Journal of King Saud University – Computer and Information Sciences 34 (2022) 7126–7137

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Artificial intelligence-based fault prediction framework for WBAN

Mamoun Awad [a], Farag Sallabi [b,*], Khaled Shuaib [c], Faisal Naeem [d]

[a] Department of Computer Science and Software Engineering, UAE University, United Arab Emirates
[b] Department of Computer and Network Engineering, UAE University, United Arab Emirates
[c] Department of Information Systems and Security, UAE University, United Arab Emirates
[d] Department of Electrical Engineering, National University of Computer and Emerging Sciences, Pakistan

ARTICLE INFO

Article history:
Received 23 February 2021
Revised 22 August 2021
Accepted 14 September 2021
Available online 22 September 2021

Keywords:
WBAN
Machine learning
Fault management
Fault prediction
Sensor health packets

ABSTRACT

Wireless Body Area Networks (WBAN) can provide continuous monitoring of patients' health. Such monitoring can be a decisive factor in health and death situations. Fault management in WBANs is a key reliability component to make it socially acceptable and to overcome pertained challenges such as unpredicted faults, massive data streaming, and detection accuracy. Failures in fault detection due to hardware, software, and network issues may put human lives at risk. This paper focuses on detecting and predicting faults in sensors in the context of a WBAN. A framework is proposed to manage AI-based prediction models and fault detection using thresholds where four Machine learning techniques: Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Support Vector Machines (SVM), and Decision Trees (DT), are used. The framework also provides alarm notifications, prediction model deployment, version control, and sensing node profiling. As a proof of concept, a fault management prototype is implemented and validated. The prototype classifies faults, manages automation of sensing node profiling, training, and validation of new models. The obtained experimental results show an accuracy greater than 96% for detecting faults with an inferior false alarm rate.

1. Introduction

The vast advances in wireless communications and semiconductor technologies led to the emergence of smart sensors and sensor networks. Smart sensors support enormous medical and non-medical applications. Wireless Body Area Network (WBAN) is a special purpose sensor network that can operate autonomously to connect medical devices implanted or attached to a human body. Adopting WBAN devices can provide personalized online care and dramatically decrease healthcare services' cost (Salayma et al., 2017).

WBAN should be reliable and effective. Reliability can be addressed by three core characteristics, fault tolerance, Quality of Service (QoS), and security (Academy, 2013). To enable seamless WBAN operations, capabilities to handle fault tolerance and recovery must be integrated into the design and architecture of such networks. For medical applications, strict QoS metrics such as packet delay, communication errors, and data loss should be maintained at an acceptable level to avoid serious consequences. Securing WBAN is a crucial factor in any successful implementation, and it has been previously addressed by researchers (Barakah and Ammad-uddin, 2012).

WBAN systems consist of hardware and software components; hence, software and hardware faults need to be addressed. Previous research focused on hardware faults in sensors, such as irregularity readings, hardware faults, and battery faults. However, software faults are also critical, including communications faults, model malformation, software bugs, inadequate validation, and induced human errors.

WBANs components that record or monitor patients' vitals may differ in hardware, software, data format, and communication range. For example, sensors designed for similar functions may have different specifications, such as those used for EEG and EMG signals. New WBAN components with improved capabilities are emerging continuously. Thus, the healthcare industry needs to be aware of the latest technologies to utilize the best fit for the intended applications. Component profiling might exhibit the right solution in which specifications from different manufacturers

* Corresponding author.
E-mail address: f.sallabi@uaeu.ac.ae (F. Sallabi).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.09.017
1319-1578/

are maintained and updated regularly. Such profiling will help in the deployment of WBAN systems based on best practices for healthcare applications.

In the era of Big Data (BD) and Artificial Intelligence (AI), fault-management systems in WBAN can be equipped with smart/intelligent backend models to help predict and detect faults. The literature is abundant with various machine learning (ML) and AI techniques used to support such systems (Sharma et al., 2010). However, due to variant trends in data, coping with new learning techniques still represents a challenge. When data streams change over time, patterns and trends change; hence, the training of used ML models on new data must be updated continuously while considering new emerging models.

This paper focuses on the design and implementation of fault management schemes in the context of a deployed WBAN where failures cannot be tolerated in life-critical healthcare systems. The paper addresses several main aspects: fault management, sensor packets management, sensor deployment and profiling, and accuracy of AI-based used models. Specifically, an AI-based framework that incorporates different ML techniques to detect faults is being proposed. The main contributions of this work are as follows:

1. Propose a fault-management framework that incorporates AI/ML models for WBAN applications.
2. Implement the ML process to support the fault tolerance framework. The process includes the automation of training, validation, and deployment of prediction models.
3. Generate empirical results for the four considered prediction models: ANN, DNN, SVM, and Decision Trees.
4. Develop a prototype for a Personal Digital Assistant (PDA) application on the patient's end and a training and data management prototype for the data repository at the backend.

The rest of the paper is organized as follows. In Section 2, we present the current state of the art in fault-management systems and applications. Section 3 provides background on fault types and their classification. In Section 4, we present our AI methodology and framework. Section 5 presents the empirical results and the developed prototype. Finally, we conclude the paper and state our future research directions in Section 6.

2. Related work

The literature is rich in fault management research that focuses on fault diagnosis, detection, and tolerance (Gao et al., 2015). However, in this section, we focus on work conducted in the area of fault management in the Internet of Things (IoT) and WBAN systems, which is still in its early stages with several open research problems (Salayma et al., 2017; Al-Turjman and Baali, 2019).

Zhou et al. (2015) addressed fault management in an IoT network when sensors with different functions were used. They applied regression analysis to approximate virtual services and developed a sensing device adaptation scheme for fault-tolerant IoT networks. However, the proposed approach does not apply to smart healthcare systems. The authors in (Gia et al., 2015) presented an IoT-based architecture to support scalability and fault tolerance for healthcare systems. They deployed a redundant sink node that backups the main sink node in communication or node failures. The authors claimed that their fault tolerance approach covered many fault situations. However, their approach handled the problem of faults reactively. Additionally, they did not report any fault detection and recovery performance rate.

Yang et al. (2015) proposed a Data Fault Detection mechanism in Medical sensor networks (DFD-M). They used a dynamic-local outlier factor (D-LOF) algorithm to identify outlying sensed data vectors. A linear regression model was used based on trapezoidal fuzzy numbers to predict which biological readings in the outlying data vector are suspected to be faulty. The authors assumed time and space correlation between readings; hence, the method does not work if the patient is connected to one sensor. A fault diagnosis model based on Deep Neural Networks (DNN) was proposed in (Sharma et al., 2010). Temporal coherence of raw time-series sensor data without feature selection and signal processing was used to train the DNN. Raw data was segmented and fed to the DNN. However, the conducted experiments did not include medical data, and the output of the DNN model was binary, i.e., a prediction was either faulty or normal.

The authors in (Zhang et al., 2017) proposed a method to detect faulty sensors by building a logistical regression model on the sink nodes using sensor nodes data. The model is deployed on the sensors to predict incoming faulty readings. The authors considered binary outcomes and employed a threshold to determine faulty versus non-faulty readings. However, to save battery life faulty, detected readings were not forwarded to the sink node. Additionally, the reported prediction rate was relatively low, which made it inapplicable for a WBAN environment.

Sharma et al. (2010) worked on different methods of detecting sensor faults. They used rule-based, estimation, time-series-analysis-based, and learning-based methods. They concluded that rule-based methods could be highly accurate, depending on the choice of parameters used. In addition, learning methods could classify and detect faults; however, training was shown to be burdensome as estimation methods could not reliably classify faults.

Opportunities and challenges of healthcare monitoring and management in the IoT paradigm with cloud-based processing were presented in (Hassanalieragh, 2015). The paper reviewed the current state and projected future directions for integrating remote health monitoring technologies into the clinical practice of medicine. They also highlighted several challenges in sensing, analytics, and visualization, which needed to be addressed.

The paper in (Li et al., 2013) proposed a layered fault management scheme for end-to-end transmission for heterogeneous IoT networks. The authors concluded that end-to-end connectivity involves different network sections with different performance and connectivity requirements. Sectional monitoring was a better choice to apply as a suitable management technique for each network section. They handled fault detection and location in a distributed manner, while fault recovery was handled in a centralized manner. A visual monitoring system for fault tolerance (VMSFT) was proposed in (Jeong et al., 2014) and (Mahapatro, 2011) to monitor all sensors within a WBAN and respond proactively to sensor failure. VMSFT was based on cloud computing service, IaaS (infrastructure as a service); however, it was restricted to local geographic areas.

The authors of (Yi and Cai, 2018) proposed a framework to manage delay-sensitive medical packets beyond WBAN. The authors focused on the random arrival of sensed medical packets at each WBAN-gateway and the medical-grade quality of service (mQoS) component by considering the behavior of smart gateways. The papers in (Al-Turjman and Baali, 2019; Sawand et al., 2015; Hartmann et al., 2019) surveyed different approaches to designing eHealth monitoring focusing on cost, usability, security, and privacy. The authors presented in detail different components of the monitoring lifecycle and essential service components. The work did not discuss fault detection and recovery or incorporate data collected in prediction or learning. In (Liao et al., 2018), the authors proposed and evaluated a wireless relay-enabled task offloading mechanism consisting of a network and a computation models. Wang *et al.* (Wang et al., 2020) proposed a multi-layer edge computing-based framework to detect emergencies securely and with low latency. The framework captured a universal data

model, a fine-grained detection model with RNN on the cloud side, and a pre-defined threshold-based detection on the edge node. In this proposed framework, practicality might have been an issue due to the required training on the edge and the cloud side. Moreover, the obtained results were only compared with SVM before claiming the model to be accurate and responsive. In (Li et al., 2008), a monitoring system of greenhouse environments was deployed to monitor equipment and provide services through hand-held devices such as PDA. The work focused on WSN technology and management strategies in a cost-effective nature.

In recent work, authors of (Tavera et al., 2021) provided a systematic review of WBAN, including applications, trends, protocols, and standards that must be considered. The authors of (Mehmood et al., 2020) and (Peng et al., 2017) adopted a cooperative communication and network coding strategy to reduce channel impairment and body fading effect. The proposed solution did not include any AI technique, though they claimed success in improving energy consumption, bit rate error, and faults.

Our work is similar to the work in (Jeong et al., 2014; Yi and Cai, 2018; Wang et al., 2020; Li et al., 2008; Zhang et al., 2017). However, we consider biomedical and sensor health packets; we apply machine learning techniques, prediction model management, sensor profiling, and threshold checking. Table 1 presents a comparison between our work and others based on different criteria and focuses.

3. Fault management framework

This section presents our proposed fault management framework based on deep/machine learning, as depicted in Fig. 1. The framework comprises hardware and software components distributed on two main sites, namely the patient's site and the data center site. We assume that an insurance company is a supporting entity for health monitoring of patients, data acquisition, and logistics.

The patient may exist in different places with different contexts and network connections, as shown in Fig. 2. The physiological data goes to an Electronic Health Record (EHR) repository, while the management data goes to a Content Management Database (CMDB) (Sallabi et al., 2018; Al Thawadi et al., 2019). As was discussed earlier, fault management is concerned with faults in the network, devices, and software. Effective fault management is critical to ensure minimal to no disruption of healthcare recipients' and providers' services.

In this framework, we considered different software and hardware faults. Table 2 presents and defines each fault and its implications.

3.1. Data center subsystem

Our framework uses different data sources such as patients' sites, caregiver sites, networking data, and prediction model data. Data from patients' sites are accumulated, persisted, and used to generate AI/Deep Learning models to build a fault tolerance system at the Data Center. Sensor data in WBAN can be of three main types: sensor raw data packets, sensor health packets, and route update packets. Sensor raw data packets, such as temperature, ECG, SpO2, carry sensed data to the gateway. Sensor health packets include data on how well the sensor performs in the network regarding radio traffic, battery voltage, Received Signal Strength Indicator (RSSI), dropped packets, etc. Routing update packets include information on how to route packets to the gateway. For our proposed framework, we only collected and analyzed the first two kinds of sensor data.

Fig. 1 presents the Data Center subsystem with four components: Data Processing, Learning and Evaluation, Data Source Component, and Model Version Control (MVC). We focus on two aspects of the fault tolerance system: the training process to predict faults and the management and deployment process for the new updates/releases of prediction models.

3.1.1. Learning and evaluation component

The training process is performed offline in the data center. The task involves mining accumulated data from different sources that include individual sensors and sites. The data is pre-processed, partitioned, filtered, and might be disguised for security and privacy purposes. After the data is cleaned, ML algorithms are applied. The learning task is an iterative process of two significant steps, namely, training and evaluation. A data scientist can use different ML techniques to produce a fault tolerance prediction (FTP) model in training. Fig. 3 (A) presents an FTP model captured through Unified Modeling Language (UML) notation. The model has a simple interface to predict and retrieve metadata and readable labels. The FTP model is evaluated to verify its reliability, accuracy, and accomplice with a standard format. Verified FTP models are then deployed to the MVC. Our prototype has incorporated four well-known effective techniques in learning, namely, ANN, DNN, SVM, and Decision Trees (DT). Our framework is flexible as more models can be incorporated and deployed later to enhance performance and cope with new emergent techniques.

3.1.2. Model version control

MVC is a version control component that manages different model versions and releases. The FTP model has two compartments, metadata and the prediction model. The metadata contains a version number, description, unique ID, and readable labels. The

Table 1
Comparison of our work vs. other works.

Research Reference	Topic	Fault Detection	Data Management	Machine Learning Approach	Prediction Model Management	Prototype as proof of concept
(Salayma et al., 2017; Gao et al., 2015; Al-Turjman and Baali, 2019; Hassanalieragh, 2015; Sawand et al., 2015; Hartmann et al., 2019)	Survey	N/A	N/A	N/A	N/A	N/A
(Zhou et al., 2015; Gia et al., 2015; Li et al., 2013)	IoT	Yes	No	Yes	No	No
(Sharma et al., 2010; Yang et al., 2015; Zhang et al., 2017)	WBAN	Yes	No	Yes	No	No
(Jeong et al., 2014)	WBAN	Yes	No	Yes	No	No
(Yi and Cai, 2018)	WBAN	Yes	Yes	No	No	Yes
(Liao et al., 2018; Wang et al., 2020; Tavera et al., 2021; Mehmood et al., 2020; Peng et al., 2017)	WSN	Yes	No	Yes	No	No
(Li et al., 2008)	WBAN	Yes	Yes	Yes	No	Yes
(Zhang et al., 2017)	WBAN	No	Yes	No	No	Yes
Our Work	WBAN	Yes	Yes	Yes	Yes	Yes

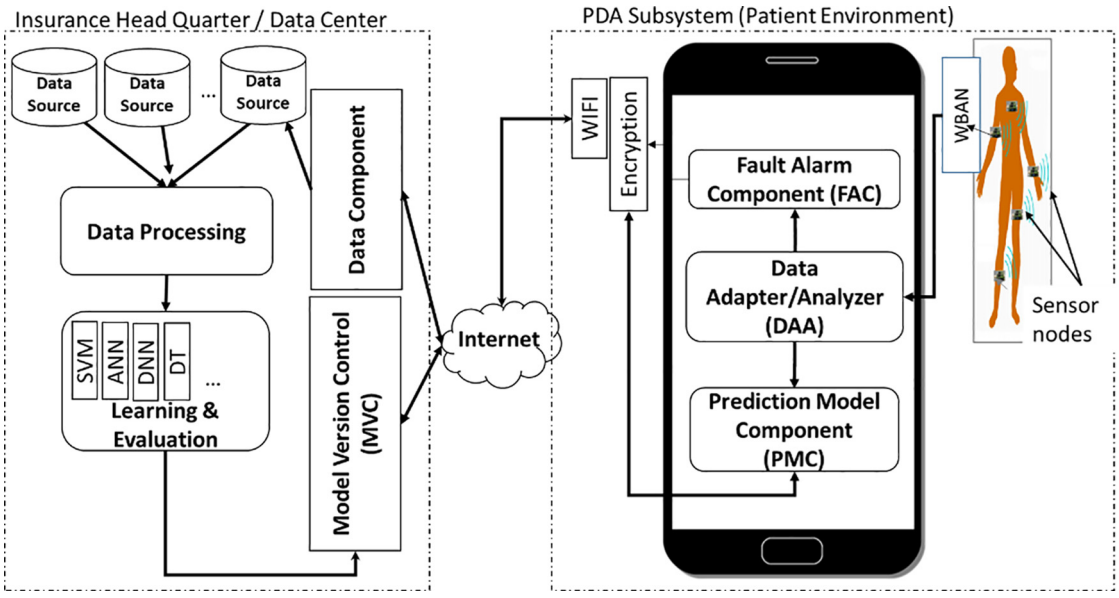

Fig. 1. Fault Management Framework.

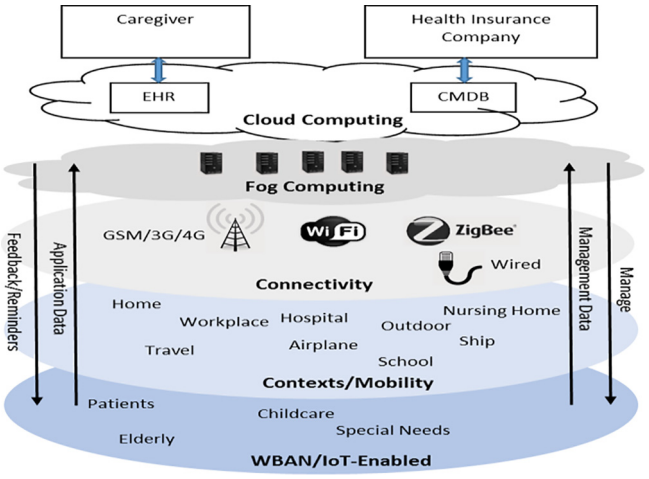

Fig. 2. Smart Healthcare Context.

prediction model is a raw data representation of the learning experience, and it is technology-specific. For example, it might be a TensorFlow model, a WEKA model, a LIBSVM model, etc.

MVC is a crucial component of the framework. Aside from managing model versions, it notifies PDA components of new updates and releases. In this framework, we consider different tasks/responses for different faults because learning one model for all faults leads to a fat model requiring long training and might suffer from low accuracy. Therefore, we have different standard models and specialized models, as well. Standard models will be used for general scenarios, while specialized models are used for emergencies and critical conditions.

Additionally, learning is an ongoing process in which retraining is performed when new data is accumulated and becomes ready for learning. The task is recurred to update the models and deploy them on production. Automating such activities poses a serious challenge to the framework's desired high level of reliability and fault-tolerant devices in a critical context that involves human lives.

Standardization of model format is another challenge due to the availability of different devices and learning techniques that can be applied. Each device may generate different data formats, and each learning algorithm can have a different model format. Currently, model formats include ProtoBuf (PB) Prototype from google and are used in TensorFlow models, Libsvm format, HDF5 format, and WEKA format. The proposed FTP model can hold various models because of the attached metadata, as shown in Fig. 3 (B). Such metadata helps the PDA to interpret, run, and predict faults correctly. In the data center, we can create different models for different faults. There are different AI techniques for different applications and contexts. We integrated four AI algorithms, namely, DT, SVM, ANN, and DNN. We have considered the three criteria in selecting these models: fault prediction context, the current state of the art, and AI emerging techniques.

Decision Tree (Yan-yan and Ying, 2015) produces an interpretable model with intuitive induction rules easily understood and calculated. SVM is a popular learning technique that searches for the optimal margin hyperplane to divide the dataset and maximize their margin. This is performed by mapping the data into a higher-dimensional space using the kernel trick (Shawe-Taylor and Cristianini, 2000). ANN and DNN are well-known for their prediction power in complex problems (Al-Turjman and Baali, 2019). ANN was inspired by biological neurons and became a very important method for prediction because of its ability to deal with uncertainty and insufficient data sets. In ANN, a two or three-layer feed-forward neural network is commonly used. However, many hidden layers (DNN) showed remarkable results in a complex problem, though training takes longer.

We used two technologies in our prototype, TensorFlow and Weka, for deep learning (DNN), SVM, and DT. Normally, based on the data complexity, prediction accuracy, and performance, the data scientist decides on the technology and the ML algorithm to be used.

3.1.3. Sensor profiles

Different medical sensors can be attached to the patient's body at the same time. Each sensor might use different technology and different data representations. For this purpose, we introduce the concept of sensor profile, which has the sensor specifications and operational thresholds. Sensors' profiles are maintained and managed by the data center as part of the Data Component. For the PDA to work correctly, sensor specifications need to be downloaded, especially when a newly deployed sensor is used or replacing a

Table 2
Hardware and software faults definitions and implications.

Fault Type	Description	Impact of fault
Outlier	A data point deviates from other observations due to noise, error, events, or malicious attacks.	It may skew the mean, variance, gradient, and other data features.
Spiked Value	The rate of change in the gradient of data over some time is greater than expected. Frequent causes include battery failure and other hardware failures or loose wires connections.	Spikes should be discarded and result in a loss of sensor data yield.
Stuck-at Zero	Sensor values experience zero variation for an unexpected length of time. Frequently the cause of this fault is a sensor hardware malfunction.	Data still holds value and can be interpreted at lower fidelity. Otherwise, discard data.
Hardware	A malfunction in the sensor hardware causes inaccurate data. This is due to environmental disturbance, short circuits, or loose wires.	Data is meaningless as the sensor is not performing as designed. It should be discarded.
Battery	Battery voltage drops to a point the sensor cannot confidently report data.	A battery failure results in useless data. The exception is if sensor behavior at low voltage gives added noise, then there may be informational value.
Environment out of Range	The environment exceeds the sensitivity range of the transducer. There may be much higher noise or a flattening of the data. It may also be a sign of improper calibration.	It still holds some information content. At a minimum, it indicates the environment exceeds the sensor sensitivity range.
Wireless connection	Connection to the gateway is lost due to interference, noise, low battery, etc.	Loss of data.
No Internet Connection	Connection to the Internet is lost due to mobility.	No connection to the backend servers.
Malformed Prediction Model	A prediction model is not conforming to the format/standard definition.	Prediction fails.
Dis-functioning prediction model	The model has a low prediction accuracy	False Positive and false-negative rates are high.

sensor with different technology. Table 3 presents the meaning of the proposed sensor profile values. Notice that the profile holds both manufacturing specifications and fault configurations, such as fault thresholds and timing. For example, ft_dead indicates the time in seconds for the PDA to wait until it sends an alert for the sensor down fault. When the data center site configurations are changed, the PDA will be notified automatically to download the new profile.

The sensors' profiles are used in our framework to detect reading faults. It was reported in (Salayma et al., 2017) that using a threshold to isolate and detect faults was not always successful because of discrepancies in hardware and other specifications. However, in this framework, we propose to automate this process by building an ML model that automatically generates samples and builds a hierarchy of thresholds for all sensors. Fig. 4 depicts the steps of building such a model. First, we randomly generate sample readings (training and validation) for each sensor based on its profile. Second, these readings are automatically labeled and passed to an ML technique. Finally, the prediction model is verified using the validation set. We generate more samples and iterate if the model's accuracy is below a threshold (<90%). Finally, the prediction model is deployed to be used by the PDA.

3.2. Patient PDA subsystem

The PDA subsystem manages the patients' environment. It receives the WBAN sensors data, analyzes health data, securely sends sensor data to the datastore, predicts faults, and notifies the patient/data center of faults. The PDA subsystem has three components: a data adapter/analyzer, fault alarm, and prediction model management. In the following subsections, we present each component in detail.

3.2.1. Data Adapter/Analyzer component

The Data Adapter/Analyzer (DAA) component is responsible for capturing WBAN sensor data. The data can be either raw traffic packets or biological data of the patient. Collected data by the PDA component is sent to the data center after being processed.

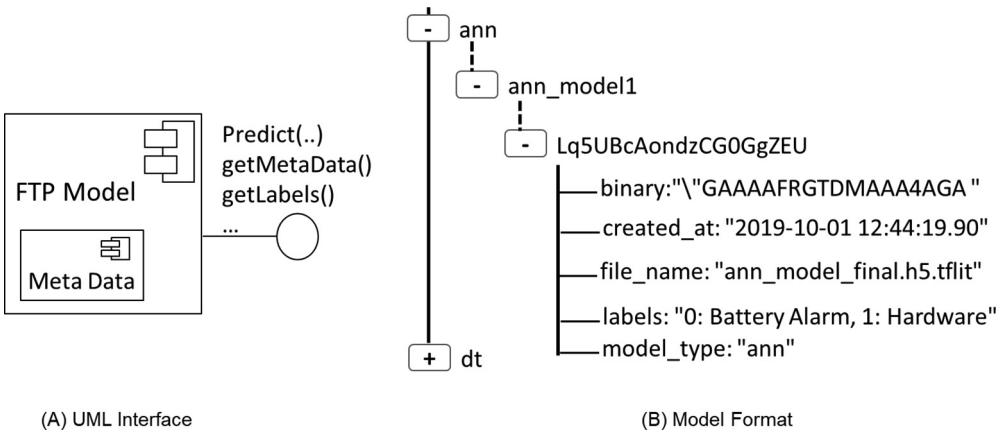

Fig. 3. FTP model structure and format, (A) interface of the FTP model in UML, (B) FTP internal details in the cloud.

Table 3
Proposed Sensor Profile Attributes Definitions.

Property	Definition
Description	General description of the sensor
Id	A unique Id of a sensor.
Model	Model description of the sensor hardware
Serial	The serial number of the sensor
Type	Type of the sensor
Ft_dead	The time in seconds indicates how long the PDA should wait until it decides a sensor down fault.
Ft_hardware	The count of consecutive hardware-fault to trigger sensor hardware fault alert, e.g., in Table 5, the PDA sends such an alert if it detects 3 consecutive faults.
Ft_low_battery	The count of consecutive low battery faults to trigger sensor low battery fault alert, e.g., in Table 5, the PDA sends such an alert if it detects 3 consecutive faults.
Ft_spike	The count of consecutive spiked sensor readings to trigger sensor spike fault alert, e.g., in Table 5, the PDA sends such an alert if it detects 3 consecutive readings higher than a max-values attribute.
Ft_stuck	The count of consecutive zero sensor readings to trigger sensor stuck fault alert, e.g., in Fig. 4, the PDA sends such an alert if it detects 3 consecutive zero readings.
Min-value	The minimum value of the sensor reading per the specifications. Values smaller than this value are considered spiked values.
Max-value	The maximum value of the sensor reading. Values higher than this value are considered a spike.

The data must be encrypted during transmission and residing at the data center for privacy and security concerns. Due to WBAN sensors' diversity, data format must follow a standard to guarantee privacy and compatibility. The Data Adapter component must be designed to accommodate seamless/smooth sensors replacement and newly emerging technologies as needed. Standards have been proposed to accomplish that, such as the ETSI TS 103 378 standards; however, an adequate, widely used standard is still needed.

Fig. 5 depicts the detailed design and the algorithm used in the prototype. Notice that DAA has four subcomponents. First, the Data Adapter, which receives raw data and standardizes it. The second is, Pre-Processor, which filters the data to extract relevant information specific to the task. Third, the Transmitter, which transmits the data to the data center. Fourth, the Fault Detector analyzes the data locally using AI and Deep learning models (saved on the PDA internally) and reports any sensor faults to the Fault Alarm component. The fault detector can predict sensor faults bypassing the pre-processed traffic to a deeply well-trained pool of models managed by the Prediction Model Component.

In addition to the prediction models, the Fault Detector checks thresholds to detect faults. In our design, we use sensor profiling to make threshold checking more practical and feasible. As explained earlier, on the patient site, a profile exists that defines each sensor's specifications, including thresholds. If a sensor is replaced or added to the patient, the PDA receives a notification from the data center and downloads the sensor's profile. The fault detection algorithm starts with the initialization of data structures. These data structures include dictionaries to track each sensor's last packet information, as seen in Table 4.

Table 5 presents the algorithm to detect faults. Lines 1–3 perform initialization and setup. At Line 4, the main loop starts infinitely to monitor sensor traffic. The algorithm waits in line 5 until a new packet arrives—lines 6–7 update data structures, which invokes the algorithm in Table 6. Lines 8–9 buffer the arriving packets and flush the buffer to the data center once full. Lines 10–12 start the asynchronous call (separate thread of execution) to separately detect each fault. For example, Line 10 starts a thread of execution to detect whether a sensor reading was beyond the min–max Range or not. The thread will execute the algorithm in Table 7 by checking the number of consecutive spiked values received for a given sensor and compare it with the profile threshold. If there is a violation of the threshold, an alert is sent to the data center, line 8 in Table 7. The thread after that waits for a given amount of time until the next check, line 11 in Table 7. Notice that separate threads can be started simultaneously for each fault or combined in one thread. As shown in Table 7, each packet can be passed to the function "predict_fault," which is designed based on fault sudden causes. However, the investigation of progressive degradation due to that might be worth exploring for future work.

In case a fault is detected, a trap is sent through a network management protocol to the data center. To avoid false alarms and unnecessary alerts, the prototype does not alert when a fault is detected. Instead, each fault has a time/count threshold that triggers the alert once reached. For example, if a dead-sensor fault is detected, the prototype does not alert unless it detects the same fault 3 times. Recall that the thread that detects faults does that repeatedly in a relatively short amount of time. This way, we can avoid unnecessary alerts due to sensors becoming blocked by physical barriers.

3.2.2. Fault alarm component

The Fault Alarm Component (FAC) notifies the data center of any sensor faults. It sends the fault notification/trap repeatedly until the issue is resolved. FAC sends fault to the patient or the attendant through voice, SMS, or both.

3.2.3. Prediction model component

The Prediction Model Component (PMC) retrieves the latest prediction models and stores them locally on the PDA. Fig. 5 shows the different sub-components of PMC. The Access sub-component retrieves the prediction models needed to monitor the patient based on the patient's profile. The Access sub-component checks

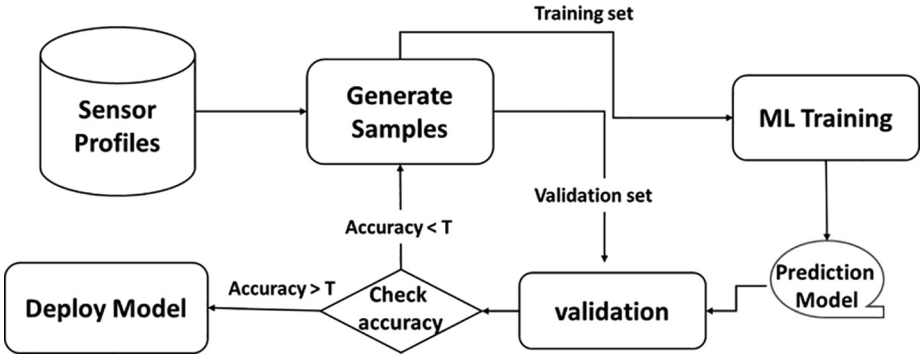

Fig. 4. Threshold Based Learning.

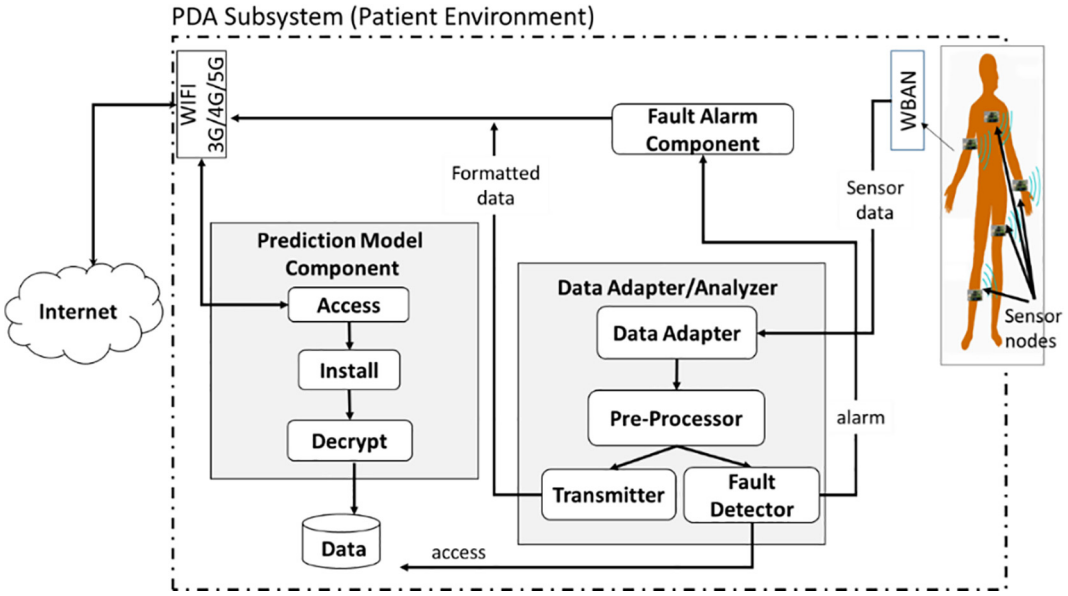

Fig. 5. PDA Subsystem in depth.

Table 4
Data structures used in the prototype.

DICTIONARY	<KEY,VALUE>	PURPOSE
Hb_dict	<Sensor id, time>	Holds the packet's last arrival time for a sensor.
Fault_dict	<sensor id, count>	Holds a number of consecutive reported faults for a sensor
Stuck_at_dict	<sensor id, time>	Holds the last time, a zero value was recorded for a sensor.
Spike_dict	<sensor id, time>	Holds the last time, a spiked value was recorded for a sensor.

for new updates on the data center and installs them locally on the PDA. For privacy reasons, prediction models are encrypted and only decrypted during installation. These prediction models are used by the Data Analyzer to monitor traffic, looking for sensor faults. Our prototype utilized built-in asynchronous listeners to automatically trigger a download of newly generated prediction models on the data center site. The prototype implemented a model validation process to verify the prediction model's proper installation and functionality before usage. Otherwise, a software alert is raised, and the PDA rolls back to the previously installed version. In addition to managing prediction models locally, the PMC manages sensor profiles that contain specifications of each sensor used on the patient's site, as was shown in Table 3.

Table 5
Fault Detection Algorithm (* means Asynchronously).

Algorithm: Fault Detection/Transmission

Input: Buffer Size N
Steps:
1. initialize data structures
2. buffer ← [] //buffer to hold packets
3. pred_models ← Read Prediction Models
4. DO-WHILE
5. Pkt ← readPacket() //read incoming packet from the WBAN
6. sensor_id ← get_sensor_id(pkt)
7. record_packet_info(pkt, sensor_id) //See Table 6
8. IF buffer is full THEN flush(buffer, data-center)
9. buffer.append(pkt)
10. Start Asynch* check_fault(spike_dict, SPIKE-Fault) //See Table 7
11. Start Asynch* check_fault(hb_dict, Sensor-Down) // See Table 7
12. Start Asynch* check_fault (ft_dict, Low-Battery) // See Table 7
13. END-DO

4. Experiments and analysis

This section presents experiments conducted in developing fault tolerance prediction models. We have three sets of experiments: The Lab-based WBAN traffic dataset, the biological dataset, and the synthesized dataset. In each subsection, we will elaborate on performance measures, the data set details, experimental setup, and results analysis.

4.1. Performance measures

In WBAN, the datasets might be imbalanced, i.e., the distribution of labels is not equally distributed. Therefore, accuracy alone is not a good performance measure to report prediction results (Fu et al., 2019). Hence, the following standard formulas were used to calculate accuracy, false alarm rate, precision, recall, and F1-score measures to assess the performance.

$$Accuracy = \frac{TP + TN}{N} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4}$$

$$FalseAlarmRate = \frac{FP}{FP + TN} \tag{5}$$

where N is the size of the dataset; TP (True-Positive) is the number of predicted positives that are positives/faults; TN (True-Negative) is the number of predicted negatives that are not positives/faults; FP (False-Positive) is the number of predicted positives that are not positive/faults; FN (False-Negative) is the number of predicted negatives that are positive/faults.

Table 6
Record Packet Information Algorithm.

Algorithm: Record-Packet-Info
Input: packet: pkt, Sensor: sensor_id Steps: 1. profile ← get_sensor_profile(sensor_id) 2. reading ← get_biological_data(pkt) 3. hb_dict[sensor_id] = current_time () 4. IF reading is not zero THEN SET stuck_at[sensor_id] = 0 ELSE stuck_at[sensor_id]++ 6. IF reading < profile[sensor_id].MAX THEN SET spike_dict[sensor_id] = 0 ELSE spike_dict[sensor_id]++ ... more ... 7. FOREACH model m in pred_models, DO 8. flag ← predict_fault(m, pkt) 9. IF flag is a fault THEN Increment ft_dict[sensor_id] ELSE SET ft_dict[sensor_id] ← 0 10. END-FOREACH

Table 7
Check Faults Algorithm.

Algorithm: Check-Fault-Time
Input: Dictionary: input_dict < sensor_id, threshold>, Fault Type: fault Steps: 1. DO infinitely 2. current_time ← get_time() 3. FOR-EACH sensor_id in input_dict Do 4. curr_status ← input_dict[sensor_id] 5. Profile ← get_profile(sensor_id) 6. was-violated-flag ← verify that a threshold count, or a time elapsed was not exceeded/violated 7. IF was-violated-flag is true THEN 8. notify-data-center (sensor_id, fault) // SNMP protocol 9. END-IF 10. END-FOREACH 11. Wait for TH seconds 12. END-DO

4.2. WBAN lab dataset

A WBAN consisting of 15 sensors was deployed in the Lab on a human-like topology. Sensor health packets and sensor data packets were collected for 30 days. Sensor health packets are regularly transmitted from each sensor node to the base station to provide status information on the sensor node's performance and connectivity. On the other hand, sensor data packets contain the actual sensor readings. Collected data was preprocessed by applying aggregations, summarization, and transformations. We examined the sensor health packets manually and identified two types of faults: battery and wireless connection failures. Table 8 presents a snapshot of the dataset in which there are three labels, normal (N), battery alarm (BA), and connection alarm (CA). The data set size is 26,364 records distributed as follows: 66.16% N, 31.74% BA, and 2.1% CA. The data indicates sensor and traffic information such as battery lifespan and Mean Time Between Failures (MTBF).

We constructed ANN, DNN, SVM, and DT prediction models to detect faults using TensorFlow, LIBSVM, and Weka libraries. The setup for training the models is presented in Table 9. In all models, the accuracy of the model was verified using a 10-fold-cross-validation technique. In k-Fold cross-validation, the dataset is split into k folds in which k-1 folds are used in training, and the remaining fold is used for testing. We repeat the process k times and report the average accuracy. The training parameters in Table 9 are based on experience and familiarity with the data. We have gone through several rounds of trying different parameters for each technique, and we reported the most effective parameters we found.

In the first set of experiments, Tables 10–13 present the prediction results of all models. All models except SVM have achieved remarkable prediction accuracy, recall, precision, and F1-score (>94%). DT and DNN have achieved zero false alarm rates. High accuracy can be explained due to the nature of the problem and the data. While DT employs induction rules, DNN and ANN have powerful prediction capabilities that depend on long training epochs complex networks. SVM works effectively with binary classification, and dealing with multiclass problems might require calibration and tune-up.

The results in Table 11and Table 13 might have suffered from overfitting. However, we have applied an early stopping method in ANN/DNN (Jabbar and Khan, 2015) and pre-pruning in DT (Bramer, 2007) to avoid overfitting. Additionally, the use of different measurements and 10-Fold cross-validation support our results.

4.3. Thresholding biological dataset

As was explained before, sensor readings can be checked against thresholds to determine their validity. However, when dealing with many sensors where each sensor has its threshold and hardware specifics, using a data mining technique to represent and manage threshold checking becomes necessary. In the second set of experiments, we apply AI on biological and synthesized datasets to detect faults through threshold checking. We use the BIDMC-PPG dataset that has medical readings (Pimentel, 2016). The data set is a collection of heart rate (HR), heart pulse, respiration rate (RESP), and SPO2 rate readings collected from 53 critically ill patients with a sample shown in Table 14. The data set comprises 100,000 readings, where 93.4% of the readings are considered normal, and 6.6% are deemed abnormal.

Additionally, we use synthesis data to emulate real scenarios assuming 500 sensors to measure different biological data for large-scale analysis. For each sensor, labeled reading was generated for training. The distribution of the generated data was 46% as normal readings and 54% as faulty. Table 15 presents a snapshot of the synthesis dataset. In another scenario, we considered that the same threshold might not work for different patients' conditions. For example, the threshold for blood pressure in normal cases is different from after surgery or during labor. Hence, a condition or status attribute was added in the dataset to reflect that the same sensor might have different thresholds depending on certain conditions. Table 16 presents a snapshot of the dataset with different conditions. Notice that eight different medical conditions were considered ranging from normal to critical. In both synthesis datasets, sensor reading and labels are generated based on sensor profiling. We used the same parameters' setup in Table 9 in training. We report the training time, testing accuracy, precision, recall, f1-score, and false alarm rate using a 10-fold cross-validation method for the results.

The results of the BIDMCC-PPG dataset are presented in Table 17. The average 10-fold cross-validation accuracy, precision, recall, and f1-score are >98% and with an inferior false alarm rate. As expected, the average training time is longer in the case of ANN and DNN.

The results of the synthesis datasets are depicted in Fig. 6 and Fig. 7. The figures show the change of accuracy versus the dataset size for DT, SVM, and ANN. The x-axis represents the number of accumulated readings per sensor. The focus here is to achieve higher accuracy with a lower number of training data. Fig. 6 presents the results of the first synthesis dataset. We observe that both SVM and DT performed very well with high accuracy. ANN achieved lower accuracy, possibly due to the need for parameter calibration and longer episodes; however, SVM and ANN were less affected by the dataset's size, while DT performed well when the

Table 8
Dataset snapshot and summary.

Node Id	Health Pkts	Node Pkts	Forwarded	Dropped	Retries	Battery	Parent	RSSI dBm	Label
1	464	4014	1062	156	13	2.7	0	−62	N
15	397	3409	19	123	511	3.1	0	−82	N
4	394	3403	0	97	1	3.1	0	−43	N
7	470	4053	32	155	60	2.9	0	−56	N
5	470	4058	158	170	696	2.6	8	−62	N
8	470	4051	1102	282	64	2.7	0	−64	N
2	470	4064	82	131	61	2.6	0	−54	N
9	470	4049	774	169	95	2.7	0	−64	N
14	470	4038	145	147	223	2.5	0	−72	BA
12	464	3991	1026	158	640	2.5	1	−86	BA
10	470	4047	1400	195	54	2.4	0	−64	BA
13	470	4041	318	398	2153	2.5	0	−80	BA
14	471	4047	145	147	223	2.5	0	−71	BA
12	465	4000	1026	158	641	2.5	1	−88	BA
10	471	4056	1400	195	54	2.4	0	−64	BA
13	471	4049	318	398	2153	2.5	0	−80	BA
12	864	7436	1026	241	1516	2.5	0	−95	CA
11	40	340	0	20	104	2.6	0	−95	CA
14	1935	16,641	2374	594	1361	2.5	5	−95	CA

Table 9
Models' parameters setup.

TensorFlow (ANN)	LIBSVM	Weka (J48)	TensorFlow (DNN)
Hidden Layers = 2	C = 1	Binary, Split = true	Hidden Layers = 10
#Nodes/layer = 16	Kernel = RBF	Unpruned = true	#Nodes/layer = 16
Activation = RELU, Softmax	Type = 1-vs-1	Confidence = 0.55	Activation = RELU, Softmax
Epoch = 100	COEF0 = 0		Epoch = 100
Evaluation = 10-fold Cross Validation			

Table 10
ANN results.

Fault/measure	Precision	Recall	F1-Score
Battery	0.986	0.981	0.983
Wireless Connection	0.946	0.947	0.944
Normal	0.953	0.964	0.959
Overall	0.961	0.964	0.962
Overall Accuracy	0.9727		
False Alarm Rate	0.047		

Table 11
DNN results.

Fault/measure	Precision	Recall	F1-Score
Battery	1	1	1
Wireless Connection	1	1	1
Normal	1	1	1
Overall	1	1	1
Overall Accuracy	1		
False Alarm Rate	0		

Table 12
SVM Results.

Fault/measure	Precision	Recall	F1-Score
Battery	0.956	0.793	0.868
Wireless Connection	0.902	0.991	0.943
Normal	0.939	0.82	0.876
Overall	0.932	0.868	0.895
Overall Accuracy	0.917		
False Alarm Rate	0.145		

data set size was greater than 80. For example, when N = 20 (data set size = 20*500), DT achieved 78% accuracy while SVM and ANN

Table 13
Decision Tree Results.

Fault/measure	Precision	Recall	F1-Score
Battery	1	1	1
Wireless Connection	1	1	1
Normal	1	1	1
Overall	1	1	1
Overall Accuracy	1		
False Alarm Rate	0		

achieved 90% and 80%, respectively. This clearly shows that accuracy is proportional to the number of readings per sensor for DT. However, such proportionality is inferior to SVM and ANN. For example, while a small number of samples is enough for SVM to achieve accuracy higher than 90%, DT requires at least 80 random reading samples per sensor to create an acceptable predictor with high accuracy. We can draw similar conclusions from Fig. 7, which shows training against the second synthesis dataset. However, the accuracy is lower due to the complexity of this synthesis dataset. In terms of training time, Fig. 8 shows that DT is trained faster than SVM and ANN even when the dataset's size is large. This is expected due to SVM and ANN's model complexity, and we obtained similar results for the second synthesis dataset. In both scenarios, SVM and DT prediction models are reliable to be deployed and used in fault prediction. For ANN, more calibration and training are needed.

4.4. PDA prototype

As proof of concept, the PDA subsystem component was developed, as shown in Fig. 1. The component is an Android mobile application that manages the data and faults on the patient's side. Prediction Models are assumed to be trained offline and automatically deployed on the cloud or data center. For this purpose, a GUI-based gadget was also developed to train, test, and deploy prediction models to a cloud database (FireBaseDB). The mobile application can download and use in fault detection. In Fig. 9, the training process can be conducted using SVM, DNN, ANN, or DT. The data scientist can configure and launch the training process. Validation is conducted either by applying 10-fold cross-validation or by providing a validation dataset. After that, the model can be deployed to the cloud database, FireBaseDB (Github.com). In this prototype, we have used training parameters per Table 9 and conducted training on 2/3 of the dataset while 1/3 was used for validation.

Table 14
Snapshot of BIDMC PPG and Respiration data sets.

Sensor ID	HR	PULSE	RESP	SPO2
1	94	93	25	97
2	94	93	25	97
3	77	75	17	96
5	99	98	12	99
6	82	82	20	99

Table 15
Snapshots of synthesis data set – A.

Sensor ID	Reading	Label
159	1.632	Fault
5	−1.34	Fault
480	7.502	Normal
150	2.80	Fault
310	4.142	Fault
450	0.088	Fault
215	4.896	Fault

Table 16
Snapshots of synthesis data set with conditions – B.

Sensor ID	Reading	Condition	Label
54	4.08	After Surgery	Fault
80	4.17	Septic shock	Fault
76	13.6	Anesthesia	Normal
20	−4.4	labor-Mother	Fault
47	11.3	Anesthesia	Normal
66	8.15	Normal	Normal
30	4.78	Labor-Baby	Fault

Table 17
BIDMCC-PPG results.

	DT	SVM	DNN	ANN
Accuracy	0.990	0.990	0.998	0.998
Precision	1	0.99	0.99	0.98
Recall	0.99	0.99	0.99	0.99
F1-score	0.99	0.99	0.99	0.98
Training Time (Seconds)	3	233.1	476	412
False Alarm Rate	0.0	0.01	0.003	0.003

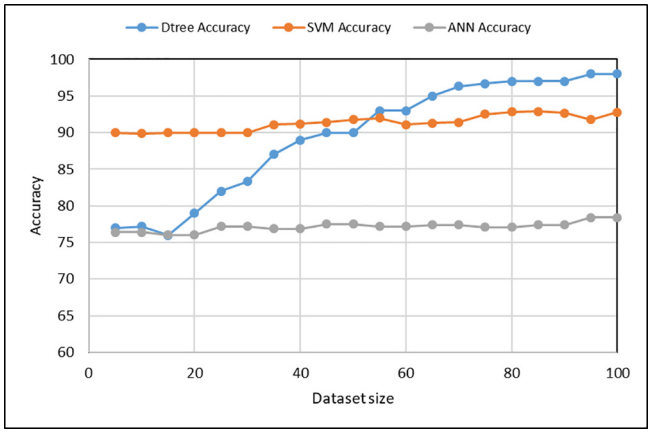

Fig. 6. Accuracy Results of DT, SVM, and ANN (1st synthesis dataset).

The mobile application is designed to accommodate WEKA models, LIBSVM models, and TensorFlow models. Each model might be trained to predict certain kinds of faults. In our prototype, all models were trained to predict four sensor faults and three types of software faults, as shown in Table 18. The design is adaptable to include more faults and predictors with small modifications. The application is also capable of automatically updating its current prediction models. We have deployed the prediction models presented in the results section on the mobile prototype. Fig. 10 depicts the prototype's main functional screenshots: downloading/retrieving prediction models from the cloud, detecting battery fault, and data management capability (sensors profiles and prediction models). All the gadgets projects, including the mobile prototype, are available online (Github.com).

4.5. Framework evaluation

The framework is flexible and adaptable since any new prediction model, new data/trends, or new sensor technology can be incorporated through the FTP-Model and sensor profile. On the other hand, training a new prediction model for a new sensor poses time delays due to data accumulation and training processes.

Our experiments used different technologies in terms of model prediction evaluation, namely, TensorFlow/Python, LIBSVM/Java, and WEKA/Java. Our final findings are that in terms of applicability, performance, and usability, TensorFlow is superior in ANN/DNN training. Regarding accuracy and results (see Tables 10–13 and Figs. 6–8), the nature of the data, pre-processing steps, and tune-up parameters play important roles in the final prediction. We found that all methods with careful tune-up parameters and good cleansing technique performed very well, accuracy >94%, and DT achieved remarkable results.

Data privacy and security are critical in WBAN. This is currently not an issue on the Edge/Cloud side; however, it poses a serious issue on the PDA side. For example, the prediction model can be tampered with by an adversary to behave maliciously. Therefore, encrypted prediction models are used to protect against such attacks. Further investigation of security and privacy issues is necessary, which will be addressed as a future extension of this work.

Usability can be measured by how easy the patient and a data scientist can use the framework. The prototype GUI is simple, with

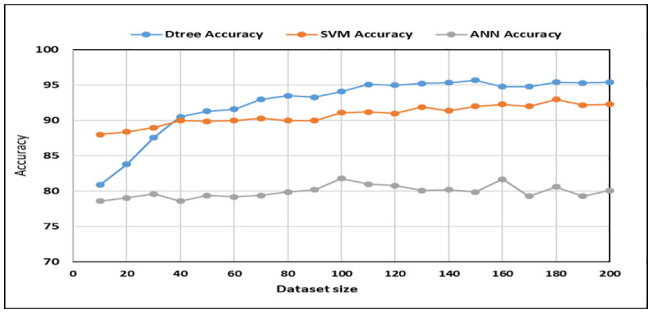

Fig. 7. Accuracy results of DT, SVM, and ANN (2nd synthesis dataset).

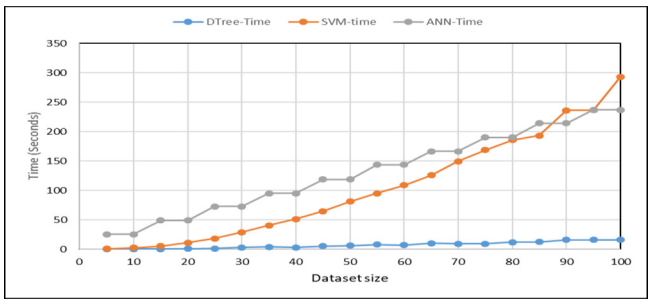

Fig. 8. Training time of DT, SVM, and ANN using 1st synthesis dataset.

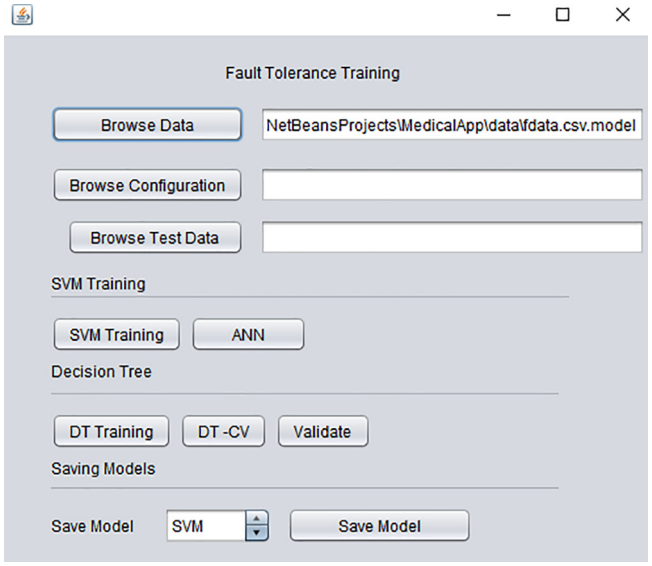

Fig. 9. Training Model Gadget.

Table 18
List of detected faults in our prototype.

Sensor Faults	Software Faults
Wireless Connection Fault	Connection Lost
Battery Fault	Malformed Prediction model
Stuck at zero	Dis-functioning Prediction model
Spiked value	

minimal interaction needed from the user and few needed settings. However, advanced settings can be used by the maintenance engineer as needed.

The prototype needs 25 MB for storage (Github.com), and we assume a customized PDA where unnecessary services and applications can be shut down. The current hardware/storage technologies are sufficient for the processing and persistence of traffic on the PDA. Traffic is buffered and forwarded to the edge/-cloud site. Although connections are very reliable, the PDA can hold buffered traffic or forward the traffic to a local base station that serves as a backup node in case of communication interruption. Additionally, such interruption will be detected by the Edge/Cloud site; an alert will be produced and handled by an on-duty engineer to resolve that.

Finally, we have addressed software and hardware failures for the sensors, PDA, and software components in terms of reliability, Table 18. The Simple Network Management Protocol (SNMP) is used to alert the Edge/Cloud side. Table 19 presents a summary of the strengths and weaknesses of our framework.

Table 19
Framework Evaluation Summary.

Criteria	Strengths	Weaknesses
Flexibility	Prediction models and sensor profile updates	Training time delay
Security & privacy	Encrypted prediction model	Not studied comprehensively
Usability	Simple GUI	Update of the App code requires manual intervention.
Reliability	A lightweight protocol, SNMP handles alerts. Many fault types are covered, including software/ hardware.	Undetected software bugs
Performance	Not an issue due to the current hardware advancement	Platform dependency

5. Conclusions

In this paper, we explored the different faults in a WBAN medical monitoring application. We proposed a framework for WBAN sensor faults and traffic management. The framework incorporates data management and machine learning-based fault detection. The framework is flexible to utilize different machine learning techniques that can be updated and deployed to detect new faults. In addition, the framework can accommodate different

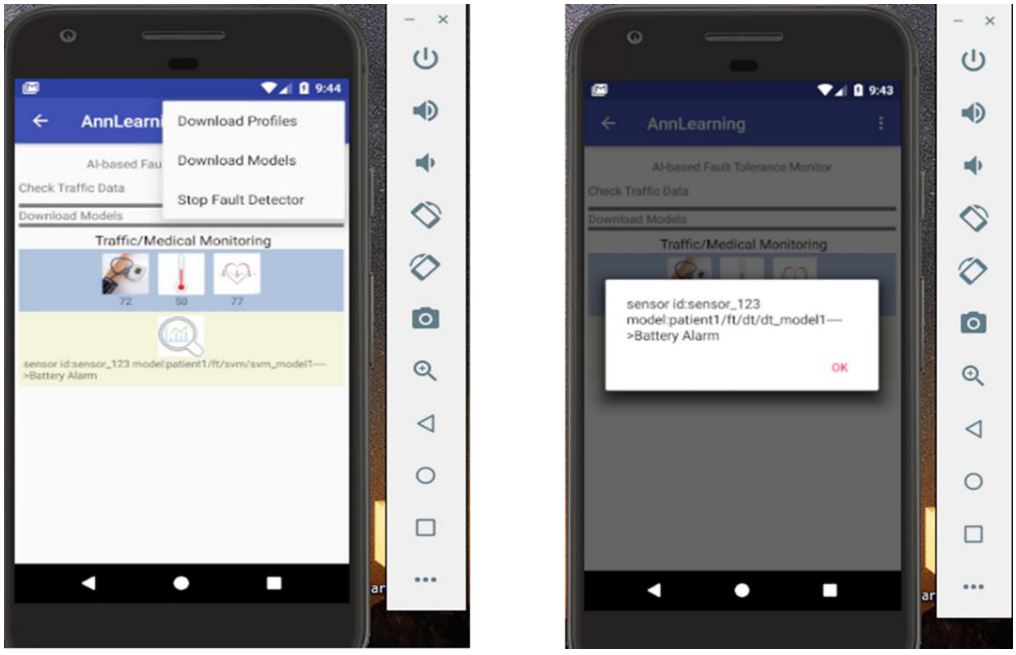

Fig. 10. Prototype Screenshots.

sensors through profiling. We investigated the use of four AI algorithms in fault prediction. We trained SVM, DT, ANN, and DNN on traffic data to predict battery and wireless connection faults. We presented the results in terms of several standard measurements and confusion matrix. Additionally, we proposed a threshold-based detection approach using ML and presented high accuracy results using DT, ANN, DNN, and SVM for the biomedical and synthesis datasets. The outcomes of our experiments showed that the nature of the data, pre-processing steps and tune-up parameters play important roles in the final prediction. We found that all methods with careful tune-up parameters and good cleansing techniques performed very well, accuracy >94%. We developed and presented a prototype for the training process, model deployment, and model prediction as proof of concept. We presented a mobile application that can detect several sensor and software faults. Our future research will be extended to explore the software side of the WBAN applications and examine additional software faults and techniques. For example, progressive degradation of fault discovery and PDA resource usage is worth investigating. Moreover, we will incorporate security and data privacy aspects into the PDA. Finally, we plan to conduct larger-scale management of prediction models on the data center to uncover additional faults.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research is funded by the United Arab Emirates University research grant number 31T078.

References

Academy, C., 2013. Network Basics Companion Guide, Video Enhanced Edition. Cisco Press.
Al Thawadi, M., Sallabi, F., Awad, M., Shuaib, K., 2019. Disruptive IoT-Based Healthcare Insurance Business Model. In: 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp. 397–403.
Al-Turjman, F., Baali, I., 2019. Machine learning for wearable IoT-based applications: a survey. Trans. Emerg. Telecommun. Technol 1–16. https://doi.org/10.1002/ett.3635.
Barakah, D.M., Ammad-uddin, M., 2012. A survey of challenges and applications of wireless body area network (WBAN) and role of a virtual doctor server in existing architecture. In: Third International Conference on Intelligent Systems Modelling and Simulation, pp. 214–219.
Bramer, M., 2007. Avoiding overfitting of decision trees. Principles of data mining, 119–134.
Fu, G., Yi, L., Pan, J., 2019. Tuning model parameters in class-imbalanced learning with precision-recall curve. Biometrical Journal 61 (3), 652–664.
Gao, Z., Cecati, C., Ding, S.X., 2015. A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches. IEEE Trans. Ind. Electron. 62 (6), 3757–3767.
Gia, T.N., Rahmani, A.-M., Westerlund, T., Liljeberg, P., Tenhunen, H., 2015. Fault tolerant and scalable IoT-based architecture for health monitoring. In: IEEE Sensors Applications Symposium (SAS), pp. 1–6.
"Github.com Available online." https://github.com/mxawad2000/WBAN-Mobile-Prototype.
Hartmann, M., Hashmi, U.S., Imran, A., 2019. Edge computing in smart health care systems: review, challenges, and research directions. Trans. Emerg. Telecommun. Technol. no. July, 1–25. https://doi.org/10.1002/ett.3710.
Hassanalieragh, M. et al., 2015. Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges. In: IEEE International Conference on Services Computing, pp. 285–292.
Jabbar, H., Khan, R.Z., 2015. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices, pp. 163–172.
Jeong, Y.-S., Kim, H.-W., Park, J.H., 2014. Visual scheme monitoring of sensors for fault tolerance on wireless body area networks with cloud service infrastructure. Int. J. Distrib. Sens. Networks 10, (4) 154180.
Li, X., Deng, Y., Ding, L., 2008. Study on precision agriculture monitoring framework based on WSN. In: 2008 2nd International Conference on Anti-counterfeiting, Security and Identification, pp. 182–185.
Li, X.i., Ji, H., Li, Y.i., 2013. Layered fault management scheme for end-to-end transmission in internet of things. Mob. Networks Appl. 18 (2), 195–205.
Liao, Y., Yu, Q., Han, Y., Leeson, M.S., 2018. Relay-enabled task offloading management for wireless body area networks. Appl. Sci. 8 (8), 1409.
Mahapatro, A., 2011. Online fault detection and recovery in body sensor networks. In: World Congress on Information and Communication Technologies, pp. 407–412.
Mehmood, G., Khan, M.Z., Abbas, S., Faisal, M., Rahman, H.U., 2020. An energy-efficient and cooperative fault-tolerant communication approach for wireless body area network. IEEE Access 8, 69134–69147.
Peng, Y., Wang, X., Guo, L., Wang, Y., Deng, Q., 2017. An efficient network coding-based fault-tolerant mechanism in WBAN for smart healthcare monitoring systems. Appl. Sci. 7 (8), 817.
Pimentel, M.A.F. et al., 2016. Toward a robust estimation of respiratory rate from pulse oximeters. IEEE Trans. Biomed. Eng. 64 (8), 1914–1923.
Salayma, M., Al-Dubai, A., Romdhani, I., Nasser, Y., 2017. Wireless body area network (WBAN) a survey on reliability, fault tolerance, and technologies coexistence. ACM Comput. Surv. 50 (1), 1–38.
Sallabi, F., Naeem, F., Awad, M., Shuaib, K., 2018. Managing IoT-based smart healthcare systems traffic with software defined networks. In: International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6.
Sawand, A., Djahel, S., Zhang, Z., Nait-Abdesselam, F., 2015. Toward energy-efficient and trustworthy eHealth monitoring system. China Commun. 12 (1), 46–65.
Sharma, A.B., Golubchik, L., Govindan, R., 2010. Sensor faults: detection methods and prevalence in real-world datasets. ACM Trans. Sens. Networks 6 (3), 1–39.
Shawe-Taylor, J., Cristianini, N., 2000. Support vector machines, vol. 2. Cambridge University Press Cambridge.
Tavera, C. A., Ortiz, J. H., Khalaf, O. I., Saavedra, D. F., & Aldhyani, T. H. (2021). Wearable Wireless Body Area Networks for Medical Applications. Computational and Mathematical Methods in Medicine, 2021.
Wang, L., Xu, B., Cai, H., Zhang, P., 2020. Context-aware emergency detection method for edge computing-based healthcare monitoring system. Trans. Emerg. Telecommun. Technol. May, 1–18. https://doi.org/10.1002/ett.4128.
Yang, Y., Liu, Q., Gao, Z., Qiu, X., Meng, L., 2015. Data fault detection in medical sensor networks. Sensors 15 (3), 6066–6090.
S. O. N. G. Yan-yan, L.U. Ying, Decision tree methods: applications for classification and prediction, Shanghai Arch. Psychiatry, vol. 27, no. 2, pp. 130–135, 2015, [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/.
Yi, C., Cai, J., 2018. Transmission management of delay-sensitive medical packets in beyond wireless body area networks: A queueing game approach. IEEE Trans. Mob. Comput. 17 (9), 2209–2222.
Zhang, R., Peng, Z., Wu, L., Yao, B., Guan, Y., 2017. Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence. Sensors 17 (3), 549.
Zhang, T., Zhao, Q., Nakamoto, Y., 2017. Faulty sensor data detection in wireless sensor networks using logistical regression. In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 13–18.
Zhou, S., Lin, K.-J., Na, J., Chuang, C.-C., Shih, C.-S., 2015. Supporting service adaptation in fault tolerant internet of things. In: 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA), pp. 65–72.