, I., Abdullah, S., Ahmed, A.: IoT-fog-based healthcare 4.0 system using blockchain technology. J. Supercomput.Supercomput. **79**(4), 3999–4020 (2023)
23. Roy, I., Mitra, R., Rahimi, N., Gupta, B.: Efficient non-DHTBased RC-based architecture for fog computing in healthcare 4.0. IoT **4**(2), 131–149 (2023)
24. Spasojevic, N., Vasilj, I., Hrabac, B., Celik, D.: Rural-urban differences in health care quality assessment. Mater. Sociomed. **27**(6), 409 (2015)
25. Selvaraj, S., Karan, A., Srivastava, S., Bhan, N., Mukhopadhyay, I., World Health Organization: India: health system review. In: Health Systems in Transition, vol. 11, no. 1. World Health Organization, Geneva (2022)
26. Baucas, M.J., Spachos, P., Plataniotis, K.N.: Federated learning and blockchain-enabled fog-IoT platform for wearables in predictive healthcare. IEEE Trans. Comput. Soc. Syst. **10**(4), 1732–1741 (2023)
27. Ali, H.M., Bomgni, A.B., Bukhari, S.A.C., Hameed, T., Liu, J.: Power-aware fog supported IoT network for healthcare infrastructure using swarm intelligence-based algorithms. Mobile Netw. Appl. **28**(2), 824–838 (2023)
28. Srirama, S.N.: A decade of research in fog computing: relevance, challenges, and future directions. Softw.: Pract. Exp. **54**(1), 3–23 (2024)
29. Dharani, A., Kumar, S.A., Patil, P.N.: Object detection at the edge using TinyML models. SN Comput. Sci. **5**(1), 11 (2023)
30. Arora, N., Kaur, P.D.: GeoCredit: a novel fog-assisted IOT-based framework for credit risk assessment with behavior scoring and geodemographic analysis. J. Ambient. Intell. Humaniz. Comput. Intell. Humaniz. Comput. **14**(8), 10363–10387 (2023)

Authors and Affiliations

R. Arthi[1] · S. Krishnaveni[2]

✉ R. Arthi
ar7862@srmist.edu.in

✉ S. Krishnaveni
krishnas4@srmist.edu.in

[1] Department of Computational Intelligence, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India

[2] Department of Computational Intelligence, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India